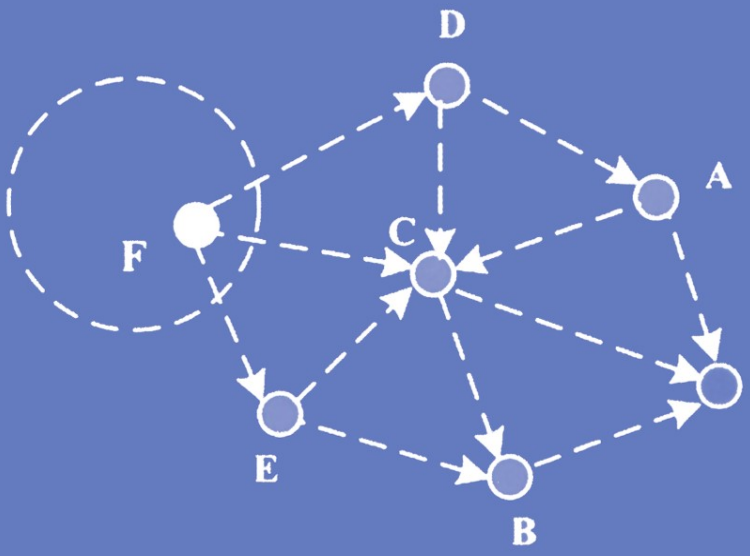
Hot Topics

LNCS 3313

Claude Castelluccia
Hannes Hartenstein
Christof Paar
Dirk Westhoff (Eds.)

Security in Ad-hoc and Sensor Networks

First European Workshop, ESAS 2004
Heidelberg, Germany, August 2004
Revised Selected Papers



Springer

Lecture Notes in Computer Science 3313

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Claude Castelluccia Hannes Hartenstein
Christof Paar Dirk Westhoff (Eds.)

Security in Ad-hoc and Sensor Networks

First European Workshop, ESAS 2004
Heidelberg, Germany, August 6, 2004
Revised Selected Papers

Springer

Volume Editors

Claude Castelluccia
INRIA, Unité de Recherche Rhône-Alpes, France
E-mail: ccastell@ics.uci.edu

Hannes Hartenstein
Universität Karlsruhe (TH), Computing Center and Institute of Telematics
E-mail: hartenstein@rz.uni-karlsruhe.de

Christof Paar
Ruhr-Universität Bochum, Communication Security
44780 Bochum, Germany
E-mail: cpaar@crypto.rub.de

Dirk Westhoff
NEC Europe Ltd., Network Laboratories
Kurfürsten Anlage 36, 69115 Heidelberg, Germany
E-mail: dirk.westhoff@netlab.nec.de

Library of Congress Control Number: 2004117659

CR Subject Classification (1998): E.3, C.2, F.2, H.4, D.4.6, K.6.5

ISSN 0302-9743
ISBN 3-540-24396-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11377689 06/3142 5 4 3 2 1 0

Claude Castelluccia, Hannes Hartenstein, Christof Paar,
Dirk Westhoff (Eds.)

Security in Ad Hoc and Sensor Networks

First European Workshop, ESAS 2004
Heidelberg, Germany, August 6-6, 2004

Preface

Ad hoc and sensor networks are making their way from research to real-world deployments. Body and personal-area networks, intelligent homes, environmental monitoring or intervehicle communications: there is almost nothing left that is not going to be "smart" and "networked." While a great amount of research has been devoted to the pure networking aspects, ad hoc and sensor networks will not be successfully deployed if security, dependability and privacy issues are not addressed adequately. These issues are very important because ad hoc and sensor networks are usually used for very critical applications. Furthermore, they are very vulnerable because they are, most of the time, deployed in open and unprotected environments.

At ESAS 2004, researchers with interests in both networking and security came together to present and discuss the latest ideas and concepts in the design of secure, dependable and privacy-preserving ad hoc and sensor networks. In the keynote speeches, Jean-Pierre Hubaux (EPFL, Switzerland) discussed the challenges of ad hoc network security, and Antonis Galetsas (European Commission, DG Information Society) presented the current and future activities of the European Commission on these topics.

Out of 55 high-quality submissions, the program committee selected 17 papers for publication. The program covered the full spectrum of security-related issues, including key distribution and management, authentication, energy-aware cryptographic primitives, anonymity/pseudonymity, secure diffusion, secure P2P overlays and RFIDs.

We would like to thank all authors, referees, supporters and workshop participants for making this workshop a successful event. Special thanks to the program committee and further reviewers for their great work and for reviewing the papers in less than 4 weeks. We hope that you will enjoy the ESAS proceedings and your research work will be stimulated.

September 2004

Claude Castelluccia
Hannes Hartenstein
Christof Paar
Dirk Westhoff

Committees

Program Co-chairs

Claude Castelluccia, INRIA, France
Christof Paar, University of Bochum, Germany
Hannes Hartenstein, University of Karlsruhe, Germany
Dirk Westhoff, NEC Europe Ltd., Germany

Program Committee

Nadarajah Asokan, Nokia, Finland
Levente Buttyan, BME-HIT, Hungary
Sonja Buchegger, EPFL, Switzerland
Claudia Eckert, TU Darmstadt, Germany
Stefan Lucks, University of Mannheim, Germany
Refik Molva, Eurécom, France
Gabriel Montenegro, SunLabs, France
Pekka Nikander, Ericsson, Finland
Panagiotis Papadimitratos, Cornell University, USA
Ahmad-Reza Sadeghi, University of Bochum, Germany
Frank Stajano, University of Cambridge, UK
Gene Tsudik, UC Irvine, USA
Andre Weimerskirch, University of Bochum, Germany
Nathalie Weiler, ETH Zuerich, Switzerland
Susanne Wetzel, Stevens Institute of Technology, USA
Manel Guerrero Zapata, University Pompeu Fabra, Barcelona, Spain

Table of Contents

New Research Challenges for the Security of Ad Hoc and Sensor Networks

Jean-Pierre Hubaux

EPFL

Abstract. In this talk, we provide an overview of the current and upcoming research challenges for the security of ad hoc and sensor networks. We begin with the crucial problem of key establishment; we explain how mobility can be exploited to set up security associations between nodes, and we address the challenges of key setup in sensor networks. We also provide an overview of the security of routing protocols. We explain how two nodes getting in power range of each other can prove this event to a third party at a later stage.

We then address cooperation between wireless nodes, and show that this problem naturally leads to the prevention of greedy behavior in WiFi hot spots; we detail our solution to this problem, called DOMINO. We then address a very novel problem, namely the *secure* location of a node; we explain the potential of this feature, taking the examples of the secure location of smart vehicles in road traffic and the prevention of attacks against sensor networks positions. We show how this feature can be implemented by an appropriate combination of distance bounding and multilateration.

The slides of the talk are available at http://lcawww.epfl.ch/hubaux/

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, p. 1, 2005.

Public Key Cryptography in Sensor Networks—Revisited*

Gunnar Gaubatz, Jens-Peter Kaps, and Berk Sunar

Department of Electrical & Computer Engineering,
Worcester Polytechnic Institute,
100 Institute Road, Worcester, MA 01609, U.S.A.
{gaubatz, kaps, sunar}@wpi.edu

Abstract. The common perception of public key cryptography is that it is complex, slow and power hungry, and as such not at all suitable for use in ultra-low power environments like wireless sensor networks. It is therefore common practice to emulate the asymmetry of traditional public key based cryptographic services through a set of protocols [1] using symmetric key based message authentication codes (MACs). Although the low computational complexity of MACs is advantageous, the protocol layer requires time synchronization between devices on the network and a significant amount of overhead for communication and temporary storage. The requirement for a general purpose CPU to implement these protocols as well as their complexity makes them prone to vulnerabilities and practically eliminates all the advantages of using symmetric key techniques in the first place. In this paper we challenge the basic assumptions about public key cryptography in sensor networks which are based on a traditional software based approach. We propose a custom hardware assisted approach for which we claim that it makes public key cryptography feasible in such environments, provided we use the right selection of algorithms and associated parameters, careful optimization, and low-power design techniques. In order to validate our claim we present proof of concept implementations of two different algorithms—Rabin's Scheme and NtruEncrypt—and analyze their architecture and performance according to various established metrics like power consumption, area, delay, throughput, level of security and energy per bit. Our implementation of NtruEncrypt in ASIC standard cell logic uses no more than 3,000 gates with an average power consumption of less than 20 μW. We envision that our public key core would be embedded into a light-weight sensor node architecture.

1 Introduction

Wireless *distributed sensor networks* (DSN) are expected to be used in a wide range of applications, from monitoring wildlife and collecting microclimate data

* This material is based upon work supported by the National Science Foundation under Grants No. ANI-0133297 (NSF CAREER Award) and No. ANI-0112889.

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 2–18, 2005.

[2, 3, 4] to a number of military applications like target tracking [5] and detection of biological or chemical weapons. The current generation of wireless sensor nodes is still relying on batteries as its source of power. The limited lifetime of batteries, however, significantly impedes the usefulness of such devices since maintenance accesses would become necessary whenever the battery is depleted. Furthermore, the intention of having large amounts of tiny nodes scattered over a large area would render maintenance impractical. Next generation sensor nodes will therefore combine ultra-low power circuitry with so-called *power scavengers*, which allow for maintenance-free operation of the nodes. This opens up a whole new range of applications where the nodes can be placed in inaccessible locations.

Power scavengers are devices able to harvest small amounts of energy from ambient sources such as light, heat or vibration. This energy is stored in a capacitor and can be used to power the sensor node either continuously, for small amounts of power, or in intervals if the demand is higher. At least $8\mu W$ of power can be generated using MEMS-based power scavengers, as reported in [6]. Other larger systems are able to generate much more power [7], but these are typically not integrated on-chip with the actual sensor node. It is expected that future MEMS-based scavengers will be able to deliver power up to $20\mu W$ continuously.

Due to the sensitive nature of many of the anticipated applications of DSN, a certain minimum level of secure communication between sensor nodes and base station is required. This includes data confidentiality and integrity. Both can be provided through encryption of the data. One of the biggest problems in using secret key algorithms—apart from their size and scalability issues—is the protection of the sensitive key material. Sensor nodes might be deployed in an untrusted environment, e.g. for military applications. The capture of a single node by an adversary should not jeopardize the integrity of the entire network. In a setting where the sensor nodes send encrypted data to a base station a public key scheme is of great advantage as here each node contains only public key material not private. Previously proposed security protocols such as SNEP and μTESLA [1] provide secure authentication using only symmetric key techniques. In order to provide authentication to insecure nodes μTESLA has to emulate asymmetry through a delayed disclosure of symmetric keys. While Carman et al. acknowledge in [8] that symmetric key techniques are attractive due to their energy efficiency, they also conclude that all symmetric key based key exchange protocols analyzed by them exhibit limitations in their flexibility. The emulation of an asymmetric cryptographic primitive requires that each node is time synchronized with the base station and has key management functions and ample storage. As the symmetric keys are revealed sequentially over time, nodes might have to store multiple messages before they can be authenticated. This broadcast authentication scheme also implies that the keys shared among all nodes need to be updated in regular intervals, requiring broadcasts from the base station to all nodes. As in many settings the base station can not directly communicate with all nodes, these keys need to be forwarded from node to node. This protocol overhead leads to increased energy consumption of the nodes as

keys and key management messages need to be transmitted frequently. Complex key management and high storage requirements for multiple keys and messages put a considerable burden on the power consumption of the nodes. The use of public key cryptography would eliminate the need for complicated protocols and at the same time would also increase the security of the entire system, since only the public key of the base station would have to be embedded into the nodes.

The challenge is to overcome the considerable computational complexity of standard public key encryption algorithms and make public key encryption possible in self powered sensor nodes. Traditional schemes like RSA or ElGamal require considerable amounts of resources which in the past limited their use to large-scale platforms like networked servers and personal computers. Mobile equipment with less computational resources, such as cell phones, Personal Digital Assistants (PDAs) and pagers, therefore uses much more efficient elliptic curve based algorithms such as EC-DH and EC-DSA which execute considerably faster while preserving the same level of security [9]. The operands of EC-cryptosystems are much shorter than those in traditional schemes. Unfortunately the improved computational efficiency of ECC comes at the price of much more complex arithmetic primitives and a large number of temporary operands, whereas RSA or ElGamal require only one single arithmetic primitive and few operands. The heterogenous structure and larger storage requirements of ECC make it less scalable and in effect less attractive for energy efficient low-power implementations.

In this paper we compare two architectures that implement two different types of public key crypto-systems with promising characteristics. The first one, Rabin's Scheme [10], is a specialization of the well known RSA algorithm [11] where the exponent is fixed to the value 2. As with RSA, the security of Rabin's scheme relies on the hard problem of factoring large integers. The second algorithm, NtruEncrypt [12], was introduced in 1996 by Hoffstein, Pipher and Silverman. NtruEncrypt is a public key cryptosystem where security is based on the hardness of the Shortest Vector Problem (SVP) in a very high dimension lattice. It still uses relatively large operands, but it reduces the overall asymptotic complexity of the encryption operation to $O(n^2)$ compared to RSA's $O(n^3)$. In both cases we concentrate on the encryption operation only. The decryption of the sensor data would be performed by the more powerful base station. We analyze the performance of these architectures by means of various established metrics in the field of computer organization, like power consumption, area, delay, throughput and latency. We also include some that are not as commonly encountered, such as level of security and energy per bit encrypted. We demonstrate that ultra-low power implementations of public key cryptography are feasible. Our interest, however, is mainly focused on the computational aspects of the underlying arithmetic primitives and as such we refrain from deeper discussion of protocol issues and the cryptographic services that need to be provided by these systems.

The remainder of the paper is structured as follows. After a brief description of the cryptosystems in Section 2, and an introduction into low-power design

techniques 3, we will focus on their application for implementing the algorithms in Section 4. In Section 5 a brief definition of the metrics of interest is followed by an extensive comparative analysis of our two architectures. The final section concludes our findings and points out directions for future work.

2 Preliminaries

Rabin's Scheme and NtruEncrypt are two very different public key algorithms. In this section we first describe the selection of the algorithm specific parameters to make them comparable. Then we give a brief overview of their function.

2.1 Parameter Selection

In order to better compare these two algorithms of disparate properties and parameter sets, we chose system parameters of both algorithms to offer a closely matching level of security. For definition of this level we refer to the widely recognized definition of equivalent security by Lenstra and Verheul [13]. Amongst others they cover RSA as the principal example for cryptosystems where security is based on the *Integer Factorization Problem*, which is also the basis for Rabin's Scheme. Their analysis, however, does not include a definition of equivalent security for a lattice based scheme like NtruEncrypt. For our purposes we therefore refer to the analysis of Hoffstein, Silverman and Whyte [14], who present a similar evaluation of NtruEncrypt's security level, also in terms of equivalent security.

While in practice certain classes of applications might require a higher level of security than others, we regard our designs simply as a proof of concept and hence chose to implement them at a comparatively low level of security. It should, however, be relatively straightforward to estimate the cost of higher security level implementations based on the analysis that we give at the end of this paper. For Rabin's Scheme we selected an operand size of 512 bits, which according to Lenstra and Verheul [13] provides a security level of around 60 bits. In the case of NtruEncrypt we chose the system parameters as $(N, p, q) = (167, 3, 128)$, based on findings in [14], offering a security level of 57 bits.

2.2 Rabin's Scheme

Rabin's Scheme was introduced in 1979 in [10]. It is based on the factorization problem of large numbers and is therefore similar to the security of RSA with the same sized modulus. Rabin's Scheme has asymmetric computational cost. The encryption operation is extremely fast, however decryption times are comparable to RSA of the same modulus. This asymmetry makes Rabin's Scheme especially interesting for our application. Here is a brief description of the Rabin's Scheme. For a more detailed description and the mathematical proofs see [10][15].

Key Generation.

1. Choose two large random strong prime numbers.
2. Compute $n = p \cdot q$.
3. Pick a random number b for which $0 \leq b < n$.
4. The public key is (n, b), the private key is (p, q).

Encryption.

1. Represent the message as an integer x for which $0 \leq x < n$
2. Compute the ciphertext $\mathrm{E}_{n,b}(x) \equiv x(x + b) \mod n$, as defined in [10]

Only the public key n, b is required for encryption. If we fix b to 0 then $\mathrm{E}_{n,b}(x)$ becomes a simple squaring operation $\mathrm{E}_n(x) = x^2 \mod n$. Rabin's Scheme requires only one squaring, whereas RSA requires several squarings and multiplications for encryption. Therefore encryption with Rabin's Scheme is several hundreds of times faster than RSA [11].

Decryption. involves finding the four square roots x_1, x_2, x_3, and x_4 of $c = \mathrm{E}_n(x) \equiv x^2 \mod n$. Certain simplifications are possible if $p \equiv q \equiv 3 \mod 4$. We would like to point the interested reader to [15] for a complete description of these algorithms. A hardware implementation of the decryption function is certainly feasible but beyond the scope of this paper.

2.3 The NtruEncrypt Public Key Cryptosystem

NtruEncrypt is a relatively new cryptosystem that claims to be highly efficient and particularly suitable for embedded applications such as smart cards or RFID tags, while providing a level of security comparable to that of other established schemes, in particular RSA. While it has not yet received the same level of scrutiny for establishing its resistance to cryptanalysis, there is evidence for efficiency in the simplicity of its underlying arithmetic. In this section we briefly describe the basic setup of NtruEncrypt and its operations. For more in-depth descriptions of the mathematical properties of NtruEncrypt we refer to [12, 16].

NtruEncrypt is based on arithmetic in a polynomial ring $R = \mathbb{Z}(x)/((x^N - 1), q)$ set up by the parameter set (N, p, q) with the following properties:

- All elements of the ring are polynomials of degree at most $N - 1$, where N is prime.
- Polynomial coefficients are reduced either mod p or mod q, where p and q are relatively prime integers or polynomials.
- p is considerably smaller than q, which lies between $N/2$ and N.
- All polynomials are univariate over the variable x.

Multiplication in the ring R is sometimes referred to as "Star Multiplication" based on use of an asterisk $\circledast$ as the operator symbol. It can be best described

as the discrete convolution product of two vectors, where the coefficients of the polynomials form vectors in the following way:

$$\begin{aligned} a(x) &= a_0 + a_1 x + a_2 x^2 + \ldots + a_{N-1} x^{N-1} \\ &= (a_0, a_1, a_2, \ldots, a_{N-1}) \end{aligned}$$

Then the coefficients c_k of $c(x) = a(x) \circledast b(x) \bmod q, p$ are each computed as the summation of partial products $a_i b_j$ with $i + j \equiv k \bmod N$. The modulus for reduction of each coefficient c_k of the resulting polynomial is either q for Key Generation and Encryption, or p for Decryption, as briefly described below. A thorough description of these procedures along with an initial security analysis can be found in [12].

Key Generation. The following steps generate the *private key* $f(x)$:

1. Choose a random polynomial $F(x)$ from the ring R. $F(x)$ should have small coefficients, i.e. either binary from the set $\{0, 1\}$ (if $p = 2$) or ternary from $\{-1, 0, 1\}$ (if $p = 3$ or $p = x + 2$ [16, 17]).
2. Let $f(x) = 1 + pF(x)$ [1].

The *public key* $h(x)$ is derived from $f(x)$ in the following way:

1. As before, choose a random polynomial $g(x)$ from R.
2. Compute the inverse $f^{-1}(x) \pmod q$.
3. Compute the public key as $h(x) = g(x) \circledast f^{-1}(x) \pmod q$.

Encryption.

1. Encode the plaintext message into a polynomial $m(x)$ with coefficients from either $\{0, 1\}$ or $\{-1, 0, 1\}$.
2. Choose a random polynomial $\phi(x)$ from R as above.
3. Compute the ciphertext polynomial $c(x) = p\phi(x) \circledast h(x) + m(x) \pmod q$.

Decryption.

1. Use the private key $f(x)$ to compute the message polynomial $m'(x) = c(x) \circledast f(x) \pmod p$.
2. Map the coefficients of the message polynomial to plaintext bits.

3 Low-Power Design

This section provides a brief introduction to Low-Power Design. The power dissipation in CMOS devices can be summarized by the following equation [18]:

[1] It is not strictly necessary to construct $f(x)$ in this way, but it is recommended in order to decrease the decryption failure rate. It is important, however, that $f(x)$ be invertible $(\bmod p)$ and $(\bmod q)$.

$$P = \underbrace{\left(\frac{1}{2} \cdot C \cdot V_{\mathrm{dd}}^2 + Q_{\mathrm{sc}} \cdot V_{\mathrm{dd}}\right) \cdot f \cdot N}_{P_{\mathrm{dyn}}} + \underbrace{I_{\mathrm{leak}} \cdot V_{\mathrm{dd}}}_{P_{\mathrm{leak}}} \tag{1}$$

The term P_{dyn} represents the dynamic power dissipated during circuit activity. Circuit capacitance C, short-circuit charge Q_{sc} and supply voltage V_{dd} are technology dependent parameters [18] outside of our influence. The switching activity N and operating frequency f, however, can be influenced, and thus minimized, by architectural decisions. The second term P_{leak} represents the static power dissipation due to the leakage current I_{leak}. The leakage current is directly determined by the number of gates and the process technology. For more information about low-power design see [19]. Since we are using a standard cell based design flow, transistor level circuit optimizations are outside of the scope of this paper. In order to minimize the power consumption, we optimized our gate level design according to the following rules:

- The number of transitions ('0' to '1' and '1' to '0') has to be minimal.
- The circuit size should be minimized.
- Glitches cause unnecessary transitions and therefore should be avoided.

Our work is focused on the architectural aspects of low-power design, not on any specific VLSI techniques. Our architectures are implemented using a common CMOS standard cell design flow: circuit specification in structural VHDL, functional RT level simulation (ModelSim), synthesis (DesignCompiler Ultra, TSMC 0.13μm standard cell library), power optimization using annotated switching activity and delay information (DesignCompiler, PowerCompiler and ModelSim), and power analysis (PrimePower, back-annotated wire capacitances).

The TSMC library we use is fully characterized for timing and power consumption and includes several different wireload models for worst case estimation of interconnect capacitances. We would like to stress at this point that, although we use a low-voltage library, it is not in any way optimized for low-power designs.

4 Implementation

In order to provide a common ground for both implementations we had to make certain assumptions about the application scenario, which we state in the following paragraphs. Subsequently we describe the specifics of both implementations.

4.1 Assumptions

Sensor networks typically consist of a number of tiny nodes communicating with a base station [1]. The base station collects the data from the sensors and communicates with the outside world. The sensor nodes have only limited power and can therefore only communicate directly with nodes in close vicinity. They

establish a routing tree with the base station at its root. The base station is assumed to have sufficient power for all computations and communications with the nodes and the outside world. Based on this setting we made the following assumptions:

- As stated in the introduction, we only consider the encryption operation of both systems. The purpose of this paper is to show that public key cryptography is computationally feasible in this environment.
- Depending on the exact application scenario it might be possible to fix the public key to a constant value. This is extremely beneficial for ultra-low power implementation, since the key can be embedded statically and does not require costly storage elements. In our implementations the public key is either hardwired or realized as a look-up table in combinational logic.
- Power consumption and energy efficiency are two different things. Depending on the actual application scenario one might want to trade off the two metrics differently over one another.

4.2 Rabin's Scheme

We have shown in Section 2.2 that the basic function for encryption in Rabin's Scheme is a simple squaring operation $E_n(x) = x^2 \mod n$, if we set $b = 0$. Squarers are a special form of multiplier. While any multiplier can be used to compute the square of a number, special-purpose squarers usually require significantly less hardware and are faster [20] by exploiting the symmetry of the squaring operation.

Squarers can be implemented in many ways. As our main concern is to conserve power we chose a bit-serial approach. The main advantage of a bit-serial design is that it minimizes the number of gates and reduces wire lengths—all factors that are of concern with regards to the circuit's power consumption. The bit-serial approach is ideal for modular reduction. Using the most significant bit (MSB) first method, modular reduction can be performed elegantly after each partial product addition. The generation of the partial product sequence, however, requires an extra 512-bit register. This is very expensive in terms of area and leakage power as each flip-flop is the equivalent of 6 gates. Therefore, we implemented the squarer as a bit serial modular multiplier where multiplicand and multiplier are hard-wired to the same input. All 512 bits of input are available in parallel at the same time. As a multiplier does not take advantage of the symmetry in squaring we expect it to consume more switching power. However, due to its smaller footprint the leakage power is also greatly reduced. At the low clock frequencies commonly encountered in sensor nodes, the influence of leakage power is the dominant part. An additional advantage of this approach is that this unit can easily be converted to a full multiplier for an implementation of RSA or a similar algorithm.

Figure 2 shows the architecture of our squarer. It is a standard bit serial multiplier design comprised of a Left Shift Register, a Bit Multiplier, a Left Shift unit, and the main units Adder and Sum Register. In order to perform modular multiplication we added two multiplexers which toggle the input of the adder between

the next partial product and the 2's complement of the modulus n (reduction). The control logic determines whether a reduction operation is necessary after an addition. Since we are using the same adder for both functions, the number of clock cycles needed for one squaring is data dependent and at most 1024.

The most complex part of the squarer is the Adder. There are two basic adder designs that are suitable for a low power implementation, namely carry-save adder and ripple-carry adder. A ripple-carry adder uses fewer gates and hence consumes the least amount of leakage power, but as the worst case carry chain is the longest, this adder also has the longest delay. The propagation of carries causes glitches which in turn cause a very high dynamic power consumption. A carry-save adder on the other hand propagates carries only by one position, hence there are no glitches, resulting in insignificant amounts of delay and dynamic power consumption. Its disadvantage is that the result is kept in redundant carry-save representation which requires 512 additional flip-flops. This in turn causes a higher consumption of leakage power. Since partial products and complements of the modulus can be accumulated in redundant form, the final non-redundant result needs to be computed only at the very end of the multiplication which takes 512 additional clock cycles.

Neither of both approaches seems optimal for this implementation, so we tried to strike a balance between power and speed. For our adder we are using a ripple-carry adder and insert a carry-save bit on every 8th bit position. Hence the carries ripple for a maximum of 8 bits causing some glitching but significantly less than a full ripple-carry adder would. The dynamic power consumption is therefore much lower than for a full ripple-carry adder. This adder also needs only 64 additional flip-flops to store the carry bits, which is 448 flip-flops less than necessary for a full carry-save adder. This approach, however, introduces a new difficulty. After adding a partial product to the sum, the result has to be shifted. This would misalign the saved carry bits [2]. Hence, carry bits need to be re-aligned before shifting the sum. This is done by adding the carry bits to the sum in the appropriate position and saving the carry bits at the new position. The cost for this is a 512 bit multiplexer, 512 additional clock cycles and a slightly more complex control logic.

The Control logic is comprised of two state machines and one counter. The counter is implemented as a Linear Feedback Shift Register (LFSR) and "counts" up to 512. LFSRs have reduced switching activity and are faster than regular counters, hence reducing the effects on the critical path delay. Furthermore it is clock gated and can be reset. The counter is used to count all the multiplication steps and also to count the worst case number of steps necessary to ripple all 64 carry-save flip-flops. The main state machine of this control logic keeps track of the overall operation of the circuit. The second state machine takes care of arithmetic operations of the circuit. Furthermore it is responsible for the clock gating of the counter and the left shift register (see Figure 2).

[2] This problem does not occur when a full carry-save adder is being used as there is one carry bit associated with every bit position

4.3 NtruEncrypt

The basis for our ultra-low power NtruEncrypt architecture is the multiplication operation in the ring R, a cyclic convolution of two polynomials of the same degree N. Considering a scenario, in which a sensor node encrypts a message and sends it to the base station, allows us to make the following observations which are helpful in creating an ultra-low power architecture. Similar observations can also be made for the case of decryption, but these are omitted here due to space restrictions.

- As mentioned at the beginning of this section, the public key of the node $h(x)$ is constant and embedded in the device. Since p is also constant, we can store a pre-scaled version of the public key $h'(x) = ph(x) \bmod q$. Thus we only need to compute $c(x) = \phi(x) \circledast h'(x) + m(x) \bmod q$.
- Coefficients of the public key $h(x)$ are computed modulo q and therefore occupy the larger of two wordsizes, while those of the random polynomial $\phi(x)$ are reduced modulo p. For our choice of $p = 3$ each coefficients of $\phi(x)$ is encoded as two bits. Since the public key is constant and realized as a look-up table, only $2N$ bits of storage are required as opposed to $N\lceil \log_2 q \rceil$.
- We assume that we have a good source of random bits available for generation of the random polynomial $\phi(x)$. In this paper we focus on the computational aspects of NtruEncrypt only, and therefore random number generation falls outside of the scope of this paper. For information on a compact implementation of an RNG based on digital artefacts requiring only a few hundred gates we refer to [21].

The algorithm consists of two nested loops: The outer loop iterates over all N coefficients of the result. The inner loop computes the coefficient by accumulating products of the form $a_i b_j$, with index i increasing and j decreasing mod$N$. The three major building blocks comprising the data path of the circuit—public key LUT, arithmetic units and circular buffer—are illustrated in Figure 3. The public key look-up table is realized in combinational logic that lends itself to optimization through the synthesis tool. The circular buffer consists of $2N$ bits of storage elements containing the coefficients of the random polynomial $\phi(x)$. Data enters the buffer through a multiplexer which connects the two ends of the buffer and forms a ring. Both, public key LUT and circular buffer, feed into the arithmetic units (AUs) which multiply and accumulate the operands. The smallest version of the circuit implements only a single AU. Yet, the architecture allows the implementor to scale up the number k of parallel AUs relatively easily, with minimal impact on the other elements of the design. Section 5.4 elaborates further on NtruEncrypt's inherent scalability. An AU consists of a partial product generator, a carry-save adder and a register. For any long operand a and short operand b the partial product generator will compute $ab \bmod q$. By choosing $p = 3$ and $q = 128$ the modular reduction of the intermediate result $c = \sum a_i b_j \bmod q$ comes essentially for free through simple truncation of bits at positions $\geq \log_2 q = 7$.

The control logic is designed to be as simple as possible in order to avoid being the bottleneck in terms of power consumption. The two nested counters needed

for keeping track of coefficients in the inner and outer loop of the algorithm are implemented as Linear Feedback Shift Registers (LFSR) for reduced switching activity. Furthermore, clock gating is used extensively whenever possible, to avoid any unnecessary switching activity and reduce parasitic wire capacitance.

In the case of only a single AU each round of computation takes $N + 8$ clock cycles to complete, with one coefficient per round. The eight additional clock cycles are necessary for addition of the message coefficient and propagation of carries in the carry-save adder. The total number of clock cycles for a full polynomial multiplication of N coefficients therefore takes $29,225$ clock cycles ($N = 167$). If k AUs are computing coefficients in parallel, the rounds overlap partially and the number of clock cycles amounts to $(N+8)(\lceil N/k \rceil)+k-1$. For a high degree of parallelization k the number of clock cycles can thus be reduced dramatically, i.e. to only 433 cycles for $k = 84$.

5 Analysis

In this section we analyze the proposed architectures according to various metrics of interest to ultra low-power applications such as sensor nodes. Since both architectures and algorithms are distinctly different from each other a direct comparison is difficult. We alleviate this situation by fixing system parameters to values that match security levels of both systems as closely as possible, as mentioned in Section 2.1.

5.1 Definition of Metrics

Chip Area. The number of equivalent gates (2-input NAND gate) used by the circuit. This metric is independent of the process technology, and correlates well with the actual area of the physical layout.

Power Consumption. This is the total power consumption of the circuit, categorized into static and dynamic power. This metric is highly dependent on the process technology. In this context, however, both architectures use the same target library so that differences in power consumption are a direct consequence of differences in the architecture.

Throughput. Specifies the number of plaintext bits that are encrypted per second. This metric is independent of any message expansion properties of a given system.

Energy per Bit Encrypted. Amount of energy necessary to encrypt a single bit of the message. This metric can be used to compare the energy efficiency of cryptosystems with a roughly equivalent level of security. It is independent of the actual operand length.

Scalability. Refers to the possibility of scaling an algorithm between bit serial and highly parallelized realizations in an efficient manner. A closely related concept is modularity, which is an indicator of how easily simple processing elements

can be replicated for a higher degree of parallelization of a task in performance critical settings.

5.2 Rabin's Scheme

The main concern driving our low-power implementation of Rabin's Scheme is its storage requirement. Many well known techniques for optimizing a modular squarer require either more circuitry or more storage elements. At our targeted clock frequency of 500 kHz the static power consumption is dominant and therefore has to be minimized. Hence, we built a squarer as a bit-serial multiplier, operating on the entire width of the 512 bit multiplicand and on a single bit of the multiplier at a time. In order to conserve area we use the same adder for accumulating the partial products, modulo reducing the results, and re-aligning the carry bits before each shift. This approach consumes a chip area of less than $17,000$ gates with its accompanying static power consumption of $117.5\mu W$. The dynamic power consumption at 500 kHz is $30.68\mu W$ resulting in a total average power consumption of $148.18\mu W$ (Table 1). It increases linearly with the operating frequency as shown in Figure 1. A breakdown of the power consumption by functional blocks reveals that the adder consumes 40% of the power and all storage elements combined consume 38%. The power consumption of the complex control logic for this circuit is negligible at 2%.

5.3 NtruEncrypt

The hardware friendly arithmetic underlying the NTRU system lends itself very well to highly scalable and low-power implementations, since the computation of each individual coefficient is independent from one another. At the same time we can reorder the computation in a way that facilitates parallel computation of multiple coefficients. This can be achieved by simply replicating arithmetic units and slightly adjusting the control logic. The circular buffer allows parallel access to multiple coefficients in sequential order as illustrated in Figure 3, Section 4, thereby avoiding memory access bottlenecks.

A breakdown of the power consumption by functional blocks reveals that the most significant contribution is made by the circular buffer (77%), while an arithmetic unit contributes the least amount (6%). The cost for an implementation with only a single arithmetic unit is therefore relatively high compared to a more parallelized variant. The small cost of adding arithmetic units, on the other hand, allows for a high degree of parallelization. This level of scalability is advantageous when it comes to achieving optimal energy efficiency. In the following analysis we therefore also consider performance estimates for a highly parallelized ($k = 84$) variant of our NtruEncrypt architecture, based on data obtained from simulation of the digit serial implementation ($k = 1$).

Our smallest implementation of NtruEncrypt with a single arithmetic unit takes up a chip area of less than 3000 equivalent gates, including the circular buffer and the combinational look-up table of the public key. Gate level power simulation indicates an average power consumption of less than $20\mu W$ at a clock

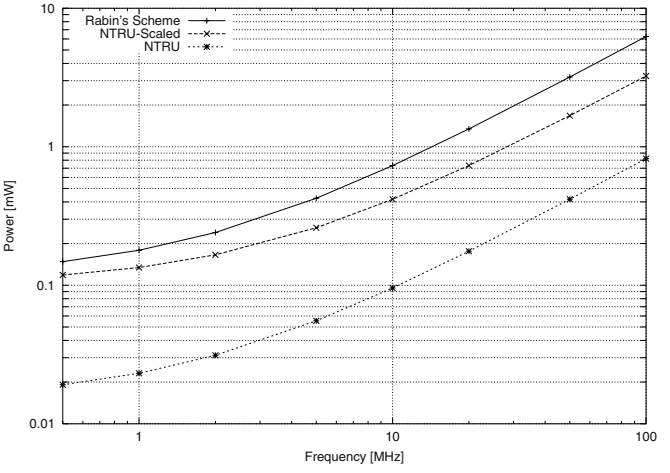


Fig. 1. Power Consumption over a Range of Clock Frequencies

frequency of $500kHz$, close to the amount of static leakage power (see Table 1). As before with Rabin's Scheme we see dynamic power consumption beginning to dominate at faster clock speeds as it increases linearly with the frequency (Fig. 1). This is in agreement with our expectation from (1).

5.4 Comparison

Table 1 shows a direct comparison between Rabin's Scheme and both variants of NtruEncrypt in the metrics defined above. The architectures of Rabin's Scheme and the simple variant of NtruEncrypt were both intended to achieve the least possible power consumption given the available standard cell library, without necessarily reaching optimal energy efficiency. The results are summarized in the first two columns. After an initial analysis of the architectural differences we decided to include estimates for a highly parallelized variant of NtruEncrypt in the third column of the comparison. The degree of parallelization $k = 84$ was chosen in a way that the area footprint roughly matches that of Rabin's Scheme, and secondly that $\lceil N/k \rceil - N/k$ is as small as possible. This is to divide the number of coefficients N in a way that utilization of the AUs is high during the last round of computation.

Rabin's Scheme takes up almost six times the area of simple NtruEncrypt with a single AU. On the other hand it also has the advantage of performing almost forty times better. This is to be expected due to its large operands and full-word arithmetic. If, however, the absolute area and power requirements are the limiting factor, it might not be flexible enough. Also, our estimates for the parallelized variant of NtruEncrypt indicate that it outperforms Rabin's Scheme by nearly factor two using the same area footprint. From the figures in Table 1 it is evident that static leakage power is the main culprit for the relatively high energy consumption of both implementations. We would like to stress the fact

Table 1. Summary of comparison between Rabin's Scheme and NtruEncrypt

	Rabin	Ntru ($k = 1$)	Ntru ($k = 84$)
Equivalent security	60 *bits*	57 *bits*	57 *bits*
Area [eqv. gates]	16,726	2,850	16,200
- combinational	8,875	523	7,000
- storage elements	7,851	2,327	9,200
Delay (avg. # cycles)	1,440	29,225	433
Avg. power @ 500kHz	148.18 μW	19.13 μW	118.7 μW
- static (%)	117.5 μW (79.3%)	15.10 μW (78.9%)	103.06 μW (86.8%)
- dynamic (%)	30.68 μW (20.7%)	4.03 μW (21.1%)	15.64 μW (13.2%)
- peak power	169.8 μW	20.22 μW	n/a
Energy	426.76 nJ	1,118.15 nJ	102.79 nJ
- per bit encrypted	833.5 pJ (512 bits)	4,235.41 pJ (264 bits)	389.4 pJ (264 bits)
Throughput	177.8 $kbits/s$	4.52 $kbits/s$	304.85 $kbits/s$

that leakage power is highly technology dependent and that the ASIC standard cell library we use is not optimized for low power design. The dynamic power consumption of an architecture, on the other hand, is proportional to its switching activity. It therefore makes sense to differentiate between these two influences if we want to compare the relative merits of one architecture over the other, independently of the process technology. It turns out that dynamic power consumption in Rabin's Scheme is nearly twice as high as in NtruEncrypt's case, despite the same area and the fact that leakage power differs by only 12%.

The throughput that either architecture can achieve at a given clock frequency depends on the number of clock cycles for an encryption and the number of plaintext bits per block. In Rabin's Scheme the plaintext is up to 512 bits long. At a clock frequency of 500 kHz and an average of 1440 cycles per operation this translates into a maximum theoretical throughput of 177.8 kbits/s. Since NtruEncrypt uses N ternary coefficients we can determine its throughput in terms of kbits/s by first converting the capacity of the message polynomial $m(x)$ into bits. $N = 167$ ternary coefficients can hold information equivalent to $\lfloor N \log_2 3 \rfloor = 264$ bits. The entire encryption operation takes 29225 clock cycles for NtruEncrypt with a single AU, and 443 cycles with 84 AUs. Operating at the same clock frequency, the simple variant compares unfavorably to Rabin's Scheme at only 4.52 kbits/s throughput, almost 40 times less. The estimates for the highly parallelized variant, however, indicate a performance level of 304.85 kbits/s, nearly twice the throughput of Rabin's Scheme.

For any cryptographic scheme there is a multitude of possible design choices by which power consumption can be traded off against performance and vice versa. Ultimately, however, we would like to know the amount of energy that is necessary for an elementary encryption operation, i.e. the cost of encrypting a bit of data at a certain level of security. The amount of energy for the entire operation is the product of average power consumption and the time it takes to

complete that operation. Considering the amount of plaintext data that can be encrypted in one operation, we determine the amount of energy per bit encrypted as

$$E_{\text{bit}} = \frac{P_{\text{avg}} \cdot n_{\text{cycles}}}{f_{\text{clock}} \cdot l_{\text{op}}}$$

where l_{op} is the operand length in bits, i.e. 512 for Rabin's Scheme and 264 for NtruEncrypt. As we have discussed earlier, Rabin's Scheme uses more power than NtruEncrypt with a single AU, but it also takes much fewer clock cycles to complete. We can make a similar observation by looking at the energy per bit metric. The amount of energy necessary to encrypt a single bit with NtruEncrypt is about five times higher than with Rabin's Scheme. The picture changes yet again when we consider NtruEncrypt's parallelized variant. Our estimates suggest that the amount of energy per bit drops by nearly factor 11 and is thus less than half the amount of Rabin's Scheme.

To put our results into perspective, we compare them to estimates reported in [8] that were obtained from simulation of various public key algorithms on existing general purpose processor architectures. An implementation of the emerging scheme XTR on the ARC3 processor suggests an energy consumption of around 130 μJ at a security level that is comparable to RSA-1024 or 72 bits of equivalent security. Despite the difference in security levels, this is still between factor 100 and 1000 more energy than what our architectures require, proving the strength of customized application specific architectures.

6 Conclusions

We have demonstrated in this paper that it is possible to design public key encryption architectures with power consumption of less than 20 μW using the right selection of algorithms and associated parameters, optimization and low-power techniques. In spite of the common perception of public key cryptography, it is possible to achieve a level of power consumption low enough to allow its use even in self-powered sensor nodes. Our implementation is based on a regular ASIC standard cell library that is not specifically optimized for low-power. It is thus possible to achieve even better results than ours, although that is not the point we are trying to make here. The use of public key schemes facilitates much simpler security protocols than those currently in use with the sensor network community, and has a potential impact on a much wider range of applications. RFIDs and contactless smart cards are further examples of ubiquitous computing applications requiring energy efficient cryptographic functions. So far public key cryptography has not even been considered for these devices due to its perceived complexity.

Our findings show further that schemes based on traditional modular arithmetic, such as Rabin's, does have a significant disadvantage compared to new and emerging schemes represented here by NtruEncrypt. The use of arithmetic in a polynomial ring allows for a very compact, yet scalable implementation in

hardware. Additionally, NtruEncrypt's decryption operation—although not further considered in this paper—is based on the same arithmetic operation. This opens up the possibility for realization of two way key exchange protocols, while this is more difficult with Rabin's Scheme, due to its asymmetric properties of encryption and decryption.

Further research into energy efficient cryptographic primitives is necessary, but our findings give us the confidence that public key cryptography in ubiquitous computing applications is possible and that it can be done efficiently using customized hardware architectures.

References

1. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.E.: SPINS: security protocols for sensor networks. Wireless Networks **8** (2002) 521–534
2. Fulford, B.: Sensors gone wild. Forbes Global (2002) http://www.forbes.com/global/2002/1028/076_print.html.
3. Polastre, J.: Design and implementation of wireless sensor networks for habitat monitoring. Master's thesis, University of California at Berkeley (2003)
4. Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., Anderson, J.: Wireless sensor networks for habitat monitoring. In: First ACM Workshop on Wireless Sensor Networks and Applications, Atlanta, GA, USA. (2002)
5. Burne, R., et al.: Self-organizing cooperative sensor network for remote surveillance: improved target tracking results. In: Proceedings of the SPIE. Volume 4232., Boston, SPIE, SPIE-Int. Soc. Opt. Eng, USA (2001) 313–321
6. Meininger, S., Mur-Miranda, J., Amirtharajah, R., Chandrakasan, A., Lang, J.: Vibration-to-electric energy conversion. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **9** (2001) 64–76
7. Amirtharajah, R., Chandrakasan, A.P.: Self-powered signal processing using vibration-based power generation. IEEE Journal of Solid-State Circuits **33** (1998) 687–695
8. Carman, D.W., Kruus, P.S., Matt, B.J.: Constraints and approaches for distributed sensor network security. Technical report, NAI Labs, Security Research Division, Glenwood, MD (2000)
9. Weimerskirch, A., Paar, C., Shantz, S.C.: Elliptic curve cryptography on a Palm OS device. In V. Varadharajan, Y.M., ed.: The 6th Australasian Conference on Information Security and Privacy (ACISP 2001). Volume 2119 of LNCS., Heidelberg, Springer-Verlag (2001) 502–513
10. Rabin, M.O.: Digitalized signatures and public key functions as intractable as factorization. Mit/lcs/tr-212, Massachusetts Institute of Technology (1979)
11. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21** (1978) 120–126
12. Hoffstein, J., Pipher, J., Silverman, J.: NTRU: A ring-based public key cryptosystem. In Buhler, J., ed.: Algorithmic Number Theory (ANTS III). Volume 1423 of LNCS., Berlin, Springer-Verlag (1998) 267–288
13. Lenstra, A.K., Verheul, E.R.: Selecting cryptographic key sizes. Journal of Cryptology: The Journal of the International Association for Cryptologic Research **14** (2001) 255–293

14. Hoffstein, J., Silverman, J., Whyte, W.: NTRU report 012, version 2. estimated breaking times for NTRU lattices. Technical Report 12, NTRU Cryptosystems, Inc., Burlington, MA, USA (2003)
15. Menezes, A.J., van Oorschot, P.C., Vanstone, S.: Handbook of Applied Cryptography. CRC Press Inc. (1997)
16. Hoffstein, J., Silverman, J.H.: Optimizations for NTRU. In: Proceedings of Public Key Cryptography and Computational Number Theory, de Gruyter, Warsaw (2000)
17. Bailey, D., Coffin, D., Elbirt, A., Silverman, J., A.Woodbury: NTRU in constrained devices. In Ç. Koç, Naccache, D., Paar, C., eds.: CHES 2001. Volume 2162 of LNCS., Berlin, Springer-Verlag (2001) 266–277
18. Devadas, S., Malik, S.: A survey of optimization techniques targeting low power VLSI circuits. In: Proceedings of the 32nd ACM/IEEE Conference on Design Automation. (1995) 242–247
19. Rabaey, J., Pedram, M.: Low Power Design Methodologies. Kluwer Academic Publishers, Norwell, Massachusetts (1996)
20. Parhami, B.: Computer Arithmetic: Algorithms and Hardware Designs. Oxford University Press (2000)
21. Epstein, M., Hars, L., Krasinski, R., Rosner, M., Zheng, H.: Design and implementation of a true random number generator based on digital circuit artifacts. In Walter, C.D., Çetin K. Koç, Paar, C., eds.: CHES 2003. Volume 2779 of LNCS., Berlin, Springer-Verlag (2003) 152–165

Appendix

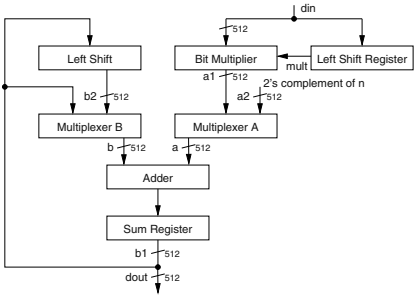


Fig. 2. Block Diagram for Rabin's Scheme

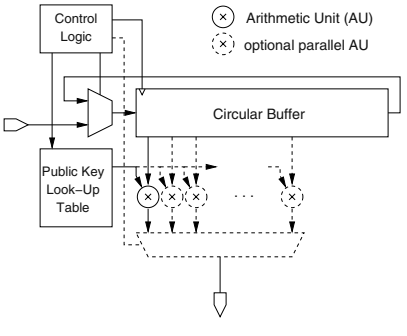


Fig. 3. Block Diagram for NtruEncrypt

Exploring Message Authentication in Sensor Networks

Harald Vogt

Department of Computer Science,
ETH Zürich
vogt@inf.ethz.ch

Abstract. This paper explores the design space for message authentication in sensor networks. Several types of authentication are put into relation: end-to-end, hop-to-hop, and physical and virtual multipath authentication. While end-to-end authentication provides the highest and most general security level, it may be too costly or impractical to implement. On the other end of the security scale, hop-to-hop authentication can be implemented with little effort but provides security only to a highly restricted attacker. Multipath authentication provides an intermediate security level that may be appropriate for many applications of sensor networks, trading energy for security guarantees. Virtual multipaths offer an improvement, reducing energy demands while retaining crucial security properties of physical multipaths.

1 Introduction

Sensor networks are a novel paradigm for large-scale distributed systems. Individual sensor nodes are resource-restricted, typically battery-powered devices equipped with a radio interface for communication. A typical communication pattern is the aggregation of sensoric data and transmitting results to the edge of the network. Sensor networks are often operated in conjunction with fixed or mobile base stations that collect data, serve as network bridges and provide computational resources.

The potentially unlimited size of a sensor network with thousands of nodes and the need to manage with limited resources and conserve energy as much as possible, on each single node as well as throughout the network, makes secure communication challenging. Various factors are important: Nodes depend on each other for correct operation. Messages have to be transmitted over several hops, since direct communication between arbitrary nodes is impossible due to limited radio range. Nodes have little knowledge of other, distant nodes.

Wireless communication links and multi-hop message transmission are extremely vulnerable to eavesdropping and manipulation. A node that wants to collect sensor data from distant peers must at least be able to check the integrity of the received data. In a strict sense, this is only possible if data is authenticated. This, however, requires that the sender's identity is known and depends

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 19–30, 2005.

on the existence of a common security framework in which the sender and the receiver are embedded. Such a framework is usually established by an institution that is trusted by both communication endpoints, e.g. a base station (online) or a certification authority (offline).

Many applications for sensor networks need only restricted communication modes, such as between nodes and the base station. In that case, security can be supported using the resources available at the base station. Sensor nodes are required only to have a trust relationship with the base station, which imposes moderate requirements on memory and cpu power of single nodes.

Several questions arise when base stations are assumed. Is there only one base station? Must all traffic be routed through it? Can I add my own base station, using the sensor network as a service? What happens if a base station is compromised? How about combining sensor networks that depend on different base stations? It seems that the dependence on a base station constrains the applicability of a sensor network. Additionally, efficiency can often be increased if communication does not involve a base station. Therefore, we would like to be able to build sensor networks in which node-to-node communication is possible in a secure way.

The purpose of this paper is to assess the options for protecting the integrity of messages in sensor networks without the need to rely on base stations. The goal is not to have a solution that protects against all kinds of attacks, but to achieve a certain level of security at reasonable cost.

One of the distinctive characteristics of sensor networks is the fact that the identity of individual nodes should not be important to the correct operation of the system. First, limited storage capacity makes it impossible for a node to keep specific information on even a moderate (compared to the overall network size) number of other nodes. Second, other attributes such as the geographical position or the quality of sensor measurements are more important to achieving the objective of a sensor network installation than the existence of a single specific node. If one node fails, another nearby node would take over its responsibilities. Thus, the identity of the data collecting node changes, but this fact should be transparent to clients or distant nodes that are interested mainly in data quality.

However, identity is important to be able to verify that a message has been sent by a legitimate entity. A simple trust framework for message transmission is hop-to-hop authentication of data, where there is one key per communication link. Here, communication endpoints have no knowledge about each other, but intermediate nodes are trusted not to manipulate the message. This trust is justified if nodes are correctly implemented and not subject to manipulation. The problem with this approach is the fact that even a small number of compromised nodes can severly affect the security of the network. The situation is similar when a globally shared key is used to authenticate messages. In the case of a globally shared key, all communication is compromised with even a single compromised node. With simple hop-to-hop authentication, a malicious node controls the traffic on all communication paths it is part of. Therefore, such a framework offers protection only against limited, external attacks.

The other extreme is a pairwise end-to-end relationship between all communicating entities. Such relationships are, however, costly to establish and to maintain. As we will discuss in the following section, communication relationships are often ad hoc and short-lived, so the effort of establishing an end-to-end relationship may often not be justified.

An extension to simple hop-to-hop authentication is the use of multiple paths over which messages are transferred. Node-disjoint paths mitigate the impact of small numbers of compromised nodes that try to disrupt communication or manipulate messages. On the other hand, multipath message transmission has severe drawbacks, such as increased energy consumption. An alternative is the variation where a different path is chosen for each transmission, thus increasing the probability that a compromised node is circumvented, and the energy consumption is balanced among nodes.

These approaches are, however, limited by the connectivity of the network, which determines the number of node-disjoint paths between any two nodes. If an attacker chooses the attacked nodes carefully, the effectiveness of an attack can be drastically increased.

To counter the problems of multipath routing, we propose the use of multiple authentication paths over a single communication path [14], thus increasing the reliability of message transmissions, especially the protection of message contents against manipulation. This allows for secure communication between arbitrary nodes in the network with high probability without the requirements imposed by end-to-end techniques.

In Sect. 2, we discuss several communication patterns prevalent in sensor networks and touch upon their security requirements. Sect. 3 describes the security guarantees delivered by different authentication schemes. We argue in favor of virtual multipath authentication in the context of sensor networks. Sect. 4 briefly shows how keys can be established between neighbouring nodes, which is necessary for the proposed approach. The final two sections discuss related work and conclude.

2 Communication Patterns

There are some general communication patterns in sensor networks that are applicable to a wide range of applications. In this section, we argue that it is generally beneficial that sensor nodes exchange data with each other before a base station is involved. Thus arises the need for protection of this communication, which is discussed in the next section.

2.1 Content Based Routing

An important mode of operation in a sensor network is the routing of a message according to its contents, which is often inherently multicast [3, 5], instead of a designated receiver. Entities interested in certain types of events announce their interest, or events are distributed based on geographical information, for

example. The advantage is that event sources don't have to store a mapping from event types to identities of interested entities but need only keep a small amount of routing information, if any.

During such kind of communication, sender and receiver remain unknown to each other. The source does not know in advance, and will never learn, which are the receivers of its messages. This poses several security problems, such as the enforcement of access control policies and message authentication. From the receiver's point of view, it might not be important from which node exactly a message originates. The receiver's interest lies mostly in the integrity of messages. If end-to-end mechanisms are available, identity can be used to check the authenticity of the data, which also ensures integrity. In many cases it may be sufficient that message integrity is preserved with high probability, for example if there are many sources and messages are aggregated before taking crititcal actions.

A client outside of the network perceives the sensor network as a single entity, just like the user of a web service, who does not care about single machines. In contrast to the internet, where resources are plenty and certificates can be used to ensure service integrity, other means must be used for sensor networks. Also, web servers are kept in closed compartments, whereas sensors are deployed in open environments. Factors such as physical appearance are more likely to convince a client of the service quality.

2.2 Aggregation

Aggregation (and correlation) of sensor data is a prerequisite for detecting higher-order events that cannot be reliably inferred from data of single sensor nodes, for example the trajectory of a moving object [12]. A distinction can be made between *local* and *distant* aggregation. Many tasks involve the cooperation of neighbouring nodes, while distant nodes have to be involved for phenomena that affect large areas, such as earthquakes. While it might be possible to collect all data at a base station and perform the aggregation there, it is likely more efficient to aggregate data within the sensor network first.

2.3 Node-to-Node Versus Base-Oriented Communication

It is often assumed that security-relevant communication should involve a base station. This is reasonable, since it is much easier to guarantee end-to-end security properties in conjunction with a well-equipped base station. However, some tasks, such as aggregation, are more efficient when performed to an extent as large as possible within the network. Requiring a base station restricts the applicability of a sensor network. We hope that our examples show that node-to-node communication is a valuable concept for sensor networks. If this type of communication is about to take place in a sensor network, appropriate security mechanisms have to be adopted. In the following, we argue that other means beside end-to-end mechanisms are feasible and can give appropriate security guarantees.

3 Security Guarantees

What are appropriate security guarantees for a sensor network? In this section, we try to shed some light on the space of possibilities. End-to-end properties are usually considered as the highest level of security achievable, since all potential intermediaries are eliminated (apart from denial of service attacks). But end-to-end properties are nullified if an endpoint is compromised – and in sensor networks, every node is an endpoint. (This is in contrast to a virtual private network, where only a small subset of internet hosts are considered legitimate endpoints.)

Possible failure of nodes should be designed into algorithms for sensor networks as it is highly likely that a certain percentage of them will fail during normal operation. A sensor network should be able to tolerate a certain number of malicious nodes as well. The security of a sensor network is then a function of the ratio of compromised vs. correct nodes, and can be expressed as the probability of being able to compute correct results from sensor data. This security model is supported by other approaches we present in this section, besides end-to-end mechanisms.

3.1 End-to-End

The usual approach for securing communications in a network is to establish an end-to-end trust relationship between the sender and the receiver of a message. An important distinction in this regard is that between *entity authentication* and *message authentication*. In the first case, the mere identity of a communication peer is verified, while in the latter case, the origin of a message and the integrity of its content is assured. We are mainly interested in the latter.

End-to-end mechanisms are based on the existence of a trusted authority. This authority issues credentials and verification tools to all nodes. Credentials are used to assert the authenticity of data packets. Verification can be done either "online" or "offline", online meaning that the trusted authority is active and can be contacted. Thus, it is necessary for each node to establish a trust relationship with the authority server only. This approach is pursued with the μTESLA protocol [9], for example. Offline verification means that each node is self-sufficient and can verify other nodes' credentials on its own. This can be implemented with public key cryptography.

Most practical sensor networks rely on online base stations. These are not necessarily created for security purposes, but are required for other tasks, such as positioning [11] and data aggregation [13]. Since base stations can be equipped with much more resources than sensor nodes, adding security features on top of them is a viable approach.

We see several problems with the end-to-end approach to security, especially in the context of sensor networks:

- **Extensibility.** As the size of a sensor network grows, paths between a base station and individual nodes become longer. This induces a growing delay and increased traffic at more nodes and increased load on the base station.

- **Interoperability.** Sensor networks from different operators cannot be combined easily, since all traffic must be routed through the base stations for authentication.
- **Base station as single point of attack.** An attacker will always try to choose the easiest way attacking a system. If a large part of the traffic is routed through one base station, this base station becomes an attractive target and must be protected appropriately. This might be challenging in certain settings (e.g. when the base station needs to be located within a hostile area), while in others, this might be advantageous (e.g. when the base station is under constant surveillance). Protecting the base station only, however, might result in a false sense of security. If an attack on the base station is virtually impossible, an attacker will concentrate on the sensor nodes themselves, which cannot go without protection (increasing overall cost).
- **Unbalanced energy consumption.** The need to route a large portion of data traffic through a base station implies that sensor nodes near this base station spend their energy faster than nodes that are farther away. Thus, sensors need to be more densely deployed around base stations, or designed asymmetrically. Both solutions increase cost.
- **Computing power requirements.** While symmetric key operations are feasible on small sensor nodes, asymmetric key cryptography, required by offline verification, is not feasible for many types of sensor nodes.

Employing end-to-end security techniques is the most secure mode of operating a sensor network. For a successful attack, an attacker must gain control either over all involved sensor nodes (or a majority thereof), or over the base station. Therefore, both the base station and the sensor nodes have to be protected against manipulation.

3.2 Hop-to-Hop Authentication

If individual sensor nodes are well protected against tampering, it is reasonable to rely on them to reliably and correctly forward messages on behalf of other nodes. Attacks on communication links can be thwarted by link encryption and authentication, or the use of a (regularly updated) globally shared key [1]. Adversaries that are not capable of planting their own nodes within the network or take control of existing nodes, cannot manipulate messages.

But if an adversary manages to compromise even a single legitimate node, the danger arises that all communication within the network becomes subject to eavesdropping and manipulation, for example through a successful sinkhole attack [8]. Thus, the integrity of every single node is important for the security of the overall network. In this sense, every node depends on each other. This is an extremely strict requirement, which is unlikely to be met in practical deployments. Hop-to-hop authentication is therefore only suitable under a severly restricted adversary model.

3.3 Multiple Path Authentication

Multiple paths mitigate the problem of dependence. Instead of relying on a single node to forward messages correctly, a node cooperates with a number of nodes in its neighbourhood. A message can be sent on alternating paths, or on multiple paths in parallel. Both reduces the impact of isolated failures. Multipath routing has been used mostly for fault tolerance and load balancing, and recently for failure recovery [7] in sensor networks. Disjointness is often not strictly required for such applications.

Multipath routing in conjunction with hop-to-hop authentication results in multipath authentication. A message is sent over multiple, strictly disjoint, paths. If different versions of a message are received, the recipient chooses the majority version. All other paths can be markes as untrustworthy, since they delivered a presumably incorrect message. This is similar to the approach described in [10], where PGP keys are authenticated over multiple disjoint paths (these are not communication paths, but paths in a certification graph).

There are several problems with multipath authentication, when physically disjoint paths are used:

- More nodes need to spend energy on routing.
- Multiple disjoint paths require a minimum degree of connectivity. Some nodes may be only loosely connected to the network and cannot profit from multipath routing.
- It is more demanding to find and maintain sets of disjoint paths between two communication endpoints, compared to a single path. In the worst case, multipath routing boils down to (partial) flooding of the network, for example when a message is to be delivered to several nodes in different parts of the network.

Apart from these problems, multipath authentication reflects an important concept in sensor networks: cooperation of neighbouring nodes. By authenticating messages to each other, they eliminate the impact of maliciously behaving nodes.

3.4 Virtual Multipath Authentication

Instead of transferring multiple physical copies of the same message, we can restrict ourselves to one physical communication path with superimposed (virtual) authentication paths. These are established among nodes on the communication path that share pairwise secret keys. The distribution of these shared keys can be random,or can be based on a regular pattern. Here, we are focusing on a regular distribution.

A regular distribution guarantees that there exists a number of node disjoint authentication paths between any two communicating nodes. The following simple scheme defines such a regular key distribution. It provides two authentication paths per communication path. We call it the *Canvas* scheme [14].

Initially, each node shares a secret key with each of its neighbours. Neighbours include all nodes that are reachable either through a direct communication link,

or through at most one intermediate node (which has to be a direct neighbour). Sect. 4 describes how to establish such a key setup. This guarantees that on any communication path between two nodes, there exist two disjoint authentication paths. Note that also direct communication links are authenticated. Fig. 1 shows an example.

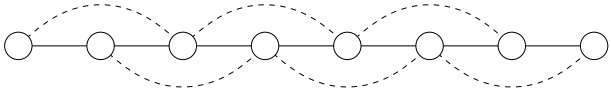


Fig. 1. A *Canvas* communication path

Message forwarding works as follows. A message travelling a path $S_0, S_1, S_2, \ldots, S_n$ is authenticated twice before it is forwarded. S_0 creates MACs intended for nodes S_1 and S_2. S_0 can only reach S_1 directly and relies on S_1 to transmit the MAC intended for S_2. Before S_1 forwards the message, it creates two new authentication codes itself for S_2 and S_3. This is continued until the message reaches its final destination.

Before a node forwards a message, it checks the authentication codes from the two preceding nodes. If both codes indicate that the message has not been manipulated, the node forwards the message. An exception arises when a message is created, where only one MAC needs to be checked by the immediate neighbour of the source node.

It is obvious that two adjacent nodes can cooperatively compromise the communication path. They are able to manipulate and inject arbitrary messages that are routed through them. This seems to be only a slight improvement over simple hop-to-hop authentication at first. Instead of compromising one node, an attacker now has to gain control over two of them. And since they are co-located, an attack should be easy. Thus it seems nothing much is gained.

In order to show that the Canvas scheme provides a significant improvement, we first have to make clear what types of attacks we can expect to counter with the security schemes proposed in this paper, and which we cannot.

If there is an attack possible that exploits a fault present in all sensor nodes, and the attack can be automated (like a typical attack on hosts in the internet), the effort to compromise only one node is essentially as high as to compromise a large number of nodes or even all of them. Such attacks cannot be countered by any of the security schemes described in this paper. They must instead be be tackled by careful system design, intrusion detection techniques, quick response etc.

We propose a metric, called "live paths", to assess the security of a sensor network. A path is called *live* if and only if its both endpoints are not compromised. The rationale is that if an endpoint is compromised, it does not contribute meaningfully to the overall result of a computation. This is true even if end-to-end security measures are avaialable. The set of of live paths is thus an indicator of the quality of the network.

Additionally, we call a path "functional" if it is both live and the endpoints can communicate securely. With simple hop-to-hop authentication, a path becomes non-functional if at least one node on the path is compromised. With the Canvas scheme, a path remains functional unless two adjacent nodes are compromised. Note that under end-to-end security, a live path is always functional.

A simulation on a sensor network with 280 randomly placed nodes on a plane of 500 on 500 square meters and a communication range of 50 meters shows the impact of an attack under the different security schemes (Fig. 2). The simulated attacker acts "smart" with regard to the Canvas scheme. Instead of attacking isolated nodes, pairs of nodes are being attacked. A path is always a shortest path between a pair of nodes.

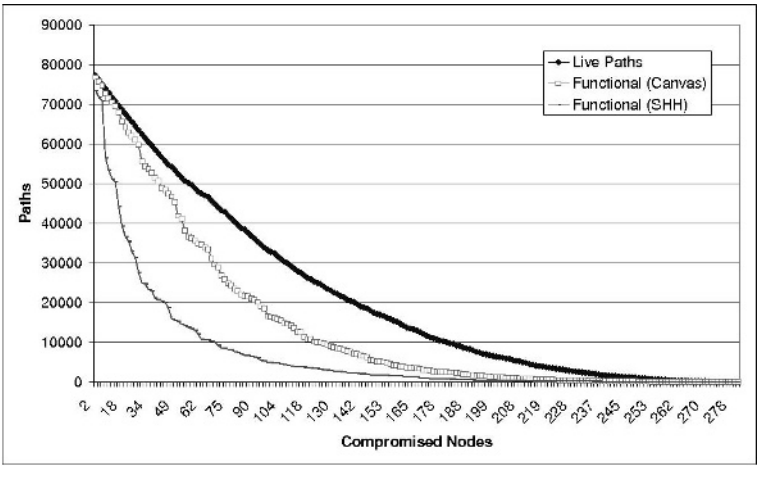


Fig. 2. Degradation of functional paths in a network under attack with regard to different security schemes

Obviously, the quality of the network degrades with the number of nodes being compromised, meaning that secure communication becomes less likely. The number of functional paths drops sharply when the simple hop-to-hop scheme (SHH) is employed. The Canvas scheme is more resilient and guarantees secure communication for a significantly larger fraction of the nodes. The optimum achievable – through end-to-end techniques – is represented by the curve denoting the live paths.

It may be subject to debate to what degree live and functional paths are a meaningful metric for the security of a sensor network. On an abstract level, it seems to be sensible. Consider for example a user who wishes to query the sensor network and connects to an arbitrary (non-compromised) node. The more functional paths there are to other nodes, the higher the probability will be that the resulting data has a certain quality.

To summarize, we bring forward two arguments that show that the Canvas scheme can be a significant improvemet.

1. The effort that must be invested for an attacker to compromise a communication path is doubled. This can be enough to detain a potential attacker, if individual nodes are sufficiently protected. Similarly, we can say that if the probability for a successful attack on a single node is small enough, it is highly unlikely that an attacker succeeds in compromising two adjacent nodes.
2. Even if the attacker manages to compromise a certain number of nodes, the impact is rather small at first. With a growing number of compromised nodes, the Canvas scheme performs significantly better than hop-to-hop authentication.

The Canvas scheme makes it necessary that with each message, three MACs are transmitted. It is therefore affordable only for messages of a certain size. Also, several symmetric key operations (verification and MAC creation) are necessary at each hop. This could be a several drawback if such computations are energy intensive or slow on a certain node type. However, there is room for improvement, for example by using specialized, more efficient cryptographic hardware support.

Finally, we would like to point out that the Canvas scheme can be generalized such that more than two authentication paths are available per communication path. This is achieved by increasing the "reach" of a node to its k-hop neighbours. The distance between a pair of nodes sharing a key would increase. In the extreme case, we would arrive at the end-to-end situation, where each node shares a key with each other (if k equals the diameter of the network).

4 Key Setup

Several proposals have been made for setting up keys in sensor networks. One of the most intriguing has been made by Eschenauer and Gligor [6]. Each node is assigned an identity that uniquely determines a limited set of key values being drawn from a common set. The identity can be made public without helping a potential attacker. Based on their identities, two nodes can determine their common subset of key values. This subset is unique to this pair of nodes with high probability and can be used to exchange a secret key without the use of computationally intensive public key cryptography.

The key setup for the Canvas scheme is straightforward. A node establishes a unique secret key with each of its neighbours, including the ones that are not directly reachable, based on their identities. A man-in-the-middle attack is impossible since identities are unique. A node imitating several identities would have to produce key values he does not know.

Note that with such an identity scheme, also end-to-end security properties can be established, if each endpoint knows the identity of the other one. However, as we have shown above, a distinctive end-to-end relationship is rather the exception. Also, after the initial phase, the memory storing the key values could be reused for applications. Then, no more keys can be negotiated.

5 Related Work

Virtual authentication paths have been studied by Beimel and Franklin [2], who differentiate between the communication graph of a network, and the authentication graph defined by the keys shared among nodes. They give a characterization of reliable message transmission, which depends on the connectivity of the communication graph and the union of both graphs.

Multipath routing for sensor networks is examined in [7] with a focus on failure recovery and minimization of energy consumption. Strictly disjoint paths are not required, but it is desirable to have backup paths in case of failure.

Zhu et al. [16] discuss several important issues in sensor network security. It is assumed, for example, that sensor nodes can withstand a physical attack at least for a small amount of time, which gives the sensors time for neighbour detection and key establishment. In this work, protocols are given for establishing keys on several levels, including group, cluster and pairwise keys, which are based on the identity of nodes.

The idea of establishing pairwise keys based on pre-distributed random values is explored in several papers [4, 6, 15]. Chan et al. [4] uses multiple paths for key reinforcement, which further helps increase the resilience against link key compromise.

In a paper by Zhu et al. [17], interleaved message authentication, similar to our *Canvas* approach, is used for filtering false messages on their way from a sensor node to the base station. This principle is applied in addition to strong source authentication using a key that is shared between the base station and the sensor node.

6 Conclusion

Sensor networks constituted from a huge number of small, resource-restricted devices, need an understanding of security that takes their applications, their architectures and the capabilities of single nodes into consideration. Sensor networks operate under much tighter constraints than networks of personal devices (where humans are present most of the time) or conventional distributed systems like local area networks. In this paper, we have presented part of the design space for communication security in sensor networks, focusing on message integrity. The approach of virtual multipath authentication prefers localized communication over long-distance protocols and is thus scalable to very large network sizes.

References

1. Stefano Basagni, Kris Herrin, Danilo Bruschi, and Emilia Rosti. Secure Pebblenets. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 156–163. ACM Press, 2001.
2. A. Beimel and M. Franklin. Reliable Communication over Partially Authenticated Networks. *Theoretical Computer Science*, 220(1):185–210, 1999.

3. David Braginsky and Deborah Estrin. Rumor Routing Algorthim for Sensor Networks. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 22–31. ACM Press, 2002.
4. Haowen Chan, Adrian Perrig, and Dawn Song. Random Key Predistribution Schemes for Sensor Networks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 197–213. IEEE, May 2003.
5. Frank Dürr and Kurt Rothermel. On a Location Model for Fine-Grained Geocast. In *Ubicomp 2003: Ubiquitous Computing*, volume 2864 of *LNCS*, pages 18–35. Springer-Verlag, 2003.
6. L. Eschenauer and V. D. Gligor. A Key-Management Scheme for Distributed Sensor Networks. In *CCS'02*. ACM, 2002.
7. D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks. *Mobile Computing and Communications Review*, 1(2), 1997.
8. Chris Karlof and David Wagner. Secure Routing in Wireless Sensor Networks. *Elsevier Ad Hoc Networks*, 1(2-3):295–315, September 2003.
9. Adrian Perrig, Rober Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. SPINS: Security Protocols for Sensor Networks. *Wireless Networks*, 8(8):521–534, 2002.
10. Michael K. Reiter and Stuart G. Stubblebine. Resilient Authentication Using Path Independence. *IEEE Transactions on Computers*, 47(12):1351–1362, 1998.
11. Kay Römer. The Lighthouse Location System for Smart Dust. In *Proceedings of MobiSys 2003 (ACM/USENIX Conference on Mobile Systems, Applications, and Services)*, pages 15–30, San Francisco, CA, USA, May 2003.
12. Kay Römer. Tracking Real-World Phenomena with Smart Dust. In *1st European Workshop on Wireless Sensor Networks (EWSN)*, volume 2920 of *LNCS*, pages 28–43. Springer-Verlag, 2004.
13. Robert Szewczyk, Joseph Polastre, Alan Mainwaring, and David Culler. Lessons from a Sensor Network Expedition. In *1st European Workshop on Wireless Sensor Networks (EWSN)*, volume 2920 of *LNCS*, pages 307–322. Springer-Verlag, 2004.
14. Harald Vogt. Integrity Preservation for Communication in Sensor Networks. Technical Report 434, ETH Zürich, Institute for Pervasive Computing, February 2004.
15. S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing Pair-wise Keys for Secure Communication in Ad Hoc Networks: A Probabilistic Approach. Technical Report ISE-TR-03-01, George Mason University, March 2003.
16. Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, pages 62–72. ACM Press, 2003.
17. Sencun Zhu, Sanjeev Setia, Sushil Jajodia, and Peng Ning. An Interleaved Hop-by-Hop Authentication Scheme for Filtering False Data Injection in Sensor Networks. In *IEEE Symposium on Security and Privacy*. IEEE, 2004.

Secure Initialization in Single-Hop Radio Networks*

Mirosław Kutyłowski and Wojciech Rutkowski

Institute of Mathematics,
Wrocław University of Technology
{Miroslaw.Kutylowski, Wojciech.Rutkowski}@pwr.wroc.pl

Abstract. We consider single-hop radio networks, where collisions in the shared channel cannot be detected (no-CD model). A radio channel can be accessed by an adversary trying to degrade functionality of the network, so we are interested in algorithms that work in the presence of an adversary, who knows the algorithm executed and may try make it faulty by injecting own messages. We also focus on algorithms that are time and energy efficient.

We propose a randomized initialization algorithm for a single-hop no-CD radio network. The algorithm has time complexity $O(N)$ and energy cost $O(\sqrt{\log N})$. This is not much worse than the best fragile algorithms constructed so far ($O(N)$ in time complexity and $O(\log\log N)$ energy cost). Our algorithm succeeds with probability $1-2^{-\Omega(\sqrt{\log N})}$ in presence of an adversary, who has energy cost $\Theta(\log N)$.

1 Introduction

Networks based on a radio communication channel have an Achilles' heel – they can be easily attacked by an adversary who causes transmission collisions. If communication algorithms do not take such attacks into account, it may happen that anybody can make algorithm execution faulty. These problems are particularly acute for algorithms that are used for self-organization of the network. One of the most important fundamental tasks is initialization – assigning consecutive labels to the stations in the network.

Our goal is to design an initialization algorithm that is both immune against an adversary and remains efficient in terms of execution time and energy cost.

Radio Network Model. A radio network (RN for short) consists of processing units, called *stations*. The stations communicate through a single shared communication channel. Since a shared communication channel may be implemented by a radio channel, they are called radio networks. Since in general neither the number of stations nor their ID's are known, they fall into category of *ad hoc networks*.

There are many potential applications of such networks, sensor networks for collecting environment data [4] is just one of the examples.

In this paper we consider *single-hop* RN's: if a station is sending a message any other station receives it, provided that its receiver is on. So single-hop RN's are networks

* partially supported by Polish Committee for Scientific Research, grant 7 T11C 032 20 in years 2002-2003, and by grant 3 T11C 033 26 in year 2004.

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 31–41, 2005.

working within a small proximity. The main problem with RN's is that if two stations are sending simultaneously, then a *collision* occurs. In such a case the messages get scrambled. We assume that the result is indistinguishable from a random noise – the model is called *no-CD* RN. This is a pessimistic assumption, since occurrence of a collision may also be a source of valuable information, particularly during the network self-organization phase. In accordance to physical reality and industrial standards, we also assume that a station cannot transmit and listen at the same time.

Communication between stations occur within *time slots* that are the same for all stations. This is possible, since the stations may be synchronized via a radio signal from a common clock. A GPS system can be used for this purpose as well. During a time slot a station either transmits or listens or works only internally. In the first two cases we say that the station is *awake*, in the last one – it is *asleep*.

There are different scenarios considered in the literature regarding the stations and their knowledge: either the number of stations is known, or the number of stations is known up to some multiplicative constant, or the stations have no idea about the number of other active stations. In this paper we design an initialization algorithm, so we assume that stations have no IDs,or they have IDs in a big range $1..R$ where $N = o(R)$.

Further information concerning ad hoc networks can be found in a handbook [8].

Complexity Measures. Given a single shared communication channel two factors may become a bottleneck: execution time and energy cost of transmission. While the issue of time complexity is obvious, energy cost needs some explanation. Many networks consist of battery operated devices. Re-charging or exchanging these batteries might cause many problems – sometimes it is impossible. Even if battery can be replaced or recharged, an interruption due to battery exhaustion might cause problems. So there is a high motivation to design energy efficient algorithms. It turns out that the most costly (in terms of energy consumption) is listening and transmitting messages. On the other hand, energy consumption of processor and sensors can be neglected in such systems.

We define the *energy cost of a station* as the number of time slots, during which a station was awake. *Energy cost of an algorithm* is the maximum energy cost over all stations involved in algorithm execution. For a randomized algorithm $\mathcal{A}$, we say that with probability p the energy cost is $f(n)$, if an execution of $\mathcal{A}$ has energy cost bounded by $f(n)$ with probability p.

Initialization Problem. Some algorithms running on RN's in a multiprocessor environment require that all stations are labeled with consecutive numbers. So we have to solve the following *Initialization Problem*:

> A RN network is given consisting of n active stations. Each active station has no knowledge which stations are active (except itself) and does not know n. To each active station assign a number in the range $[1,\ldots,n]$ so that each number $i \in [1,\ldots,n]$ is used exactly once. After this procedure each active station knows the number assigned to it.

In this paper we may assume that the stations know a number N such that $cN \leq n \leq N$ with high probability for some constant c. Let us remark that there is an efficient size approximation algorithms which finds such an N (see [2]). It seems that they can be tuned to an adversary immune version.

Previous Results. There are many initialization protocols in the literature so far. As we work in the no-CD RN model, we can limit to few ones.

The first algorithm on the no-CD model was designed by Nakano and Olariu [6]. However this method does not succeed when the sending station cannot check the channel status (according to IEEE 802.11 standard).They relied on the tree leader election algorithm, which help them build a sparse table. Then they run a prefix sum algorithm to count the number of alive stations. The algorithm was divided into phases. First, all the stations try to get a temp-ID in the range of 1 to $8n$. Later, all unsuccessful stations try to do it in the range $[8n+1..12n]$, geometrically decreasing the length of the interval range. Then, after shuffling the first $8n$ temp-ID with the rest we get a dense distribution of ID's (in the sense that every interval of length $O(\log n)$ has at least one station with temp ID assigned). So the tree prefix sum algorithm is performed in each of $\frac{n}{\log n}$ groups. Then all the groups perform again this tree prefix sum algorithm and calculate their own ID. As it is obvious that tree algorithms are very fragile to adversary attacks, this algorithm cannot be considered as adversary immune.

A modification of the previous algorithm was presented by Jurdziński, Kutyłowski and Zatopiański in [3] in the no-CD model. The purpose of [3] was partial initialization in which stations are organized in pairs, so that listening while transmitting a message can be emulated. In the first phase authors tried to pair the stations and hence run the initialization algorithm from [6]. Of course that could not initiate all the network, however a vital part of it. Then these stations were used as the echo for the rest of the algorithm and thus a CD model algorithm could be performed, starting from some ID, next after the maximum ID already assigned in the first phase. This algorithm cannot also be regarded as adversary immune.

Another paper [5] presented an adversary immune algorithm that initializes a small number of stations $O(\log N)$. We extend this result to full initialization.

Security Issues. Practical applications of RN's must take into account various dangers resulting from the communication model. Although in a wired network an adversary attacking the system might inject packets with manipulated data concerning their origin, it is possible to trace their origin to some extent. In wireless networks it is much harder. As long as no special physical equipment is deployed, the mobile intruder is safe. Almost all algorithms built so far for RN's disregard this issue. This is a severe drawback, since for many applications ad hoc networks must offer a certain security level.

New Result. Our main goal is to design an initialization algorithm that is energy efficient and tolerates an adversary having limited energy resources. The secondary goal is to preserve linear execution time. (Obviously, $\Omega(n)$ steps are necessary for initialization of n stations in our model.) We get the following result:

Theorem 1. *Consider a single-hop no-CD radio network consisting of $n = \Theta(N)$, $n \leq N$ stations sharing a secret key. Assume that the stations are not initialized with any ID's. Then it is possible to initialize all stations with energy cost $O(\sqrt{\log N})$ within time $O(N)$, so that the outcome is faulty with probability $O(2^{-\sqrt{\log N}})$ in a presence of an adversary which has energy resources of $O(\log N)$.*

So we want to extend the leader election result from [5] using the same energy resources. Initialization is a much harder task, since the consecutive numbers have to be assigned to all stations and not only to a single one.

2 Basic Techniques

In this section we introduce some "building blocks" of our algorithm. We put them separately in order to improve readability of the main part of algorithm description. These techniques are simple, some of them are well-known, standard tools in this area.

Cryptographic Assumptions. We assume that the stations participating in the initialization procedure share a secret s unknown to the adversary, deployed during initialization of these stations. The secret s can be used to derive further keys used for protecting communication. Let us say that at time t the stations sharing s may use key $s_t = F(s,t)$, where F is an appropriate cryptographic pseudorandom one-way string generator. (This protects also from replay attack - repeating by an adversary intercepted transmissions.)

We assume that all messages sent during the initialization procedure are encrypted with such keys so that the messages transmitted cannot be distinguished from a random noise by an adversary. Even on weak devices encrypting with stream ciphers seems to be plausible, since information volume is low and pseudorandom bit streams can be prepared in advance.

Since encryption hides not only communication contents but also communication occurrence from the adversary, we may assume that the knowledge of the adversary is confined to the knowledge of the algorithm executed, its starting point and the approximate number of participating stations. The adversary cannot fake or modify messages of the protocol; the adversary may only try to cause collisions and in this way to break down the protocol. This assumption models also a broad family of transmission faults.

Initial Selection. As one of the building blocks we use the following simple procedure well known from many communication protocols [6, 7]. By a *participant* we mean any station which performs the protocol. Let us assume that there are $n = \Theta(N)$ participants and N is known to all stations (including the adversary); b is a protocol parameter. First each participant P tosses a coin and with probability 0.5 it becomes a *sender*, otherwise it becomes a *receiver*. Then P participates in b rounds. During a round a sender P first decides to be active with probability $1/N$, if it is not active it skips the rest of the round. During a round an active sender P executes the following steps:

step 1: P transmits a random message,
step 2: P listens,
step 3: if P has received its own message in step 2, then it transmits it again.

During a round an active receiver P executes the following steps:

step 1: P listens,
step 2: P transmits the message received in the previous step,
step 3: P listens.

If the sender gets its own message, it knows that there was no collision and exactly one receiver has responded. At step 3 the receiver checks whether there was no collision. In this case we say that a pair of participants, the sender and the receiver, succeeds, and we assign them the round number as an ID number. Such an event occurs with probability approximately $\frac{1}{e^2}$.

In order to keep energy cost $O(1)$ we also assume that a participant remains inactive after the first round in which it was active (no matter whether this round ended with a success or not). Despite that the process is no more a sequence of independent Bernoulli trials, one can prove that the number of successes does not change substantially [2].

Note that initial selection is resistant against an adversary: we assume that its energy capacity is $O(\log N)$, so for $d \gg \log N$, the adversary can attack only a small fraction of rounds.

Relay Procedure. Assume that some stations have unique labels in the range $[1..d]$, say $s_1, \ldots, s_g$, where $s_i < s_j$ for $i < j$. *Relay procedure* informs the station with label $s = s_i$ about number i in the following way: at step $2s-1$ a station P with label s listens – if it receives a message $i-1$, then it means that $s = s_i$. At step $2s$ the station responds by sending $i-1$ again. During the following steps $2s+1$, $2s+3$, ... station P keeps sending i until at some step $2s+2t$ it receives response i. This means that $s_{i+1} = s+t$ and that the station with label $s+t$ takes over.

Relay procedure may be used to initialize the stations that have succeeded during initial selection. The procedure has two major drawbacks: First, if $s_{i+1} - s_i$ is large, then the station with label s_i consumes a lot of energy. Second, the adversary can make a collision in order to prevent delivery of an acknowledgment. After that the whole computation would be faulty.

Time Windows. [5] Suppose that stations A and B share a common secret s and that A has to sent a message to station B at some step i of the algorithm executed. Assume that a time window consisting of r consecutive steps is reserved for this purpose. The problem is that an adversary may try to make a collision and destroy the transmission in this way. In order to avoid such a collision the moment of transmission for step i is determined as $t = f(s,i)$ for some secure pseudo-random function f. For stations A and B the choice is deterministic, while the adversary cannot determine t.

If the adversary detects a transmission, then it is too late for disturbing it. So the adversary may only attack by transmitting at random moments. So, with energy cost m, probability of collision equals $\frac{m}{r}$. So by having a large time window r we may keep collision probability low.

The main drawback of this method is a trade-off between security and time complexity: executing each step of a protocol with a time window of size r increases time complexity by a factor of r. So it is not directly suited to achieve time optimal solutions.

Interleaving Technique. Now we introduce a simple but very useful trick that contributes a lot to efficiency of our solution. Assume we have to perform the same algorithm $\mathcal{A}$ in k independent groups. We can run them concurrently as in *time division multiple access* (TDMA mode): the ith step of group j will be executed in step $(i-1)\cdot k + j$. If $\mathcal{A}$ tolerates a small number of collisions caused by an adversary, then we can modify

TDMA in order to achieve robustness against the adversary without applying time windows technique. Namely, we use a secure pseudorandom permutation generator π: for step number i and secret s it outputs a permutation $\pi(i,s) \in \mathbb{S}_k$. Then step i of algorithm $\mathcal{A}$ in group j is executed at step $(i-1)\cdot k + \pi(i,s)(j)$. So, from a point of view of a single run of $\mathcal{A}$, its every step is executed with time window of size k.

Interleaving technique has one major advantage over time windows: the same effect of confusing an adversary attacking a single groups is achieved, while we do not waste time for leaving most time slots unused for communication. Of course, now the adversary knows that it always hits somebody, but the chances that the same group is attacked repeatedly are bounded.

Adversary Immune Leader Election. Now we describe some features of the leader election algorithm presented in [5]. It has time complexity $O(\log^3 N)$, but essentially the number of steps executed is $O(\log^{3/2} N)$, where each step is executed in a window of size $O(\log^{3/2} N)$. An introductory phase of this algorithm is initial selection executed with parameter $\Phi = \Theta(\log^{1.5} N)$. Later phases are executed by the stations that have succeeded during initial selection.

The main part of the algorithm consists of *group election* executed sequentially in $\Theta(\sqrt{\log N})$ groups, each working on a group of $d = \Theta(\log N)$ consecutive ID's. Once a group elects a leader it prevents all later groups from electing a leader – in this way at most one group elects a leader that becomes the final leader elected by the algorithm. Disabling later groups is achieved by appropriate collisions. (This seems to contradict the assumption that the algorithm is collisions immune; however, the group that has elected a leader knows a secret value used by the following groups to make secret pseudorandom choices of transmission times.)

Group election procedure consists of two phases: the first one is called *building chains*, the second one – *merging chains*. The first phase is a modified relay procedure. The change is that once a station seeks long for the next active station ($\Theta(\sqrt{\log N})$ times), it stops the search. On the other hand, if a station i does not hear any predecessor at step $2i-1$, it assumes that either there is no predecessor or it has stopped due to high energy usage. In this way, the modified relay procedure might fail to link all active stations, and a number of *chains* of linked stations may emerge. Each chain consists of all active stations with ID's from some interval and the station with the highest ID in a chain knows all ID's of active stations in this interval. The second modification in the relay procedure is that one can make it immune against collisions – instead of faulty results (due to a collision caused by an adversary), the worst that can happen is stopping a chain at the moment of attack. This goal is achieved by a simple modification in the relay procedure. The stations work in pairs (as given by the initial selection). Instead of sending a message once, it is sent twice by both stations of a pair. By a simple case inspection one can show that no inconsistency would appear provided that the stations apply a decision procedure described in [5].

During the merging phase the goal is to join the chains. We assume that before the chains are built, the ID's are permuted according to some pseudorandom permutation computed with the secret shared by the stations. So even if the adversary caused a certain number of collisions during the initial selection, the ID's excluded by him would be dispersed over all possible ID's. So the allocation of unused ID's (due to adversary

and to lack of success) can be treated as random from an external point of view. Probability that a chain gets broken due to a gap of size $\Omega(\sqrt{\log N})$ is $2^{-\Omega(\sqrt{\log N})}$. So the main reason for which a chain can be broken are collisions caused by an adversary during execution of the relay procedure – this occurs with probability $O(1/\sqrt{\log N})$. In this case one chain stops exactly where the next one starts. So in order to connect these chains it suffices that the last station in one chain contacts the last station in the next chain. The time moment for this message can be determined based on the ID of the beginning of the next chain – this value is known to both stations. Well, the adversary knows this moment, too, so the next trick is that during merging phase the stations electing the group leader shift the ID's by a pseudorandom value based again on the shared secret.

The outcome of the group selection is either no leader or a set of stations (a chain) in which every station knows all ID's of active stations in the chain. With high probability the number of active stations in a such a chain is $\Omega(d)$. For our purposes we may modify the procedure so that if some boundary number $c \cdot d$ of active stations is not reached (for some constant c), the group switches off. So:

> if the group is not disabled by another group, then with probability $1 - o(1)$ we get a set of $\Omega(d)$ active stations (a chain) such that each station knows all ID's of the active stations in the group. So we can easily initialize the active stations in the group with consecutive numbers.

We see that the leader election procedure from [5] does slightly more than electing leader. In fact, a set of $\Theta(\log N)$ stations are initialized. However simple extensions of this algorithm to an initialization procedure yields a poor solution. The reason is time complexity $\Theta(\log^3 N)$ of the leader election. So for instance applying the leader election algorithm repeatedly on the set of yet uninitialized stations would provide an algorithm with time complexity $\Omega(N \log^2 N)$, while our target is a linear time.

Joining Procedure. Assume that we are faced with the following situation: there are K groups of stations; within each group the stations are initialized and there are at least $r+1$ of them; the groups are labeled with numbers 1 through K and each group member is aware of its group label. We show how we can *join* the groups so that the set of stations of all groups except a few stations becomes initialized.

The joining procedure works as follows: group i would like to learn how many stations are in groups 1 through $i-1$. For this purpose the groups communicate like in relay procedure: at phase i a group $\mathcal{B}$ of r stations work in behalf of groups 1 through $i-1$. First, each station from $\mathcal{B}$ sends a message with the number of stations initialized in groups 1 through $i-1$; the messages are sent in different time slots in time windows of size s_1. The first station of group i chooses at random $l = O(\sqrt{\log N})$ moments out of all transmission moments and listens. If it succeeds at least once, then it retransmits the message received in a **single** moment known to members of all groups in a time window of size s_2. All stations from group i and of $\mathcal{B}$ listen. If the message comes through, then the stations of group i with labels 2 through $r+1$ become the set $\mathcal{B}$. Otherwise, $\mathcal{B}$ remains unchanged and we discard from the initialization all stations from group i with labels higher than $r+1$.

Let $w(i)$ denote the number of stations in groups 1 through i (we take into account the fact of discarding some stations from initialization). Finally, the station of group

i with label j gets label $w(i-1)+j$. The number of stations that get discarded from initialization in this procedure and the energy cost depend on the parameters r, s_1, s_2.

We also consider a *modified joining procedure*: we may assume that a group contains either at least $r+1$ stations or no station. The algorithm works almost as before.

3 Adversary Resistant Initialization

Overview. The initialization algorithm consists of the following phases:

Phase 1: initialization performed concurrently in $k = \Theta(N/\log^3 N)$ groups of polylogarithmic size;

Phase 2: joining the groups from Phase 1 into a set of $D = \Theta(n/\log^2 N)$ initialized stations;

Phase 3: it consists of 4 subphases, each of them increase the number of initialized stations by a factor of $\Theta(\sqrt{\log N})$ with high probability;
each subphase consists of two parts: during Subphase 3a we form *collection groups* served by already initialized stations (so called *servants*), assign some of the remaining stations to the collection groups and initialize these stations within collection groups; during Subphase 3b we join collection groups;

Phase 4: joining the collection groups and initializing their stations with numbers 1 through U, where $U = \Theta(D\sqrt{\log N})$;

Phase 4: having already $\Omega(N)$ stations initialized we use them to initialize the remaining stations in a way analogous as in Phase 3 but with modified parameters.

Phase 1. First, every active station chooses independently at random a number in the range $[1,k]$, where $k = \Theta(N/\log^3 N)$. We say that a station that have chosen number i belongs to *election group* i. Then each election group runs independently the leader election algorithm from [5]. However, instead of using time windows of size $O(\log^{3/2} n)$ in each run of the leader election algorithm we interleave k elections. Then each of k election groups initializes at least $c\log N$ stations, for some constant c, or no station, if leader election fails. Each run requires $O(\log^{3/2} N)$ communication steps, so after interleaving them we use together $O(k\cdot\log^{3/2} N)$ steps, which is $o(N)$. Energy cost is $O(\sqrt{\log N})$. Each run succeeds with probability $1-2^{-f}$, where $f = \Omega(\sqrt{\log N})$, hence with high probability at least a constant fraction of groups succeeds. Moreover, the probability of failure of $2\sqrt{\log N}$ consecutive groups is $O(N^{-2})$.

The stations that become initialized within the election groups are called *local leaders*. Each such local leader knows the index of its election group, the number of stations initialized in this group, and its index within the group. For the sake of simplicity we may assume that each successful group elects exactly $c\log N$ local leaders - simply we disregard the stations with higher index.

Phase 2. We apply *modified joining procedure* from Section 2 for the election groups from Phase 1. The parameters used are: $r = c\log N$, $s_1 = \log^2 N$, $s_2 = \log^3 N$. Then computation time is $O(N)$, energy cost is $O(\sqrt{\log N})$. Let D denote the number of stations initialized during Phase 2. Then with probability $p = 1-2^{-\Omega(\sqrt{\log N})}$ (derived from Stirling formula) $D = \Theta(k\cdot\log N) = \Theta(N/\log^2 N)$.

Phase 3. The goal of this Phase is to get $\Omega(N)$ initialized stations. For this purpose we repeat four times Subphase 3a and 3b – the number of initialized stations increase by a factor of $\sqrt{\log N}$ in each subphase. We start a subphase by splitting already initialized stations into *collection groups* of size $2h$ for $h = 2^{3\sqrt{\log N}}$. Then they help to assign yet un-initialized stations to collection groups. Each group should get $\Theta(h\sqrt{\log N})$ such stations. Then in Subphase 3b we join collection groups.

Subphase 3a. Assume we have $G = \Theta(\frac{N}{\log^t N})$ initialized stations ($t = 2, 1.5, 1, 0.5$). We split them into $H = G/2h$ groups of size $2h$. The stations assigned to group j will be called *servants* of *collection group* j and are indexed 1 through $2h$.

First, each station that is not a servant chooses independently at random a number in the range $[1,N]$. Let us consider a station A that has chosen a number t. If $t > G \cdot \sqrt{\log N}$, then A remains idle during this subphase. Let us consider the opposite case. Let $t = m \cdot H + j$, where $j < H$. Station A listens at step $3t-2$ of Phase 3. At this moment, one of the servants of group j is sending – the servant with index $2u-1$, where $u = m \bmod (\log N)$. The message transmitted is the number of stations v already assigned to collection group j. Station A responds with some control message at step $3t-1$. If there is no collision, then servant $2u$ of group j confirms at step $3t$ that everything went fine. If station A receives this acknowledgment, it regards itself as the station with index $v+1$ among those stations that have joined collection group j and waits till the end of the subphase. At steps $3t-2$ and $3t$, the servants from group j with indexes $2u+1$ and $2u+2$ listen, too. They learn v and recognize whether a new station has joined collection group t.

Consider what happens, if an adversary disturbs communication. If a collision occurs at step $3t-1$, then no station joins the collection group at this moment. If the adversary makes a collision at step $3t-2$ or $3t$, then the servant $2u-1$ cannot guarantee that the servant $2u+1$ is aware of v. In this case the servant $2u-1$ temporarily takes over the next step(s) that would be executed by servant $2u+1$. In this way the adversary cannot cause any inconsistency. The only problem is that it can force some servant to work in behalf of other servants and terminate execution due to energy limit. However, the adversary cannot influence more than $O(\log N)$ collection groups, so it does not reduce the number of assigned stations significantly.

At the end of Phase 3 in each collection group the current number of stations that have joined this group is broadcast. An appropriate message is sent by the servant that was responsible for the last step in which a station could join this collection group. The remaining servants of this collection group and the stations that have joined this group listen. If an adversary collides this message, then joining this collection group is canceled for all stations. This can be repeated $O(\sqrt{\log N})$ times to decrease adversary chances. We can do it without violating energy cost $O(\sqrt{\log N})$ and linear time. In this way all stations assigned to the same collection group get the same view of the situation (except the broadcasting stations – these H stations are excluded from the further computation and finally they are assigned some initial set of labels).

Subphase 3a requires time $\Theta(G \cdot \sqrt{\log N})$ which is $O(N)$. A limitation on energy cost of each servant is $O(\sqrt{\log N})$, the other stations have energy cost $O(1)$. With high probability the number of stations that have joined the collection groups is $\Theta(G \cdot \sqrt{\log N})$.

Subphase 3b. We perform *joining procedure* of the collection groups. The parameters used are: $r = 2h$, $s_1 = 1$, $s_2 = 2h$. Then computation time is $O(N)$, energy cost is $O(\sqrt{\log N})$. Probability of preventing message passing from group i to group $i+1$ is $O((\log N/h)^{\sqrt{\log N}})$ which is $o(N^{-2})$. Colliding broadcast in group $i+1$ succeeds with probability $O(\log N/h)$. Probability that the adversary manages to do it for $\sqrt{\log N}$ consecutive groups (that would interrupt the joining procedure) is $O((\log N/h)^{\sqrt{\log N}}$ which is $o(N^{-2})$.

Phase 4. After Phase 3 we have a set $\mathcal{L}$ of $l = \Omega(N)$ initialized stations and a set $\mathcal{M}$ of $m = O(N)$ yet not initialized stations. Our goal is to initialize stations from $\mathcal{M}$ using stations from $\mathcal{L}$. For this purpose we follow the algorithm of Subphase 3a with slight changes.

The main problem now is not energy cost of servants, but energy cost of stations from $\mathcal{M}$. Our first goal is to assign each station from $\mathcal{M}$ a unique temporary label in the range $[1, \alpha N]$ for some constant α. We assign also the stations from $\mathcal{L}$ to these labels – so that each station is responsible for $O(1)$ labels.

The assignment of stations from $\mathcal{M}$ is performed in $O(\log\log N)$ rounds (the idea is borrowed from [6]): in the first round labels 1 through βN are chosen independently at random by stations from $\mathcal{M}$. A station A that has chosen j sends at step $2j-1$, at step $2j$ the member of $\mathcal{L}$ assigned to j responds with the message received at step j. If there is no response, then there was a collision at step $2j-1$ or $2j$ and A goes to the next round. Since with high probability the number of stations that go to round 2 is at most δN, we reduce the number of labels used in round 2 to $\gamma\beta N$ (with $\gamma > \delta$) and repeat the same procedure. Following the calculations from [6] one can show that after the last round all but $O(\log N)$ stations from $\mathcal{M}$ have their temporary labels in the range $[1, \alpha N]$. As in Subphase 3a, we use collection groups of size $2h$, but now at most $2h$ stations from $\mathcal{M}$ may join a collection group i, namely the stations with temporary labels in the interval $[(i-1)\cdot 2h - 1, i\cdot 2h]$. After initializing collection groups, we run Subphase 3b (which fails with probability $o(N^{-2})$).

Again, the adversary may disturb communication, but only $O(\log N)$ groups might be influenced, making together $O(h\log N)$ stations uninitialized. To get rid of this problem we simply repeat Phase 4 once more. Now probability that an adversary hits any collection group that contains joining stations is $O(h\log N/N)$ which is $o(2^{-\sqrt{\log N}})$.

Conclusions and Future Work

We concentrated ourselves on asymptotic behaviour of the initialization procedure. The choice of parameters was adapted to this goal. In the case of networks of realistic size, the design can be fine tuned. The point is that the same tricks may be used there.

The algorithm proposed works also for the case when an adversary can detect a transmission related to the protocol execution as well as collision that has occurred at such a moment.

References

1. Bordim, J. L., Cui, J., Ishii, N., Nakano, K.: Doubly-Logarithmic Energy-Efficient Initialization Protocols for Single-Hop Radio Networks. IEICE Transactions on Fundamentals '2002, 5 , 967-976
2. Jurdziński, T., Kutyłowski, M., Zatopiański, J.: Energy-Efficient Size Approximation for Radio Networks with no Collision Detection. COCOON'2002, LNCS 2387, Springer-Verlag, 279-289
3. Jurdziński, T., Kutyłowski, M., Zatopiański, J.: Weak Communication in Single-Hop Radio Networks – Adjusting Algorithms to Industrial Standards. Concurrency and Computation: Practice & Experience, 15 (2003), 1117-1131. Preliminary version in: Weak Communication in Radio Networks. Euro-Par'2002, LNCS 2400, Springer-Verlag, 965-972
4. Estrin, D.: Sensor Networks Research: in Search of Principles. invited talk at PODC'2002, www.podc.org/podc2002/estrin.ppt
5. Kutyłowski, M., Rutkowski, W.: Adversary Immune Leader Election in ad hoc Radio Networks. ESA'2003, LNCS 2832, Springer-Verlag, 397-408
6. Nakano, K., Olariu, S.: Energy Efficient Initialization Protocols in Radio Networks with no Collision Detection. ICPP'2000, IEEE 263-270
7. Nakano, K., Olariu, S.: Randomized Initialization Protocols for Ad-hoc Networks. Transactions on Parallel and Distributed Systems '11 (2000), IEEE 749–759
8. Stojmenović, I. (Ed.): Handbook of Wireless Networks and Mobile Computing. Wiley 2002

Some Methods for Privacy in RFID Communication

Kenneth P. Fishkin[1], Sumit Roy[2], and Bing Jiang[1, 2]

[1] Intel Research Seattle, 1100 NE 45th St,
Seattle, Washington, 98105 USA
Kenneth.p.fishkin@intel.com
[2] Department of Electrical Engineering, University of Washington,
Seattle, Washington, 98195 USA
{roy, bjiang}@ee.washington.edu

Abstract. For RFID tags to gain general acceptance, they will have to offer powerful and flexible privacy mechanisms. After reviewing existing and upcoming privacy mechanisms for RFID privacy, we propose that a key aspect of RFID communication with passive tags, namely its required energy transference from an external antenna, may offer promise when developing privacy mechanisms. We present two proposals for such mechanisms. In the first mechanism, analysis of the received signal by the tags can be used to estimate reader distance (and hence trust). We show that a simple metric analogous to signal to noise ratio correlates well with rough distance. In the second, antenna energy is used to power a tiered authentication scheme, in which tags reveal more information about themselves to more trusted and/or "energetic" readers.

1 Introduction

Radio Frequency Identification (RFID) technology is a technology which allows very cheap wireless tags to communicate identification to an interrogating reader located some distance away. Conceptually, this is similar to a bar-code, but the wireless nature of the communication allows for significant qualitative and quantitative advances: the reader need not have line-of-sight to the tag, the tag can store and communicate many more bits of information, multiple tags can be interrogated by the same reader, the reader can be located to read passing tags automatically without explicit user action, and so forth. While the basics of this technology have existed for decades (with their traditional application domain being that of tagging and tracking livestock), recent advances in the capabilities of the tags (operating range, amount of memory, etc.,) coupled with drastic decreases in the cost of both readers and tags (at this writing, tags are available in the €0.30 range (and dropping rapidly), compared to €3.00 a few years ago) has changed the landscape of deployment. The ability to do much more at much less cost has combined to transform RFID from a "fringe" technology into something which is poised to be deployed in the billions per year [1]. However, with this explosion of deployment comes an explosion of responsibility – the responsibility to ensure that the data on the tag is only read by desired readers in desired ways. When RFID was a fringe technology, this was not a major issue. However, this issue must now be addressed, or the RFID explosion may not occur, or may occur in a much more limited fashion.

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 42 – 53, 2005.

The basic RFID scenario is as follows. An RFID reader emits a waveform of some energy at some frequency. That energy is then caught by an antenna on a distant, passive tag. The energy is used to both energize the computation on the tag and energize the returning waveform reply from the tag containing e.g. the tags globally unique ID. Slightly more sophisticated operations are possible, e.g. requests to read/write flash memory on the tag. Note that the reader signal has no disambiguation or identification associated with it.

The historic focus on livestock tracking as the main RFID app has had some positive influences, such as the existence of very physically robust tags (there are tags which can survive autoclaving, which can be pounded into tree trunks, etc.) which can be read without any special user action, it has had some unfortunate influences as well. In particular, there has historically been no emphasis on, or consideration of, privacy or security in the RFID communication: neither livestock nor its owners were concerned with this issue. Accordingly, in most existing RFID protocols, a tag will provide all information unquestioningly to any reader. The second main application domain which has emerged, that of supply chain management (tracking goods from the factory through the distribution center and on to the destination store), has also had only minimal privacy focus: in this realm, RFID tags are viewed as better barcodes, and require no more privacy or security than those do. The original RFID protocols for supply chain management, accordingly, had the same open access protocol as had been used by livestock, i.e. none. However, in introduction to protests from privacy advocates (cf [2]), the proposed protocol has been modified:

2 The Kill Switch

While the standard is still evolving, the predominant proposed privacy mechanism is the so-called "Kill Switch". While the details, again, are still evolving, the basic concept has remained the same (we base our discussion on the most recently published publicly available specification [3]). Each tag has a password (how many bits, and whether the password can be modified, is in flux). When a tag receives a "KILL" command from a reader, accompanied by the appropriate password, the tag essentially "Kills itself" – it sets an internal bit permanently, and so long as that bit has been set, the tag no longer responds to any interrogations from any readers. Conceptually, the tag has de-activated itself. The typical envisioned scenario is that a tag responds unquestioningly to all queries until the good it is attached to is purchased, at which point it shuts down. The idea is that the needs are met of both industry (which is largely focused on tracking the good up to the point of purchase) and privacy advocates (which is largely focused on post-purchase privacy), while requiring only very minimal changes to tag hardware and communication protocols

The proposal is simple and effective in this domain, but has some weaknesses:

1. It is an "all or nothing" privacy mechanism – the tag responds to everyone until the kill switch is set, and then responds to no-one. There is no way to have finer-grained disclosure, e.g. to disclose the expiration date on a pill bottle to a reader in a person's medicine cabinet, but not to anyone else.

2. The user has no way to know whether the tag has actually received the KILL command, let alone that the command was interpreted successfully.
3. It appears that the tag will reveal its password to anyone who asks. Therefore it is very easy for malicious readers to KILL tags prematurely.

A very recent, as of yet unpublished, proposed extension adds a "CONCEAL" command. As long as the CONCEAL bit is set (and unlike KILL, this is a reversible state), the tag will still respond to all queries, but with a random ID. The idea is that this allows a count metric (how many tagged items are in the truck?) without revealing detailed information as to the nature of the items. While this is a positive step away from the "all or nothing" paradigm, it still has the same underlying weaknesses of the KILL command.

3 The Blocker Tag

To address some of the weaknesses and simplicities of the "Kill Switch", a second recent proposal that has received much attention is the "Blocker Tag" proposal of Juels et al. [4].

The Blocker tag exploits a characteristic of RFID communication: the tree-walking protocol a reader uses to determine which tags it sees. When a reader sends a signal out into space, looking for tags, it asks the question: "are there any tags out there whose ID starts with <B>"? (<B> is some bit string). There are three possible answers the reader can receive:

1. No answer.
2. Exactly one answer: the full ID of a sensed tag.
3. More than one answer. In this case, the reader gets a "jumble" of potentially overlapping signal responses – it cannot parse all the responses at one time. Instead, therefore, it recurses, asking for tags whose ID begins with <B>0, and then for tags whose ID begins with <B>1. (Technically, for speed purposes, this can be optimized, for example some current readers immediately issue 8 queries, recursing to depth 3 at one step, but the principle is the same).

The blocker tag responds by *always responding* to the reader query – essentially, it ignores <B>. In this way, it can serve to passively jam the reader, forcing the reader to fruitlessly chase down very long trees.

The blocker tag can act either reflexively or transitively. By a "reflexive" tag, we mean a blocker tag which prevents itself from being read. In this case, the blocker tag is conceptually equal to the "Kill switch", although it is implemented in a very different manner. By a "transitive" tag, we mean that the main purpose of the blocker tag can be to prevent the reader from reading *other* tags nearby. For example, a privacy-conscious consumer might scatter blocker tags about their house or person, to ensure that any other RFID tags in the neighborhood are jammed via their proximity to the blocker tag.

A blocker tag can choose when to act in this hostile manner. For example, there might be "danger zones", so that a reader will only be jammed if it enters a certain "forbidden" area of the tree. This is normally conceived of in a breadth-first way: you can read everything from the tree rooted at <X>, but nothing from the tree rooted at <Y>. For example, the envisioned check-out scenario is that the tag receives a

command asking it to switch its ID from <X> to <Y> - this functionally achieves the similar effect of the KILL switch. Note that this could also be implemented in a depth-first manner. A blocker tag might for example allow its first 14 bits to be queried, yielding information roughly equivalent to a bar code, but then begin jamming if a reader pries beyond that point.

The Blocker tag can therefore be viewed as an extension of the KILL switch, with much more flexibility as to whether it responds, and the ability to "kill by association" nearby tags. It does have some problematic aspects, however:

1. While the blocker tag can control whether it responds, again its only response is to respond with either everything or nothing.
2. The decision as to whether to respond is invariant with respect to the particular reader: all readers who request the same information will retrieve the same result.
3. With time, the jamming can be overcome. Suppose the tree-walking mechanism is in some area of the tree where only the blocker tag exists. Even if the response has been more circumspect such that it only says "I exist", and does not return its ID, the tree-walker can determine that it is in the presence of a blocker tag whenever only one tag responds to bitstring <B>, and yet it receives exactly one response to both <B>0 and <B>1. Therefore, when walking a deep tree, many parts of the tree will, in practice, be quickly pruned significantly.

Our first proposal is similar to the blocker tag, in that we again return to an examination of the idiosyncratic nature of the RFID communication protocol, to see if there are privacy-enhancing techniques available within the RFID environment that are not available in more standard security domains. We combine that concept with another existing concept, that of location-sensitive verification (see e.g. [5]), where the tag seeks to distinguish between different requests for the same information by using knowledge about the spatial location/orientation of the requestor.

We wish to maintain the desirable properties of the KILL switch and blocker tag: a solution which can work with little or no change to tag or reader hardware, little or no change in the communications protocol, and no battery or energy source in the tag. This drastically reduces the space of appropriate techniques. We propose, however, that at least one such technique may be of use which fits those constraints, one which ties the level of information revealed to the distance between the tag and the reader.

4 Distance Implies Distrust

Assume a scenario in which some hostile RFID reader wishes to interrogate (or even change) the information on an RFID tag. Note that often (though not always) such scenarios involve a reader which is physically distant from the receiving tag. This is because the closer the reader is, the more subject it is to scrutiny by the wearers, owners, and/or users of the tagged object. If the tag is located inside a house, for example, it is far more likely that an attack will come from outside the house, than from an intruder inside the house. If the tag is located on a person, it is much more likely that an unwanted, unwarranted request will come from far enough away that the recipient does not see the reader. We therefore propose that RFID privacy mechanisms can and should use the physical distance between the information requestor and the information owner as part of their algorithms.

However, how is a tag to infer the distance between itself and an RFID reader? Existing algorithms such as the ECHO protocol mentioned earlier [5], which use RF and ultrasound transmission to echo-locate, cannot be employed here, due to the extremely difficult RF-only environment. The cost and power requirements of ultrasound are prohibitive in this domain, and pure RF echo measurements (i.e. using the radio both ways) would incur several challenging problems; namely the presence of multipath that could be effectively exploited by attackers and thus necessitate significant advances over the current ECHO protocol. The question is, then, whether we can do rough distance inference given only the RF signal coming from the reader. Several approaches are possible:

4.1 Triangulation via Time-of-Arrival Analysis

One common technique for distance inference is triangulation: we could imagine solutions in which a set of tags compare their received signals, perform cross-correlation, infer time-of-difference in time-of-arrival, and from that infer a fairly accurate distance measurement. However, this solution violates our requirement for minimal cost and infrastructure, and even then, the variations in the RF field might well make the calculation very unreliable.

4.2 Triangulation via Signal Strength Analysis

Another common technique, which requires no inter-tag communication, is to look at the amount of energy received by the tag. By comparing the amount of energy received to that known to be originally sent, a distance estimate can be derived. While this solution requires no infrastructure, and would be very cheap to implement, it will not work in this RF environment, for two reasons. First, there is no way to know how much energy the reader sent out: different readers of different sizes and ranges differ in their energy output, and of course a hostile distant reader could simply increase its energy sufficiently to "mimic" the energy level of a weaker, closer reader. Second, the RFID energy field is notoriously irregular, full of local variations, "nulls", whorls, and so forth, and affected by many different environmental conditions: Fishkin et al. [6], for example, describe 10 different variables that can all affect the signal received by a tag from a reader, and that is not an inclusive list: one simply cannot reliably infer distance from signal strength.

4.3 Noise Analysis

We have found that a minor variation on the signal strength analysis *does* correlate tag distance, fairly well, to received energy. The variation is to look at the *noise* in the received signal, in addition to the strength of the signal. Current RF tags and readers do not allow easy access to their returned signals. We therefore use an approximation: current RF readers do support a "POLL" command, whereby a reader polls for tags in range. If one does a number of poll operations (e.g. 20), we find that the number of responses serves as a reasonable proxy for the signal. The figure below shows a representative example, where the tag is responding to slightly less than 50% of the POLL commands. Note the noise in this signal, showing that even when the environment is fixed, the RF response still fluctuates significantly.

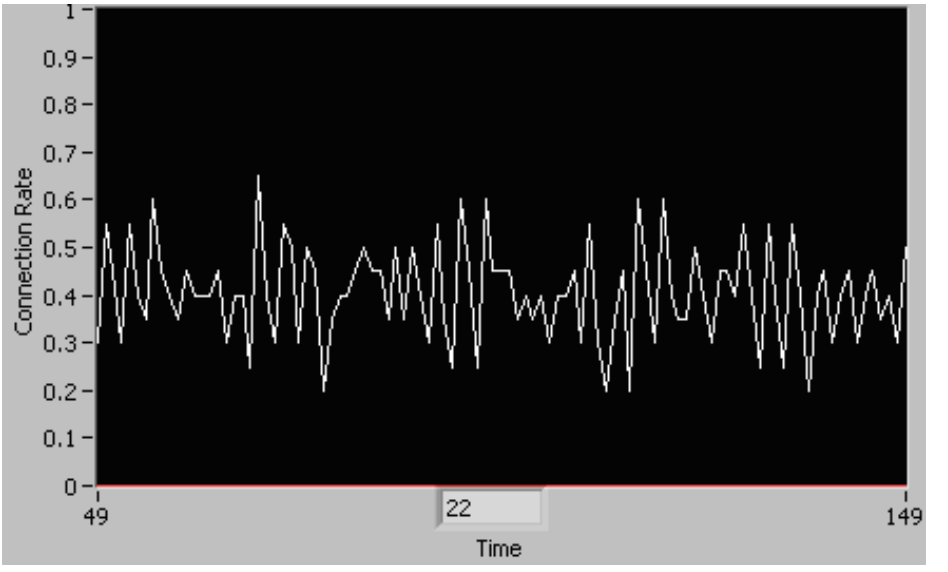


Fig. 1. POLL responses over time

We then performed a series of measurements, using the most common commercially available reader (the Alien technology 915 MhZ reader), with a common commercially available tag, also from Alien technology. As above, the Y-axis represents the response rate (the percentage of responses to a POLL), over 20 polls. The X axis represents time, with 128 such response rate measurements being obtained. Each such X vs. Y plot was then analyzed for mean, standard deviation, and the ratio of standard deviation to mean *FF* (the Fano factor [7], used to approximate signal-noise):

To make the data in this table clearer, Figure 2 graphs μ only: the average signal strength with distance. Note how it demonstrates the idiosyncracies of RFID communication: signal strength sometimes increases with distance (sometimes dramatically), there are often "nulls" (dead spots) between viable distances, and sometimes increasing the angle of the tag to the antenna, which should serve to strictly reduce the amount of energy received and re-transmitted, instead serves to increase it (probably due to reflections off chairs, metal wall beams, etc):

As we can see, there is no reliable correlation between signal strength and distance in practice, especially as a tag cannot know its orientation relative to the reader.

However, if we instead graph *FF* as a function of distance, the graph changes significantly, for the better: see Figure 3.

While the correlation is not perfect, particularly at long ranges, it is generally fairly reliable. An important special case here is that $FF = 0$ at all orientations up to around 3.5 feet. Interestingly, this is also when the Fraunhofer "far field" effect kicks in [8]. This effect begins roughly at a distance of $2\ L*L / \lambda$, where L is the diameter of the reader antenna, and λ is the wavelength of the signal, and is when the energy wave

Table 1. The mean (μ) and standard deviation (σ) of response rate, and the ratio *FF* of the two, as tag distance and orientation with respect to a reader varies

Distance (ft)	Angle (deg.)	μ	σ	*FF*
3	0	1.0	0.0	0
3	30	1.0	0.0	0
3	60	1.0	0.0	0.0
3.5	0	1	0	0
3.5	30	1.0	0.0	0.0
3.5	60	0.99867	0.021929	0.021958
4	0	9,8843	0.010754	0.111834
4	30	0.9957	0.0154	0.015467
4	60	0.448047	0.118289	0.26401
4.5	0	0.72226	0.07705	0.106679
4.5	30	0.48085	0.0748	0. 0.1555
4.5	60	0.05898	0.0534	0.905392
5	0	0.70117	0.078485	0.111934
5	30	0.88632	0.5602	0.632052
5	60	0.7957	0.9114	0.114541
5.5	0	0.5105	0.06826	0.133712
5.5	30	0.54296	0.09108	0.167747
5.5	60	0	----	----
6	0	0	----	----
6	30	0.9957	0.0154	0.015467
6	60	0.28906	0.08033	0.277901
6.5	0	0.99453	0.018	0.018099
6.5	30	0.99648	0.01283	0.012875
6.5	60	0.96446	0.0372	0.038571
7	0	0.4914	0.0905	0.184168
7	30	0.69726	0.05929	0.085033
7	60	0.4207	0.06714	0.159591
7.5	0	0	----	----
7.5	30	0.0371	0.0522	1.407008
7.5	60	0.04335	0.03457	0.797463
8	0	0.05	0.05058	1.0116
8	30	0	----	----
8	60	0	----	----
8.5	0	0.100039	0.060018	0.599946
8.5	30	0.08398	0.06241	0.743153
8.5	60	0	----	----
9	0	0	----	----
9	30	0.01757	0.03111	1.770632
9	60	0	----	----
9.5	0	0.64922	0.08716	0.134253
9.5	30	0.6875	0.081085	0.117942
9.5	60	0.63167	0.67495	1.099858

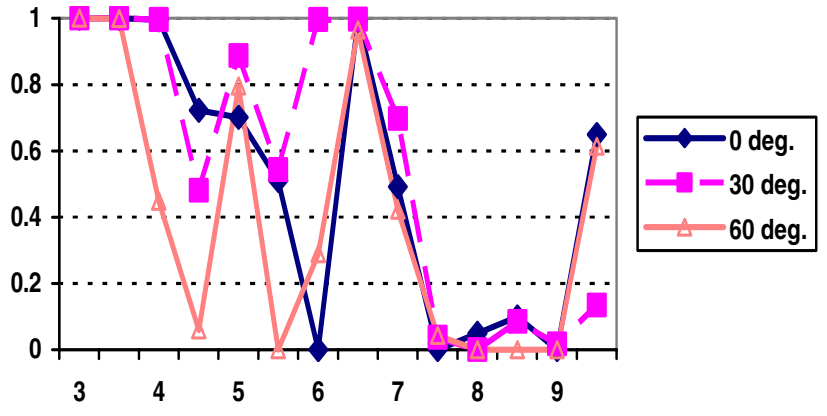


Fig. 2. Mean signal strength as a function of distance

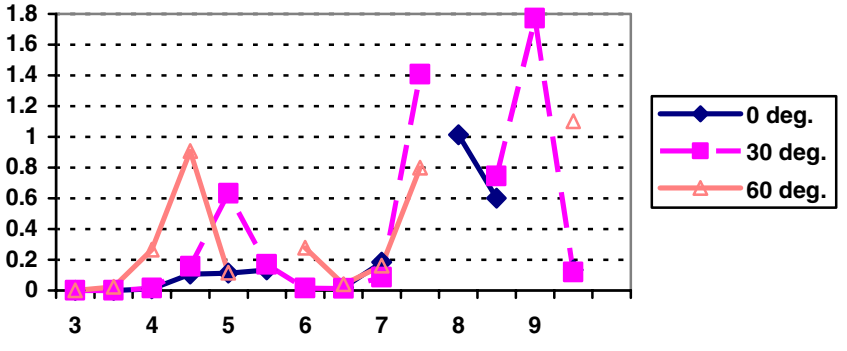


Fig. 3. Noise ratio as a function of distance

from the antenna can be considered in the far field (conceptually, when the curvature of the energy wave is nearly zero). In our case, *L*, the antenna aperture, is roughly 30 cm, and the wave frequency is 915 MhZ, giving us an approximate far-field effect at roughly 1.09m, or 3.5 feet. This means that a tag can not only get a rough estimate of difference (becoming less precise as distance increases) from *FF*, but that it can find a "cliff" between the near and not-so-near distances. A tag on a medicine bottle could, then, for example, return all its information to a reader in the near field (the reader on the shelf in the medicine cabinet should be very near), while returning no or limited information to any reader with a significant *FF*. It is also encouraging that *FF* holds roughly similar across tag orientations, making it a more generally robust and reliable indicator.

Of course, like many Ubicomp sensors, we can see that this is an imprecise sensor: in practice one would have to balance confidence in the sensor value with expected consequences when implementing a range of policies. Our point is that with this

mechanism, which requires no change to the RFID communications protocol, no change to the RFID reader, and only minimal change to the RFID tag, we can now enable a set of such policies. A tag might, for example, decide to turn itself into a blocker tag if it believes its reader to be uncomfortably distant, or only respond with a few bits of its ID, or even respond with false data: the CONCEAL command discussed earlier, for example, is a special case of such a policy: tags could seamlessly report only their existence to a reader located a distance from a truck, but then when the factory worker climbs into the truck with a handheld reader, the tags start responding with more detailed and accurate information. We repeat that we do not claim that this measurement, even if it was perfect, could address all (or even most) of the dystopic privacy scenarios, only that it could address a good chunk of them, and do so with virtually no changes in infrastructure, protocol, or hardware.

Finally, note that a distant antenna cannot imitate a closer antenna, as it could with a metric which relied on signal strength alone, as the *FF* metric would still reveal the difference. A close antenna could imitate a more distant antenna by deliberately introducing noise into its signal, but this serves only to signal greater distance (and hence less trust) than is actually the case: spoofing to pretend less distance (and hence more trust) is not possible; a desirable feature.

5 Tiered Revelation

As we mentioned earlier, whatever the policy to decide *whether* to reveal information, nearly all mechanisms make no control over the *level* of information so revealed: either everything is revealed, or nothing. The CONCEAL switch offers a 1-bit intermediary, but that is only a first step. In this section we sketch a design that incorporates recent proposals in Ubicomp sensor revelation (cf [9,10]) in which data providers *blur* their information to a variable degree depending on the bona fides of the requestor.

Our proposed revelation scheme accordingly moves from a "flat" data space (where either all bits are transmitted or none), to a tiered data space. Every level of disclosure is assigned to a certain tier. When an antenna requests information, it requests a certain tier level. All information at that level and below is transmitted in response. In this way, different readers can be provided with differing amounts of detail, the detail required for their functionality. The more detail the reader requests, the greater its authentication burden, as discussed in the next section.

For example, to show how this tiered revelation might work in practice, let's consider an RFID tagged object such as a shirt made by Benetton. Its tiered information might be structured as follows:

- Level 0 – reports that it is an object. This is useful for baseline testing of a functioning reader, and is equivalent to the CONCEAL level.
- Level 1 – additionally reports that it is a shirt, its fabric, and its color. This is useful for a reader integrated with a washer or dryer – with this information it can tailor its behavior depending on the set of clothes placed in it.

- Level 2 – additionally reports its purchase cost. Now we start to enter the realm of more skeptically granted information. This level would be useful to, for example, an insurance adjuster. By walking through a house with an RFID reader equipped with level 2 authority, it could quickly ascertain the proper amount of insurance needed to cover the objects in the home. By looking at the *FF* in the request earlier, the tag might be able to distinguish between the insurance adjuster and a distant burglar.
- Level 3 – additionally reports which factory it was made at, and at which date. This level is useful to determine if the shirt requires a recall.
- Level 4- additionally reports which store it was bought at, and at which date. This level is useful to determine if the shirt qualifies for a refund/return.

The details of course will vary in practice; this is simply intended as an illustrative example that different levels of information disclosure are needed in different scenarios for the same object.

5.1 Tiered Authentication

This tiered information structure is naturally married to a tiered authentication structure; a reader is provided the information at a given level if and only if it passes an authentication protocol for that level – the higher the level, the more rigorous the authentication protocol.

One method (though by no means the only method) is for the tag and reader to communicate using public-key cryptography, where the number of bits employed is variable, and a function of the desired level of information. In this way, higher and higher amounts of revelation are naturally associated with higher and higher amounts of computation and data transmission.

Therefore, under this protocol, a reader changes its initial request from a blanket request, to a request which indicates:

1) The desired level of revelation
2) The reader public key
3) The number of bits of encryption desired, C
4) The amount of energy received by the tag.

If C (the number of bits of encryption desired) is less than the number of bits the tag requires for that level of revelation, the tag makes no response. Similarly, if the amount of energy received is insufficient, again the tag makes no response; as mentioned below, this provides the advantage of automatically increasing the burden on more physically distant readers. Assuming both of these tests have been passed, the tag responds with a cryptographic challenge-response protocol, keyed to a C-bit encryption.

5.2 Energy-Sensitive Authentication

Our proposed algorithm requires that the reader provide the tag with a minimum amount of energy, and that that amount of energy can be a function of the level of

information disclosure required. This requirement can exist even for tags which are *not* passive, i.e. ones which don't need that extra energy. By still requiring a certain minimal amount of energy to reach them, we can again tie the notion of trust to the notion of physical proximity. Further readers, being less trusted by nature, will have to transmit a greater amount of energy than nearby readers, *even if* they pass cryptographic muster on the other criteria. Therefore, a distant hostile interrogator, even one which has cracked the authentication mechanism, may have to increase their energy level to unattainable levels, or at least levels which can be easily detected by mechanisms such as that discussed in the previous section. Again, the essential idea is that the tag reveals the desired information only if the energy signal passes cryptographic muster, *and* if the received energy is sufficient.

In this protocol, then, a reader conceptually offers up a 3-tuple: the desired level of revelation *R*, the encryption level *C*, and the energy level *E*. We point out two special-cases of this:

1. ***C=0***. In this case, the tag is not being asked to perform any encryption whatsoever. However, we are still achieving at least *some* security, by using the required energy level *E*. By requiring higher levels of energy, we may at least partially satisfy the goal of requiring greater proximity for greater revelation, without requiring any sort of encryption/decryption circuitry on the tag itself.
2. ***E=0***. In this case, the tag is not requiring any particular energy level. In this special case, tag authentication collapses to existing standard methods for tiered information interchange.

6 Conclusions

In this paper, a novel privacy-enhancing technique for RFID is put forward. Based on energy analysis, the mechanism can be divided into two steps. In the first step, an easy-to-compute metric similar to the signal-to-noise ratio is used to estimate the distance between an RFID reader and a tag. In the second step, this distance information is used as a variable in a tiered authentication scheme, where tags reveal more information about themselves to more trusted readers. Trust is a function of (a) perceived distance, (b) level of cryptographic assurance, and (c) level of information desired.

There is a great deal of future work in this area. Given the great requirements for consumer and individual privacy, and the limited and idiosyncratic nature of RFID communication, there are a host of technical, social, cultural, and political issues which will have to be considered to develop satisfactory privacy-preserving RFID deployments. We hope that this paper can help as a stepping-stone along that path.

Acknowledgements

We thank Alien Technology ™ Corporation for providing their antenna data.

References

1. http://www.rfidjournal.com/article/articleview/796/1/2/
2. http://www.spychips.org/press.html
3. auto-id center. “Draft protocol specification for a 900 MhZ class 0 Radio Frequency Identification Tag”, 23 Feb 2003.
4. Juels, A., Rivest, R., and Szydlo, M. The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy. Proceedings of CCS 2003, Washington D.C., pp. 103-111.
5. Sastry, N., Shankar, U.l and Wagner, D.. Secure verification of Location Claims. ACM Workshop on Wireless Security (WiSe 2003). September 2003, pp. 1-10
6. Fishkin, K., Jiang, B., Philipose, M., and Roy, S. I Sense a Disturbance in the Force: Unobtrusive Detection of Interactions with RFID-tagged Objects. Ubicomp 2004 (to appear).
7. Fano, U. (1947). Ionization yield of rations. II. the fluctuations of the number of ions. Phys. Rev., 72:26-29
8. Kraus, J. Antennas. McGraw-Hill. 1988.
9. Gruteser, M., and Grunwald, D. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. MobiSys 2003
10. Hong, J.I., and Landay, J. Architecture for Privacy-Sensitive Ubiquitous Computing. Mobisys 2004. to appear.

Ring Signature Schemes for General Ad-Hoc Access Structures*

Javier Herranz and Germán Sáez

Dept. Matemàtica Aplicada IV, Universitat Politècnica de Catalunya
C. Jordi Girona, 1-3, Mòdul C3, Campus Nord, 08034-Barcelona, Spain
{jherranz, german}@mat.upc.es

Abstract. In a ring signature scheme for ad-hoc access structures, members of a set can freely choose a family of sets including their own set. Then they use their secret keys and the public keys of the other users to compute a signature which enjoys two properties: the external verifier is convinced that all members of some set in the access structure have cooperated to compute the signature; but he has no information about which is the set whose members have actually signed the message.

In this work we propose such a scheme, based on the ideas of a ring signature scheme for discrete logarithm scenarios. The scheme allows the choice of any general access structure, not only threshold ones, as it happened with previous constructions. We prove that the resulting scheme is anonymous and existentially unforgeable under chosen message attacks, assuming that the Discrete Logarithm problem is hard to solve.

1 Introduction

Let us consider the following situation. All the users in some set A agree to sign a certain message. They want the verifier to be convinced that some different users (maybe a quite representative set) have cooperated in the generation of the signature. However, they want that nobody can accuse them of being the authors of the signature.

Members of A could proceed as follows: they could add other users, and form arbitrary subsets of the total set of users. This family of ad-hoc sets is the *access structure* of the scheme. The only restriction on it is that the set A must be one of the sets in the access structure. The goal of the members of A is to use their secret keys, along with the public keys of the rest of users, in order to compute a signature of the desired message, satisfying:

(i) any verifier is convinced that at least all the members of some of the sets in the access structure have agreed to sign the message; and
(ii) any external verifier has no information about which set of the access structure has actually computed the signature.

* This work was partially supported by Spanish *Ministerio de Ciencia y Tecnología* under project TIC 2003-00866.

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 54–65, 2005.

An example of such a situation can be thought inside a company or a forum in the Internet: workers of the company are divided in different branches according to their functionality, and members of the forum can be divided according to their category. Suppose all the workers in some branch of the company (analogously, all the members of the forum with the same category) want to sign a message where they complain about some point of the politics of the company. They want the head of the company to know that many different workers disagree with him, but not to know who is complaining. Members of the complaining branch can form an access structure with all the branches of the company, and compute a ring signature for this structure. The head of the company will be convinced that the complaint comes from all the members of some of the branches, but he will never know which branch dared to complain.

When all the sets in the access structure (including the real signing one) are in fact individual users, then the traditional notion of ring signature scheme is recovered. These schemes were formally introduced in [14], although some proposals had been previously presented, as a basis for the design of group signature schemes (see [5], for example). After the formal introduction of the concept, some ring signature schemes have been proposed for different scenarios [11, 1, 18, 10].

With respect to ring signatures for general ad-hoc access structures, the concept was introduced in [4] (they use the name *ad-hoc groups* to refer to these structures). They give an explicit construction for RSA based public keys, which is valid only for the case where the access structures are necessarily threshold: the sets in the structure are those with a minimum number of members.

Recently, a more general construction has been given in [17]: public keys of the users can be totally independent (different sizes and based on different paradigms), but again the scheme works only for threshold access structures.

In this work we propose a ring signature scheme for general ad-hoc access structures, not necessarily threshold. The construction is based on the ring signature scheme proposed in [10], and is valid if all users have discrete logarithm based keys with common parameters. Similar constructions can be made by using as a basis other schemes presented in [1, 18]. The only restriction is that the public keys of all the users of the system must share the same parameters.

The proposed scheme provides unconditional anonymity: any external verifier of the signature has no information about which set of the access structure is the actual author of the signature, even if he has unlimited computational resources. On the other hand, the scheme is existentially unforgeable under chosen message attacks, assuming that the Discrete Logarithm problem is hard to solve. This means that a valid ring signature for a certain access structure can be computed only if one knows the secret keys of all the members of some set in this structure.

The rest of the paper is organized as follows. In Section 2 we review the concept of ring signatures and the properties that they must satisfy. We recall a result for generic ring signature schemes that we will need to prove the security of the new scheme. This new ring signature scheme for general (not only threshold) ad-hoc access structures is detailed in Section 3. A formal treatment of the

security of the scheme is given in Section 4: first we prove that the scheme is unconditionally anonymous; then, we show that an adversary who corrupts some users cannot obtain a valid signature for an access structure where all the sets contain some non-corrupted user, even after a chosen message attack. We conclude the work and propose some related open problems in Section 5.

2 Ring Signatures

The idea of a ring signature is the following: a user wants to compute a signature on a message, on behalf of a set (or ring) of users which includes himself. He wants the verifier of the signature to be convinced that the signer of the message is in effect some of the members of this ring. But he wants to remain completely anonymous. That is, nobody will know which member of the ring is the actual author of the signature.

A ring signature scheme must satisfy three properties, that we informally describe below.

1. **Correctness:** if a ring signature is generated by following the protocol correctly, then it satisfies the verification equation.
2. **Anonymity:** any verifier should not have probability greater than $1/n$ to guess the identity of the real signer who has computed a ring signature on behalf of a ring of n members. If the verifier is a member of the ring distinct from the actual signer, then his probability to guess the identity of the real signer should not be greater than $1/(n-1)$.
3. **Unforgeability:** among all the proposed definitions of unforgeability (see [9]), we consider the strongest one: any attacker must have negligible probability of success in forging a valid ring signature for some message on behalf of a ring that does not contain himself, even if he knows valid ring signatures for messages and rings, different from the pair message-ring of the forged signature, that he can adaptively choose.

Ring signatures are a useful tool to provide anonymity in some scenarios. For example, if a member of a group wants to leak to the media a secret information about the group, he can sign this information using a ring scheme. Everybody will be convinced that the information comes from the group itself, but anybody could accuse him of leaking the secret.

A different application is the following: if the signer A of a message wants that the authorship of the signature could be entirely verified only by some specific user B, he can sign the message with respect of the ring $\{A, B\}$. The rest of users could not know who between A and B is the author of the signature, but B will be convinced that the author is A.

Recently, ring signature schemes have been also used as a primitive to construct a different kind of signatures, *concurrent* signatures [7].

2.1 Security Results for Generic Ring Signature Schemes

Following the notation introduced in [10], *generic* ring signature schemes can be described as follows. Consider a security parameter k, a hash function which outputs k-bit long elements, and a ring $\mathcal{U} = \{U_1, \ldots, U_n\}$ of n members. Given the input message m, a generic ring signature scheme produces a tuple $(\mathcal{U}, m, R_1, \ldots, R_n, h_1, \ldots, h_n, \sigma)$.

The elements $R_1, \ldots, R_n$ (randomness) take their values randomly in some large set in such a way that $R_i \neq R_j$ for all $i \neq j$; h_i is the hash value of $(\mathcal{U}, m, R_i)$, for $1 \leq i \leq n$; and the value σ is fully determined by the randomness and the message m.

Another required condition is that no R_i can appear with probability greater than $2/2^k$, where k is the security parameter. The following theorem summarizes in some way the security results given in [10], which can be applied to any generic ring signature scheme.

The results are valid in the random oracle model [3], where some hash functions are assumed to behave as totally random functions. This assumption is not true, and some authors have designed schemes which are secure in the random oracle model but cannot be secure with any particular instantiation of the hash functions [6, 12, 2]. However, these schemes are quite artificial, whereas by the moment there exist no such attacks against any practical or realistic scheme. Therefore, a proof in the random oracle model provides a heuristic argument to ensure the security of cryptographic schemes.

Theorem 1. *Consider an attacker $\mathcal{A}$ against a generic ring signature scheme, which can ask a polynomial number of queries to the random oracle. Furthermore, $\mathcal{A}$ has access to a signing oracle: $\mathcal{A}$ adaptively chooses messages and rings, and the oracle returns a valid ring signature for this pair message-ring, as long as all the information that $\mathcal{A}$ would obtain in a real execution of the signing algorithm for this input.*

Assume $\mathcal{A}$ obtains, in polynomial time and with non-negligible probability, a valid ring signature $(\mathcal{U}, m, R_1, \ldots, R_n, h_1, \ldots, h_n, \sigma)$ for some ring $\mathcal{U}$ of n members and some message m such that this pair message-ring has not been asked to the signing oracle.

Then, a replay of this attacker obtains, also in polynomial time and with non-negligible probability, another valid ring signature $(\mathcal{U}, m, R_1, \ldots, R_n, h'_1, \ldots, h'_n, \sigma')$ for the same pair message-ring, with the same random values R_i, for all $i = 1, \ldots, n$, and such that $h_j \neq h'_j$, for some $j \in \{1, \ldots, n\}$, whereas $h_i = h'_i$ for all $i = 1, \ldots, n$ such that $i \neq j$.

The idea in the proof of this theorem is to execute many times the machine $\mathcal{A}$, with the same random tape but with different random oracles for the hash function. This technique, known as *replaying attacks*, was firstly used by Pointcheval and Stern in [13]. After a certain number of executions and with a certain probability, we obtain a new valid ring signature of the same message, with the same randomness but such that the new random oracle has the same outputs than the first one for all the inputs $\{(\mathcal{U}, m, R_i)\}_{1 \leq i \leq n, i \neq j}$ except one, $(\mathcal{U}, m, R_j)$, for

which they have different outputs. This result can be used to reduce the security of some generic ring signatures to other well-known problems.

3 A Ring Signature Scheme for Ad-Hoc Access Structures

Consider the following extension of the concept of ring signatures. Suppose that a set of users $\mathcal{U}_s$ want to anonymously sign some message, in such a way that the verifier of the signature will be convinced that at least the members of some set have all agreed in signing this message, but he could not know which set has actually computed the signature, among the sets of a certain family of sets (the access structure).

Members of $\mathcal{U}_s$ can freely choose the rest of users and the family of sets that will form the access structure. We denote $\mathcal{U} = \{\mathcal{U}_1, \ldots, \mathcal{U}_d\}$ the access structure, where the set $\mathcal{U}_s$ must be one of the sets in $\mathcal{U}$.

The resulting signature will be a ring signature, taking as ring the set $\mathcal{U}$. In this way, the verifier will be convinced that at least *all* the members of some set in $\mathcal{U}$ have cooperated to compute the signature, but he will not have any information about which set in $\mathcal{U}$ is the actual author of the signature.

This extension of ring signature schemes was first considered in [4]. Their specific RSA-based scheme, however, runs only when the ad-hoc access structures are necessarily threshold (that is, they contain all the sets with a minimum number of users). Recently a more general proposal, which allows the use of different types of keys, has appeared in [17]; but again this scheme is valid only for threshold access structures.

We will assume that any specific set of users can always have access to an authenticated broadcast channel, while the information in this channel remains secret to the rest of users. This can be achieved using different cryptographic techniques (for example, broadcast encryption schemes [8]).

3.1 The Proposal

The specific scheme that we present follows the ideas of the ring signature scheme of [10], which is based in turn on Schnorr's signature scheme [15]. Let p and q be large primes such that $q|p-1$ and $q \geq 2^k$, where k is the security parameter of the scheme. Let g be an element of $\mathbb{Z}_p^*$ with order q, and let $H()$ be a collision resistant hash function which outputs elements in $\mathbb{Z}_q$.

Each user U_j of the system has as secret key a random element $x_j \in \mathbb{Z}_q^*$ and the matching public key is $y_j = g^{x_j} \bmod p$.

For a specific signature, some users choose an access structure $\mathcal{U} = \{\mathcal{U}_1, \ldots, \mathcal{U}_d\}$, such that the signing users form one of the sets in $\mathcal{U}$, say $\mathcal{U}_s$. Note that a specific user can be in more than one of the sets in $\mathcal{U}$ (maybe in all of them).

For each of the sets $\mathcal{U}_i \in \mathcal{U}$, we consider the public value

$$Y_i = \prod_{U_j \in \mathcal{U}_i} y_j = g^{\sum_{U_j \in \mathcal{U}_i} x_j \bmod p} .$$

The resulting signature of a message m is a tuple $(\mathcal{U}, m, R_1, \ldots, R_d, h_1, \ldots, h_d, \sigma)$ satisfying $h_i = H(\mathcal{U}, m, R_i)$, for all $i = 1, \ldots, d$, and the verification equation:

$$g^\sigma = R_1 \cdot \ldots \cdot R_d \cdot Y_1^{h_1} \cdot \ldots \cdot Y_d^{h_d} \mod p.$$

It remains to show how can the members of a set $\mathcal{U}_s \in \mathcal{U}$ jointly compute such a valid ring signature. For simplicity, we consider $\mathcal{U}_s = \{U_1, U_2, \ldots, U_{n_s}\}$. The algorithm is the following:

1. Each user $U_j \in \mathcal{U}_s$ chooses at random $\alpha_j \in \mathbb{Z}_q^*$ and computes $R_{s,j} = g^{\alpha_j} \bmod p$. He broadcasts the value $R_{s,j}$.
2. One of the users in $\mathcal{U}_s$, for example U_1, chooses, for all $i = 1, \ldots, d$, $i \neq s$, random values $a_i \in \mathbb{Z}_q^*$, pairwise different, and computes $R_i = g^{a_i} \bmod p$. He broadcasts these values R_i, and therefore all the members of $\mathcal{U}_s$ can compute $h_i = H(\mathcal{U}, m, R_i)$, for all $i = 1, \ldots, d$, $i \neq s$.
3. Members of $\mathcal{U}_s$ compute the value

$$R_s = \left(\prod_{U_j \in \mathcal{U}_s} R_{s,j} \right) \left(\prod_{1 \leq i \leq d, i \neq s} Y_i^{-h_i} \right) \mod p.$$

   If $R_s = 1$ or $R_s = R_i$ for some $i = 1, \ldots, d$, $i \neq s$, they return to step 1. Members of $\mathcal{U}_s$ can then compute $h_s = H(\mathcal{U}, m, R_s)$.
4. User U_1 computes and broadcasts the value $\sigma_1 = \alpha_1 + x_1 h_s + \sum_{1 \leq i \leq d, i \neq s} a_i \mod q$.
5. For $j = 2, \ldots, n_s$, player U_j computes and broadcasts the value $\sigma_j = \alpha_j + x_j h_s + \sigma_{j-1} \mod q$.
6. Define $\sigma = \sigma_{n_s}$. The resulting valid signature is $(\mathcal{U}, m, R_1, \ldots, R_d, h_1, \ldots, h_d, \sigma)$. The values h_i's can be directly deduced from $(\mathcal{U}, m, R_i)$, therefore they do not need to appear in the signature; however, we include them in order to clarify the proof of security.

It is easy to see that such a signature satisfies the required verification condition $g^\sigma = R_1 \cdot \ldots \cdot R_d \cdot Y_1^{h_1} \cdot \ldots \cdot Y_d^{h_d} \mod p$. In effect:

$$g^\sigma = g^{\sigma_{n_s}} = g^{\left(\sum_{U_j \in \mathcal{U}_s} \alpha_j + x_j h_s \right) + \left(\sum_{1 \leq i \leq d, i \neq s} a_i \right)} = \left(\prod_{U_j \in \mathcal{U}_s} R_{s,j}\ y_j^{h_s} \right) \left(\prod_{1 \leq i \leq d, i \neq s} R_i \right)$$

$$= R_s \left(\prod_{1 \leq i \leq d, i \neq s} Y_i^{h_i} \right) Y_s^{h_s} \prod_{1 \leq i \leq d, i \neq s} R_i = R_1 \cdot \ldots \cdot R_d \cdot Y_1^{h_1} \cdot \ldots \cdot Y_d^{h_d} \mod p .$$

3.2 Some Remarks

This scheme allows to detect any possible misbehavior of some of the signers in $\mathcal{U}_s$, because the correctness of the values σ_j can be verified by the rest of the signers, by using known information. Namely, for $j = 1$ the equation

$$g^{\sigma_1} = R_{s,1} y_1^{h_s} \prod_{1 \leq i \leq d, i \neq s} R_i \mod p$$

must be satisfied. For the rest of values of j, the equation that must be checked is

$$g^{\sigma_j} = R_{s,j} y_j^{h_s} g^{\sigma_{j-1}} \mod p.$$

The resulting ring signature scheme can be seen as a generic one, if we consider sets of users instead of individual ones, as possible signers of the ring. We could then use Theorem 1 in order to prove the security of this scheme.

We consider the case where the signing users form an ad-hoc access structure. But the scheme runs as well if the access structure is fixed. In this case the resulting scheme would be in fact a distributed signature scheme (or threshold signature scheme, if the access structure is a threshold one).

If we consider an access structure where all the sets are individual users, then we recover the ring signature scheme proposed in [10]. If we consider an access structure formed by a unique set with a unique user, then we recover the individual Schnorr's signature scheme [15].

The efficiency of the scheme depends on the total number of users and the number of sets in the access structure. Therefore, it is a good solution for situations where the number of sets is small, for example for the ones explained in Section 1. If the access structure is a threshold one, then the number of sets is very large (it is exactly $\binom{n}{t}$, if n is the total number of users and t is the threshold). In this case, we strongly recommend other schemes like the ones presented in [4, 17], which are specific for the threshold case.

4 Security Analysis

In this section we first argue why the proposed scheme achieves the property of unconditional anonymity. Then, we use the generic security result explained in Section 2.1 to prove that the scheme is unforgeable, in the following sense: an adversary who knows the secret keys of some set B of users cannot obtain a valid ring signature for an access structure where all the sets contain some user out of B, even after a chosen message attack.

4.1 Anonymity

Since we are assuming that signing users have a private broadcast channel, the only information obtained by an external verifier is the ring signature itself. This signature can be seen as a ring signature produced by the ring scheme proposed

in [10], where the members of the ring are now the sets of users in the access structure.

Therefore, the unconditional anonymity of the new scheme for access structures directly infers from the anonymity property of the original ring scheme (see [10] for the proof of this property). Roughly speaking, the verifier has no information about which set is the actual author of a given signature, because all the sets have the same probability of having computed it.

4.2 Unforgeability

In order to show that the new scheme is secure, we must first consider what does security mean in this kind of schemes, namely which are the capabilities and goals of an adversary who tries to successfully attack such a scheme.

The adversary is allowed to corrupt a set B of users, obtaining all the secret information owned by these users during the life of the system. The adversary can also make a polynomial number of queries to the random oracle (we assume in the proof of security that the hash function H behaves as a random oracle). Finally, the adversary can require the execution of the signing algorithm for messages and access structures that it adaptively chooses; the adversary obtains a valid ring signature, as well as all the information (secret and public) seen by the corrupted users during the computation of this signature.

We will assume that the adversary only requires executions of the signing algorithm for access structures where all the sets contain at least one non-corrupted user. Otherwise, if some of the sets was formed only by corrupted users, the adversary would have enough information to compute by itself a valid ring signature for this access structure.

Such an adversary is successful if it obtains with non-negligible probability a valid ring signature for some message m and some access structure $\mathcal{U}$, such that:

(i) the pair formed by message m and access structure $\mathcal{U}$ has not been asked to the signing oracle during the attack; and
(ii) all the sets of the access structure $\mathcal{U}$ contain at least one non-corrupted user (out of B).

Finally, we say that a ring signature scheme for access structures is secure (or unforgeable) if there does not exist any successful adversary against it running in polynomial time.

Theorem 2. *Assuming that the Discrete Logarithm problem is hard to solve, the proposed ring signature scheme for access structures is secure, in the random oracle model.*

Proof. The proof consists of assuming that there exists a successful adversary against the proposed scheme, and showing then that the Discrete Logarithm problem can be solved in polynomial time, with non-negligible probability. Since this fact contradicts the assumption that the Discrete Logarithm problem is hard to solve, we conclude that the scheme is secure.

Consider therefore an input of the Discrete Logarithm problem; that is, two primes p and q, an element g with order q in $\mathbb{Z}_p^*$, and an element Y generated by g. The problem is to find the only integer $x \in \{0, 1, \ldots, q-1\}$ such that $g^x = Y \bmod p$.

Let $\mathcal{A}$ be a successful adversary against the proposed ring signature scheme for access structures. The idea is to execute this attack, and extract the solution of the Discrete Logarithm problem from the signature forged by $\mathcal{A}$. It is therefore necessary to simulate the environment of the adversary $\mathcal{A}$.

Let B be the set of users that $\mathcal{A}$ corrupts. During its attack, $\mathcal{A}$ will ask for valid ring signatures for access structures that it will choose; users in the sets of these access structures (including the corrupted ones, in B) will belong to some bounded set. We must provide the adversary $\mathcal{A}$ with the public keys of these users. The process for computing these public keys is as follows. For any user $U_\ell \in B$, we choose at random $x_\ell \in \mathbb{Z}_q^*$ and compute $y_\ell = g^{x_\ell} \bmod p$. On the other hand, for the rest of users $U_j \notin B$, we choose at random $\tilde{x}_j \in \mathbb{Z}_q^*$ and compute $y_\ell = Y^{\tilde{x}_j} \bmod p$. We send all these values to the adversary $\mathcal{A}$.

During its attack, $\mathcal{A}$ will ask for valid ring signatures for access structures that it will choose. For any possible set of users $\mathcal{U}_i$ in such an access structure chosen by $\mathcal{A}$, we have that

$$Y_i = \prod_{U_j \in \mathcal{U}_i} y_j = g^{\gamma_i} Y^{\beta_i} \bmod p \ ,$$

where we know both values $\gamma_i = \sum_{U_\ell \in \mathcal{U}_i \cap B} x_\ell \bmod q$ and $\beta_i = \sum_{U_j \in \mathcal{U}_i \cap B^c} \tilde{x}_j \bmod q$.

Now we must answer all the queries that the adversary $\mathcal{A}$ makes. When $\mathcal{A}$ makes a query $(\mathcal{U}, m, R_i)$ to the random oracle, we choose at random a value $h_i \in \mathbb{Z}_q$ and we send it to $\mathcal{A}$. We store the relation $H(\mathcal{U}, m, R_i) = h_i$ in a random oracle list. In this way, if the same query is asked later, we can send to $\mathcal{A}$ the same output.

We must also show that we can simulate the information that $\mathcal{A}$ obtains from an execution of the signing algorithm. Suppose that $\mathcal{A}$ chooses a message m and an access structure $\mathcal{U} = \{\mathcal{U}_1, \ldots, \mathcal{U}_d\}$. We choose at random one of the sets of $\mathcal{U}$; for simplicity, we denote this set as $\mathcal{U}_s = \{U_1, U_2, \ldots, U_{n_s}\}$, which will be the real author of the ring signature. The information that $\mathcal{A}$ would obtain from such a real computation consists of all the information broadcast in the private channel of $\mathcal{U}_s$ (because we can assume that some of the users in $\mathcal{U}_s$ is corrupted, and so $\mathcal{A}$ has access to this channel), as well as the secret information generated by the corrupted players, in $B \cap \mathcal{U}_s$. The following algorithm shows how to simulate all these values, using again the properties of the random oracle model:

1. For each user $U_\ell \in \mathcal{U}_s \cap B$, choose at random $\alpha_\ell \in \mathbb{Z}_q^*$ and compute $R_{s,\ell} = g^{\alpha_\ell} \bmod p$.
2. Choose, for all $i = 1, \ldots, d$, $i \neq s$, random values $a_i \in \mathbb{Z}_q^*$, pairwise different, and compute $R_i = g^{a_i} \bmod p$ and $h_i = H(\mathcal{U}, m, R_i)$.
3. Choose at random $h_s \in \mathbb{Z}_q$.
4. For user U_1:

- if $U_1 \in B$ (this means that we know the secret key of this user, and the value α_1), compute $\sigma_1 = \alpha_1 + x_1 h_s + \sum_{1 \le i \le d, i \ne s} a_i \mod q$;
- if $U_1 \notin B$, choose at random $\sigma_1 \in \mathbb{Z}_q$ and compute

$$R_{s,1} = g^{\sigma_1} y_1^{-h_s} \prod_{1 \le i \le d, i \ne s} R_i^{-1} \bmod p.$$

5. For player U_j, for $j = 2, \ldots, n_s$:
   - if $U_j \in B$ (this means that we know the secret key of this user, and the value α_j), compute $\sigma_j = \alpha_j + x_j h_s + \sigma_{j-1} \bmod q$;
   - if $U_{s,j} \notin B$, choose at random $\sigma_j \in \mathbb{Z}_q$ and compute

$$R_{s,j} = g^{\sigma_j - \sigma_{j-1}} y_j^{-h_s} \bmod p.$$

6. Compute the value

$$R_s = \left(\prod_{U_j \in \mathcal{U}_s} R_{s,j} \right) \left(\prod_{1 \le i \le d, i \ne s} Y_i^{-h_i} \right) \bmod p.$$

   If $R_s = 1$ or $R_s = R_i$ for some $i = 1, \ldots, d$, $i \ne s$, then return to step 1.
7. Define $H(m, \mathcal{U}, R_s) = h_s$ and store this relation in the random oracle list. Define $\sigma = \sigma_{n_s}$.

It is not difficult to see that in this way the adversary $\mathcal{A}$ obtains consistent values, indistinguishable from those it would obtain in a real execution of the signing protocol. The only delicate point is that the assignment $H(m, \mathcal{U}, R_s) = h_s$, in step 7 of the simulating algorithm, can cause some inconsistency (or collision) if the query $(m, \mathcal{U}, R_s)$ has been previously made to the random oracle. However, the probability of such a collision can be easily bounded, and the probability of performing a successful simulation for $\mathcal{A}$, without collisions, is still non-negligible.

By assumption, $\mathcal{A}$ produces a valid ring signature $(\mathcal{U}, m, R_1, \ldots, R_d, h_1, \ldots, h_d, \sigma)$ for some access structure $\mathcal{U}$ of d sets and some message m such that this pair message-access structure has not been asked to the signing oracle, and such that there is at least one non-corrupted user in each of the subsets of the ring $\mathcal{U}$. Since the ring signature scheme is generic, we can apply Theorem 1, and so we obtain in polynomial time and with non-negligible probability another valid ring signature $(\mathcal{U}, m, R_1, \ldots, R_d, h'_1, \ldots, h'_d, \sigma')$ for the same access structure $\mathcal{U}$ and the same message m, with the same random values R_i, for $i = 1, \ldots, d$, and such that $h'_j \ne h_j$ for some $j \in \{1, \ldots, d\}$, whereas $h'_i = h_i$ for all $i = 1, \ldots, d$ such that $i \ne j$.

Let us consider the two corresponding verification equations, satisfied by these two valid ring signatures:

$$g^{\sigma} = R_1 \cdot \ldots \cdot R_d \cdot Y_1^{h_1} \cdot \ldots \cdot Y_j^{h_j} \cdot \ldots \cdot Y_d^{h_d} \mod p$$

$$g^{\sigma'} = R_1 \cdot \ldots \cdot R_d \cdot Y_1^{h_1} \cdot \ldots \cdot Y_j^{h'_j} \cdot \ldots \cdot Y_d^{h_d} \mod p.$$

Dividing these two equations, we obtain the equality $g^{\sigma-\sigma'} = Y_j^{h_j-h'_j} \bmod p$. Since there is at least one non-corrupted user in the subset $\mathcal{U}_j$, we have that $Y_j = g^{\gamma_j} Y^{\beta_j} \bmod p$, and $\beta_j \neq 0 \mod q$ with overwhelming probability. Then, we can write

$$g^{\sigma-\sigma'} = g^{\gamma_j(h_j-h'_j)} Y^{\beta_j(h_j-h'_j)} \bmod p \ .$$

Therefore, we have found the discrete logarithm x of Y with respect to the base g, which is:

$$x \ = \ \frac{\sigma - \sigma' - \gamma_j(h_j - h'_j)}{\beta_j(h_j - h'_j)}.$$

This inverse is computed modulo q, and it always exists, since $h_j \neq h'_j$ and $\beta_j \in \mathbb{Z}_q^*$ with overwhelming probability.

□

5 Conclusions and Future Work

In this work we consider an extension of ring signature schemes to a scenario where some set of users cooperate to compute an anonymous signature. The signing users can choose an arbitrary access structure of possible sets of signing users, with the restriction that the actual set of signers must be in this structure. The verifier of the signature will be convinced that at least all the members of some of the sets in the structure have cooperated in computing the signature, but he will not have any information about which is the actual signing set.

We give an explicit construction, which can be seen as an extension of the ring signature scheme of [10], where all the public keys of the users are discrete logarithm based, with the same common parameters. We formally prove the unconditional anonymity and the computational unforgeability of the new scheme: assuming that the Discrete Logarithm problem is hard, an adversary who corrupts some set of users cannot obtain a valid signature for an access structure where all the sets contain some non-corrupted user. The proof is in the random oracle model.

Using similar ideas, it is possible to construct schemes with the same properties, from other ring signature schemes [1, 18]. However, solving the problem for arbitrary access structures remains as an open problem in other scenarios. For example, if the public keys of the users are RSA based, there exists a solution proposed in [4], but only for the threshold case, where the sets in the access structure are necessarily those with a minimum number of users. More generally, in the case where the public keys of the users are all independent (different sizes, and based on different paradigms), a solution to the problem can be found in [17], but again it is valid only for the threshold case.

References

1. M. Abe, M. Ohkubo and K. Suzuki. 1−out−of−n signatures from a variety of keys. *Advances in Cryptology-Asiacrypt'02*, LNCS **2501**, Springer-Verlag, pp. 415–432 (2002).
2. M. Bellare, A. Boldyreva and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. *Advances in Cryptology-Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 171–188 (2004).
3. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. *First ACM Conference on Computer and Communications Security*, pp. 62–73 (1993).
4. E. Bresson, J. Stern and M. Szydlo. Threshold ring signatures for ad-hoc groups. *Advances in Cryptology-Crypto'02*, LNCS **2442**, Springer-Verlag, pp. 465–480 (2002).
5. J. Camenisch. Efficient and generalized group signatures. *Advances in Cryptology-Eurocrypt'97*, LNCS **1233**, Springer-Verlag, pp. 465–479 (1997).
6. R. Canetti, O. Goldreich and S. Halevi. The random oracle methodology, revisited. *Proceedings of STOC'98*, pp. 209–218 (1998).
7. L. Chen, C. Kudla and K.G. Patterson. Concurrent signatures. *Advances in Cryptology-Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 287–305 (2004).
8. A. Fiat and M. Naor. Broadcast encryption. *Advances in Cryptology-Crypto'93*, LNCS **773**, Springer-Verlag, pp. 480–491 (1993).
9. S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptative chosen-message attacks. *SIAM Journal of Computing*, **17 (2)**, pp. 281–308 (1988).
10. J. Herranz and G. Sáez. Forking lemmas for ring signature schemes. *Proceedings of Indocrypt'03*, LNCS **2904**, Springer-Verlag, pp. 266–279 (2003).
11. M. Naor. Deniable ring authentication. *Advances in Cryptology-Crypto'02*, LNCS **2442**, Springer-Verlag, pp. 481–498 (2002).
12. J.B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: the non-committing encryption case. *Proceedings of Crypto'02*, LNCS **2442**, Springer-Verlag, pp. 111–126 (2002).
13. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, Vol. **13** (3), Springer-Verlag, pp. 361–396 (2000).
14. R. Rivest, A. Shamir and Y. Tauman. How to leak a secret. *Advances in Cryptology-Asiacrypt'01*, LNCS **2248**, Springer-Verlag, pp. 552–565 (2001).
15. C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, Vol. **4**, Springer-Verlag, pp. 161–174 (1991).
16. A. Shamir. Identity-based cryptosystems and signature schemes. *Advances in Cryptology-Crypto'84*, LNCS **196**, pp. 47–53 (1984).
17. J.K. Sui Liu, V.K. Wei and D.S. Wong. A separable threshold ring signature scheme. *Proceedings of ICISC'03*, LNCS **2971**, Springer-Verlag, pp. 12–26 (2004).
18. F. Zhang and K. Kim. ID-based blind signature and ring signature from pairings. *Advances in Cryptology-Asiacrypt'02*, LNCS **2501**, Springer-Verlag, pp. 533–547 (2002).

Linking Ad Hoc Charging Schemes to AAAC Architectures

Joao Girao[1], Bernd Lamparter[1], Dirk Westhoff[1],
Rui L. Aguiar[2,3], and Joao P. Barraca[3]

[1] NEC Europe Ltd., Heidelberg, Germany
{joao.girao, bernd.lamparter, dirk.westhoff}@ccrle.nec.de
[2] University of Aveiro, Portugal
ruilaa@det.ua.pt
[3] Institute of Telecommunications, Aveiro, Portugal
jpbarraca@av.it.pt

Abstract. The current state of today's networks allows us to take one step further in merging the research community's work with every day's life. Wireless ad hoc networks are already well developed for specific scenarios. This work shows how to build the link between the wired network and a wireless ad hoc infrastructure, in particular routing and AAAC aspects. Such integration might lead, for example, to a better spacial and resource distributed hotspot solution.

We provide the basis for inter-operation of AAAC[1] protocols known for the fixed network, with the accounting protocol that performs the accounting and charging functions in the ad hoc network.

This paper further describes the implementation of the Secured Charging Protocol as an instantiation of a charging protocol for ad hoc networks and the features which were added to improve the interface to an external accounting system. It covers the interaction with the MANET routing protocol and how to deal with routes to or from outside the ad hoc cloud.

1 Introduction

The Internet is today's best example of the fixed networks achievement over the years. These networks have proved to be robust and to have a very high success rate. Nevertheless, technology evolves and the use of wireless devices has become as common as that of their wired counterparts. Hybrid systems are now common in homes, offices and leisure environments. Even widespread wireless networks are no longer a tendency, but a reality.

The research community has long studied these matters and as a result, ad-hoc networks have been developed, and are been looked as a communication paradigm for the future. However, we should not neglect the years of experience

[1] Authentication, Authorization, Accounting and Charging.

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 66–81, 2005.

on routing, security, QoS and network management done, even if founded on wired networks. We must use such value and adapt it into this new arena.

Hotspots have appeared as points of access to the Internet via a wireless medium in places where conventional wiring would be inconvenient or even impossible. This concept of network access seems to be able to benefit from the usage of ad hoc like infrastructures in order to maximize network connectivity.

Unfortunately, ubiquitous wireless multi-hop ad hoc networks are inherently troublesome due to privacy and security issues. Security and routing is also linked to the fact that some nodes may not wish to cooperatively forward for other node's traffic. This is why certain mechanisms have been engineered to avoid the selfish nodes problem (the case when the ad hoc node does not forward packets which are unrelated to it).

There are two main approaches to the problem: The first is to identify misbehaving nodes and exclude them from the community [1], [2]; the second, with which this paper deals, is to lead nodes to cooperation through the use of some incentive, such as proposed in Nuglets [3], Sprite [4] and SCP [5].

SCP is based in the notion that all nodes must gain when a packet is received correctly at its final destination. Nodes that send and receive data benefit from the other's cooperation, and have to pay for this service. Nodes that forward data receive money as an incentive for helping others. Since this is a highly flexible communication paradigm where a sending node can forward or vice-versa; nodes can use the money they earn by forwarding foreign data, to send and receive their own.

This is the business model adjacent to SCP, which can be used in the hotspot scenario by extending node connectivity to an ad hoc network, and provide Internet connection to a wider range of clients. Within an ad hoc cloud, nodes are persuaded to forward each other's traffic with the incentive earning money for it. On the other hand, nodes that do want to use the network resources will not mind paying accordingly for the amount of resources consumed. They know that SCP helps improving co-operation and increases the end-to-end reliability of the network.

This paper provides a realistic solution to the interaction between ad hoc and fixed networks in what concerns authentication, authorization, accounting and charging. Its main contributions include a possible way in which to deploy an ad hoc network that maintains services and applications from the fixed networks.

We present a practical, yet powerful approach to fasten the process of integration between the studied ad hoc mechanisms and current network architectures.

In the next section we introduce the Secured Charging Protocol. We continue with the implementation description of the protocol in Section 3 and 4. Section 5 introduces the integration with the routing protocol and the mechanisms necessary to extend SCP to the Internet. In the following two sections we insert SCP in the infrastructure. We further describe our testbed and results on the implementation in Section 8. We finish the paper with a discussion on future work in this area and our conclusions.

2 The Secured Charging Protocol

As we have previously argued, cooperative networks are a strong candidate to achieve realistic connectivity in a potentially selfish environment. Obviously, selfishness is particularly felt in civilian ad hoc networks with power restricted and battery constrained devices and networks open to everyone. We propose to overcome this problem and increase cooperation by using SCP. We describe the applied primitives of SCP, the assumed architecture and give an overview on the protocol.

2.1 Protocol Architecture

SCP assumes an architecture consisting of several different components. They are located in the ad hoc network and in the fixed network:

- The AAAC architecture in the fixed network is composed by the AAA Foreign Server (AAAF) and the AAA Home Server (AAAH). The AAAF belongs to the same domain as the Access Router (AR) of the ad hoc cloud. The AAAH is able to identify the node as belonging to his administrative domain. If a mobile node has not left its administrative domain, both aforemntioned AAAC components are physically the same.
- The AR is the first point of contact a node has with the network. Initially, it solely allows nodes to access the AAAC system so they can register and provide accounting information. The AR performs the role of a translator between the SCP and other AAAC protocols specific to the fixed network.
- A node of the ad hoc cloud may act in different roles depending on its place in the routing path:
  – Should the node be the first in the path, it is the sender (we will also use the notation of Mobile Node (MN) for this node). Its objective is to transmit packets in a multihop fashion across the (wireless) network even if this means he has to pay for this service.
  – If the node is receiving the packets as the final destination, it is the receiver (We will refer to this node as the Correspondent Node (CN)). For the service of receiving packets from the MN, this node is also willing to take part in the payment of the bill.
  – Any other node in the path that forwards traffic is a Forwarding Node (FN). These nodes are willing to forward packets for other nodes since they subsequently receive incentives in the form of money.
  – A special case of a FN is the Last FN (LFN), the immediate predecessor of the CN. This node occupies the last position in the routing chain before the CN. More than the functions of a normal FN, a node in this role is also responsible for transmitting the accounting information to the AR (when a connection to the AR is available) and before a charging period of the Internet Service Provider (ISP) has expired.

Note that each node may act in different roles. It may play the role of MN, CN, FN or LFN. In fact, it might even be in several roles at the same time when multiple communication sessions are active.

2.2 Protocol Overview

SCP operation can be subdivided into three different phases:

1. Registration Phase
2. Forwarding Phase
3. Charging Phase

In the one-time Registration Phase, a node will seek initial authentication within the domain the ad hoc network belongs to [6]. To do so, it may or may not have to contact its home AAAC server, e.g. via a challenge response protocol. The Security Association (SA) between different AAAC servers (AAAF and AAAH), between the AR and the AAAF and between the node and the AAAH are considered to be static. Thus, SCP does not create new SAs but simply establishes the previously known connection between the MN and the AAAH.

Registration occurs only when the node arrives to a new network where it is not authenticated or when its authentication needs to be refreshed due to an expired certificate.

Should the AR be able to prove the node's identity, it will respond with a valid certificate the node can use in his domain and a shared secret. It will also provide charging information, namely the factors by which the node will pay or be rewarded in an Access Response message.

If the AR has no confirmation on the node's identity, it will respond with an Access Response containing an error message denoting the incorrect parameter or the cause for the error.

Once the Registration is complete, a node is allowed to send, receive and forward data.

The Forwarding phase is best described as the act in which a node decides to pass a data packet to the next node in the path. This builds up to the sending and receiving of data from source to final destination. During this phase the sending node will sign the packet. Intermediate nodes will then verify the signature of the sender and include their marking on the packet by means of a hash chain. They also add themselves, should this be required[2], to the route contained in that same packet.

This hash chain is extended by hashing the previous value and a shared secret between the node and the AR. A random start value, also transmitted with the message, assures the freshness of this data.

If a node cannot verify the signature contained in the packet, the packet must be dropped since the node now is assured it will not receive any incentive for that particular transaction.

We implement the forwarding information needed by SCP as an extension header to the IPv6 protocol over which it travels. All information needed for accounting is in the packet itself.

A periodic exchange of two confirmation messages that we now describe allows for the synchronization between the LFN and the CN in what concerns the

[2] Certain routing protocols, such as DSR [7], already provide this information.

amount of data actually received. This exchange occurs only seldom and can be triggered after a certain number of packets, bytes or an amount of time. The messages used are:

Confirmation Request. This packet is sent by the LFN to the CN to ascertain the amount of data the node has received correctly. This packet is a contention mechanism not to overcharge the end-to-end pair. Since digital signatures are applied here, the information contained in this packet is non-repudiative.

Confirmation Response. As a reaction to the former packet, this message contains information on the amount of data the node received for that session.

Finally, the Charging phase of the protocol corresponds to the sporadic exchange of two packets, which takes place depending on the availability of the Access Router (this exchange of packets always occurs after the Forwarding Phase and before the end of the ISP's charging phase):

Verification Request. The LFN now sends a Verification Request message to the AR to perform the accounting function. The relevant information this message contains is the session, the route, a list of hash chains corresponding to each of the packets, the number of bytes accounted for and the number of bytes acknowledged by the CN[3].

Verification Response. The AR confirms the received accounting information.

All the packets in SCP, with the exception of the Access Request, are signed by the sender of the message, independently of whether it is data or the signaling flow. This, together with a Certifying Authority (CA) that makes sure the keys being used by the participants are trustworthy, provides a per bundle or per packet authentication.

2.3 Cryptographic Primitives

SCP makes use of two basic cryptographic primitives:

- Hash functions
- Digital Signatures

The choice of both primitives relates to the envisioned security level of SCP. We opted for a 16 byte MD5 as it provides sufficient security for the expected short lifetime of the values. A random number provides freshness to avoid replay attacks. MD5 is preimage and second preimage resistant, as well as collision resistant. These features combined with the short byte length distinguished it as a reasonable candidate. Other choices such as SHA-1 were also considered but MD5 was found preferable due to the size of the hashed value.

In the communication scenario proposed for SCP, a symmetric key scheme would not scale. Due to its distributed nature and the fact that communication does not necessarily share any common point, every communication pair

[3] The path from the LFN to the AR is again considered to be a multihop route. Nodes which will forward these messages will receive incentives.

would need its own shared secret. Although symmetric algorithms are generally faster, the low resources of the end devices make it impossible to hold one different symmetric key per peer node. Therefore, we chose to adopt an asymmetric cryptographic scheme which allows for a single public and private key pair per host. The public part of the key is then shared within the network and can easily be broadcasted in form of a certificate over the wireless medium. Since we assume temporary connectivity to the fixed network where a Certifying Authority (CA) resides, we have no need for distributed approaches.

For the digital signature choice we considered RSA [8] and Elliptic Curve Cryptosystem (ECC) [9]. Although the first provides faster execution times for the verification operation, which is especially important since SCP requires verification at each intermediate node, the size of the key and other necessary resources to run RSA have pushed it to the second position. ECC produces less overhead in the network and in storage requirements at the end points. In particular, less overhead is imposed by SCP on the network which allows the protocol to be suitable even for real time traffic. We have used our own speed optimized ECC implementation [10] using the Fixed-base comb method and the Montgomery method. Table 1 shows the execution times for different security levels. These measurements were done using a PDA device (the Sharp Zaurus) which we assume to be a realistic destination platform for a charging scenario in ad hoc networks. For SCP, the key size was chosen in accordance to a low lifetime system. This translated into a 163 bit key which is the equivalent security to a 1024 bit RSA. For detailed information, we again refer to [11].

Table 1. Execution Times for Signature Operations based on ECDSA and RSA on a Sharp Zaurus SL-5500G

Security level bit		Time for signature generation [ms]			Time for signature verification [ms]		
ECC	RSA	ECC	RSA	Ratio	ECC	RSA	Ratio
113	512	2.8	13.7	4.9	7.5	1.3	5.7
131	704	3.8	32.4	8.5	11.5	2.5	4.6
163	1024	5.7	78.0	13.6	17.9	4.3	4.1
193	1536	7.6	251.9	33.0	26.0	9.7	2.6
233	2240	10.1	731.8	72.0	37.3	20.4	1.8

3 Implementation Overview

SCP is implemented using C++ for the Linux Operating System (OS). Modules and external libraries are written mostly in the C language. The chosen OS offers many of the utilities and libraries required to work with IPv6, packet capturing and multi-tasking/multi-threading.

We have also chosen two distinct target hardware platforms: the i386 compatible PC (AR) and, the Zaurus PDA (MN, CN, FN, LFN). The latter was chosen to easily test mobility and demonstrate the protocol functionalities. Although most PCs share common architectures, this is not the case for PDAs. Our choice was driven both by the fact that the Zaurus is a PDA in the market that natively runs Linux, and the convenience of the available open source compiler for the StrongArm processor, core of this PDA.

We position SCP between the network and the transport layer. It has strong interaction with the ad hoc routing protocol as well as high level functions of the OSI model which are accessible to the application.

Certain functionalities, such as the underlying ad hoc routing protocol and the authentication methods, have been modularly built into SCP. This simplifies the process of adapting SCP to a certain network. As a side effect, they can also be dynamically loaded at runtime. For this purpose we used the Dynamic Loading Library (libdl) which Linux provides. Through the use of this library, we are able to load external functions, not compiled with our main program, at runtime according to a configuration file. After the program has read the configuration file, it loads the described modules for both routing and AAAC [12]. Moreover, the configuration can be changed and reloaded without stalling SCP. This allows a network provider to update and reconfigure all nodes and the access routers without interrupting the service.

At the Access Router, not all SCP related information should be kept in main memory as some of it is of persistent nature. For this data, we use the relational database model present in mySQL (a free SQL database, available also for Linux). Per node and per charging period SCP data is then inserted and retrieved using a set of SQL queries and a C interface to mysql. In a similar fashion, we combined mySQL databases with the Pluggable Authentication Modules (PAM). PAM is provided by Linux to form a simple modular authentication backend.

Linux itself provides the facilities to queue packets and process them in userspace, where the prototype realization of SCP resides. This greatly simplifies network level operations. Packets are captured into IP Queue (IPQ) using iptables on the hooks provided by the Linux netfilter architecture. The packet is then sent to userspace and processed. Once they are ready to be sent, they are simply re-injected back into the network with possible changes to their payload and format. Should the packet verification fail or any other error occur, the packet is dropped.

By using the capabilities of C++ to their full extent, we have been able to develop a system in which packets are read and built from the bottom and up and then delivered to the higher protocol layers. Each protocol layer was fitted into its own Class which, through inheritance, forms the actual layers as seen in the network.

As to the implementation of some of the security functions and abstraction layers, we used the Secure Socket Layer (SSL) [13] model provided by the Linux OpenSSL project, since it already provides a great amount of cryptographic

primitives. We have even extended OpenSSL to include our speed optimized ECC library.

Both the timer and the event systems required for asynchronous execution have been built into SCP. The functionalities have been divided into different threads that allow concurrent execution, therefore improving scalability. On top of this, the system provides event queuing for function scheduling.

3.1 Implementation Building Blocks

The SCP implementation architecture can be divided into four main layers, as depicted in Fig 1. The first is the Network Layer which consists of the transport for both the signaling and the data flow. In addition we define the kernel as the interface between the Network Layer and our application. For the specific SCP related implementation, we provide two more levels of abstraction: while the Packet Handler deals with the complexity of an event driven multi-threaded system, the SCP implements the actual state machine and contains the interaction between users, sessions and the databases.

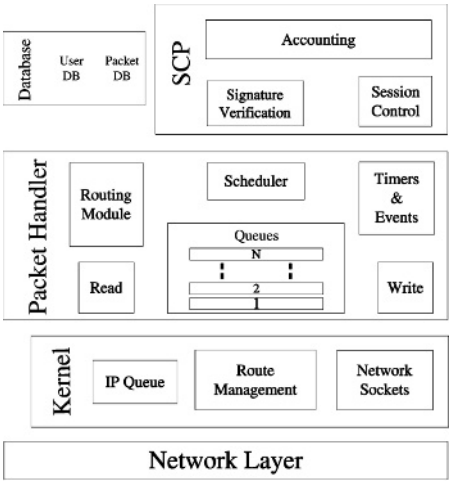


Fig. 1. Software Architecture

3.2 Building Blocks

The building blocks of our implementation are as follows:

Network Layer. We refer to the actual physical layer, the link layer and the IP layer as the Network Layer. It provides the data transmission protocols and mechanisms to retrieve information on the network.

Kernel. We focus on the Linux kernel although the model can be ported to other OSs.

IP Queue - The IPQ system provides packet queuing and processing in userspace by using iptables and special libraries.

Route Management - This part of the kernel deals with the creation and deletion of IPv6 routes. It is mainly used by the routing module to define the path a data packet takes.

Network Sockets - This is the kernel implementation of the Berkeley socket API. It provides the main functions to make use of the transmission protocol implementations from the Network Layer.

Packet Handler. The Packet Handler is an abstraction to provide a simple interface to the most common and basic operations within the program. It is the basis used by the state machine that implements the actual protocol.

Read Thread - This thread is responsible for the polling of new data from IPQ. It reacts to a new message by putting it into the SCP internal queues for processing.

Write Thread - This thread polls the queues for packets that should be sent to the Network Layer.

Routing Module - The routing module is an independent piece of software that discovers the paths between two nodes that wish to communicate. The current implementation supports a module based on static routing (for testing purposes) that depends on a configuration file, and another one using a modified version of the AODV6 HUT implementation.

Scheduler - The scheduler is responsible for service differentiation within SCP. It distributes the most resource consuming operation, the signature verification, unevenly between the different flows[4].

Queues - The Queues represent the form in which SCP differentiates classes of traffic. Packets are grouped together using their IPv6 header flow label field and a different weight is attributed to each queue. This weight is then translated in the processing time each of the queues receives to run the verification function.

Timers and Events - Because SCP is event driven, we define a special class for timers and events. Both of these classes provide support for asynchronous function calls.

Database. This building block provides an interface to the mySQL database.

User Database - This table contains the users identity and attributes to each user a username and a password. Authentication is then performed by using this table from within PAM.

Packet Database - Once the accounting packet has been verified, the information contained in it is dumped in this set of tables for persistent storage.

[4] We define a flow as being a packet with a different value in the IPv6 flow label field.

Secured Charging Protocol. This is the implementation of the protocol state machine.

Signature Verification - In this class we will find the primitives for signature verification. Because this is by far the most computationally expensive operation, it has been implemented separately for performance and scheduling issues.

Session Control - This class tracks the sessions between users in which it participates and the amount of information exchanged between end points. It is also the internal repository for the certificates relevant to other nodes.

Accounting - Finally in this module we store the accounting information for that session, should the node be the last forwarding node of a communication. This information includes the number of bytes transmitted, the number of packets and the hash chain and signature route proof for each packet. This information is cataloged by session on a per packet basis.

4 Graphical User Interface

The SCP's Graphical User Interface (GUI) at the client side provides an internal view of the program running in the nodes.

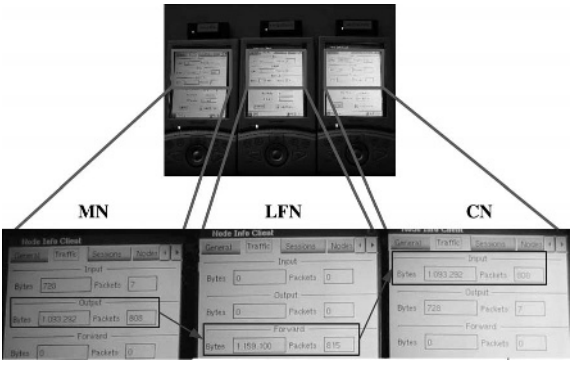


Fig. 2. In depth view of the statistics information in the GUI

The demonstrator setup depicted in Fig. 2 depicts the GUI of three nodes during the forwarding phase. The arrows indicate the flow of the packets between the three nodes while using the SCP implementation. The packets sent from the sending node on the left (MN) are forwarded by the node on the middle (LFN) and finally received by the rightmost node (CN). The LFN is now responsible for transferring this information to the AR in the charging phase. Other information such as certificates and general statistics are also available in the GUI. They are primarily used for debugging purposes.

5 Integration with the AODV6 Routing Protocol

In principle, SCP is independent of the ad hoc routing protocol (should this be proactive, reactive or any other category of ad hoc routing protocol). Nevertheless, certain functionalities have to be implemented into it for SCP to work. They are in particular necessary to calculate the next hop and retrieve the IP addresses. For our demonstration set-up we have chosen to use AODV [14].

Some routing protocol implementations, such as the HUT AODV6, make use of IPQ. Since access to this query process is exclusive and SCP also requires it, we placed special hooks into IPQ to forward packets to the routing module if required. The AODV daemon has been converted into an SCP routing module and both the information functions and the hooks are linked together with the module to empower SCP with the AODV routing protocol.

For proper operation, AODV requires a UDP port. Since some vital routes may not be established during the route discovery process, packets headed for this port cannot be processed by SCP. A rule was added with iptables so that no information passing through this port is filtered by SCP but is instead delivered directly to the AODV routing module.

To allow connections with the fixed network, we follow the same ideas presented by the drafts [15] and [16] on ad hoc Internet connectivity in a mutual direction. We also extended our charging scheme and prototype to nodes that wish to communicate with nodes outside the ad hoc network.

This work is based on a modification of the AODV6 HUT implementation which allows routes to be resolved to outside the ad hoc cloud. The SCP prototype was integrated with this modification in the same way as was described previously.

We make no assumptions on the network topology and allow multi-hop routes, even toward the Access Router.

6 Secure Charging Data Aggregation

In accordance with the last phase of the SCP protocol, accounting information is sent from the LFN and stored at the AR. This data contains the number of bytes transmitted and acknowledged since the last accounting message, the identity of all the nodes that participated in the communication, the disposition of these nodes in the route and a per packet proof of all the values mentioned.

Finally, the data stored at the Access Router contains the individual evidence of every packet transmitted on the network. It is impractical, if not unfeasible, to store all this data in the core network, where the AAA home server lies. Nevertheless, any aggregation scheme must still prevent repudiation. To cope with this problem, an extension to the base protocol has been made by adding new fields to the confirmation reply packet which is sent by the corresponding node, and to the verification request, sent to the Access Router. The former contains a signature that covers the Sequence Number of the confirmation request, the time (with a low resolution), and the number of acknowledged bytes . This sig-

nature is replay attack safe and provides proof that, at that point in time, the correspondent node acknowledged a certain number of bytes. As to the latter, the verification request, it has been completed with the date and sequence number of the confirmation request, to be used with the AAA architecture at any time and to allow the verification of the afore mentioned signature.

In the end, the information necessary at the AAA can be reduced to the total amount of money to be received by and charged to each node. This bulk information must follow the non-repudiation rules set above.

This process provides non-repudiation at the AAA architecture level of the aggregated data.

Data aggregation assumes different lifetimes for the stored data. The information kept at the Access Router is seen as having a lifespan ranging from days up to two or three months while the data in the AAA home server should not expire at all. It is envisioned that there could be points of caching of the accounting information throughout the core network. This would facilitate accounting and charging for foreign nodes at these intermediate levels. A AAA server that wishes to have a more detailed view on the accounting information gathered at the Access Router, can do so explicitly, up until the expiration time.

7 Connecting to AAA

In the real world users are distributed between different Service Providers, which may or may not have contracts between them, and may or may not let them access their services. When integrating an ad hoc network into a real case scenario, we must take this into account. Moreover, the access to one domain may be provided by several AR which share and distribute the functionality of providing authentication and accounting. In the case of SCP, should an Access Router require to verify a hash chain, to confirm the route it must have access to the shared keys. The same way, if it must validate the identity of a node, it must have the node's certificate. It is also feasible that a node registered with one AR and then provides the accounting information to another. This case requires that the second AR can verify the data and the identity as stated above.

A MN is binded to its AAAH. The information on the node's identity is only present at this point. For security and privacy issues, it must never leave the AAAH. Therefore, identity verifications must also be routed to the home domain. We distinguish between the shared keys, which must be shared by the AR of the local domain, and the node's identity and certificate, which must be issued at the home domain, yet verifiable at the local domain. Each AR serves as a translator between the MN and the AAA architecture. The messages exchanged between the MN and the AR in the Registration phase and the Accounting phase are decoded, verified and the appropriate fields are filled accordingly and inserted in the AAA protocol. The message is then sent to the home domain where decisions and final accounting are processed. A top level AAA protocol such as Diameter [17] will be responsible for routing the messages from the AR,

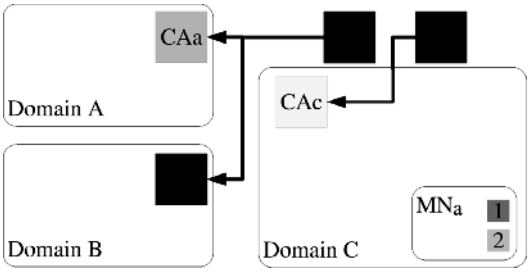


Fig. 3. Hierarchical CA architecture

through the AAAL to the AAAH. This procedure is well defined within such protocols and therefore will not be further discussed in this document.

7.1 Distributed Hierarical Certifying Authorites

To solve the problem of verifying a node's certificate, we propose a two level hierarchical distribution of CA in such a way that any node can verify the authenticity of any other node's certificate with only a subset of the CA keys.

We associate each leaf CA (or several) to a local domain. A node wishing to verify a certificate issued by the local CA must first verify that CA's certificate with it's subset of high level CA keys. Should this verification prove successful, we extend the trust chain to the local CA and can now verify the node's certificate as valid and trustworthy.

The certificates should be obtained from an interaction between the Public Key Infrastructure (PKI) and the AAA architecture. This paper does not further explore this synergy.

8 Testbed, Tests and Measurements

As part of several demonstrators, SCP was tested for scalability, performance and network impact. Although the size of the current demonstrator cannot account for the first, we have tested this SCP implementation to a maximum of 5 intermediate hops and for different traffic classes (audio, video, http).

The testbed is composed of four PDAs and two PCs. One PC is the AR and all the others act as ad hoc nodes. The nodes in this testbed do not have pre-determined roles. We conducted our tests in parallel and independent of the node's position. The position of the nodes does not impact the node to which it communicates because, in order to facilitate our experiments, we have pre-set the routes (the use of AODV is also available). All PDAs have a direct connection to the AR, even though this is not a requirement, and we established the order in such a way that the PC is never an intermediate node. This provides a worst case scenario in what concerns to resources consumed by the low powered nodes.

Note that SCP is only acceptable if the end-to-end delay and the jitter in the forwarding phase still allows for real time traffic. Our tests show asynchronous traffic is not affected by the presence of SCP.

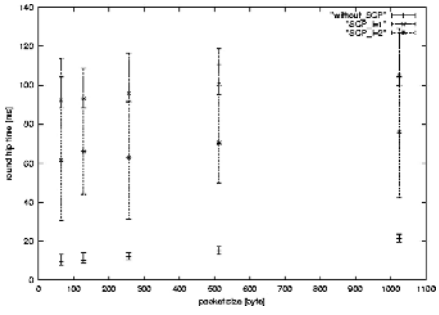


Fig. 4. Round trip times and jitter on the demonstrator in the absence and in the presence of SCP with verification rate (1/l) l=1 or l=2 and varying packet sizes

Our tests have taken the following parameters into account:

End-to-End Delay. The usage of cryptographic primitives, even if optimized for our demonstrator destination platform, severely affects the time each node needs to forward a packet. We have estimated a delay of 17ms per node at a full verification rate[5]. If we decrease the verification rate to 50% (verification of every forwarded packet at each intermediate node), which we have accepted as a still reasonable security threshold, in an average 4 hop network, we observe an end to end delay of around 40ms. This value is well within real time applications such as audio streaming that require a maximum end to end delay of around 50-70ms.

Network Overhead. For the network overhead, in addition to SCP's signaling messages which are out-of-band, we consider: A 16 byte MD5 hash value of the hash chain, the 4 byte seed for the hash chain, the route and the signature of the sender over the packet that is usually around 22 bytes. Typically, on the 4 hop network example above, we should count on 90 byte packet increase by the time the packet reaches the CN. These values depend on the level of security enforced and the number of intermediate nodes.

CPU consumption. Even with the great computational effort of the verifications we notice no humanly perceivable misbehavior on real time applications (such as audio decoding). By observation, we have determined that the implementation, even in low power devices, has no major impact on the device's performance.

The following measurements were done in the testbed described at the beginning of this section using *mgen* [18] and *netperf* [19] for connectionless and connection oriented traffic respectably. In the plot of Fig. 4 we solely focus on UDP (connectionless) traffic.

[5] This is only a prototype implementation. Once the concept is proven to work, one would implement ECC in hardware, resulting in a 10 factor faster execution times of digital signature operations.

9 Further Work and Conclusions

We believe that this work is a good starting point to combine ad hoc with fixed networks in what concerns charging. Nevertheless, it needs further exploration. Once a node reaches an administrative domain under the control of several ISPs, it must negotiate prices and capabilities with the nodes currently being served by that Access Router. A node may not wish to conform to the pricing scheme first presented in that network and therefore this is a challenging issue and certainly worth analyzing.

It is also our belief that other work in this area can be applied to deliver a gaming theory based mechanism that provides a probabilistic model for the price negotiation.

We also hope to realize a hardware implementation of the cryptography functions used in order to speed up our test results.

This paper proposes SCP as the reunion point of wired and ad hoc networks' charging and accounting, and outlines the future work in the interaction between the both.

We also show, both by prototype implementation and evaluation, that SCP can indeed be used in real life scenarios to provide a secure, incentive based approach to solve the selfish node problem.

References

1. Buchegger, S., Boudec, J.Y.L.: (Performance analysis of the CONFIDANT protocol (cooperation of nodes: Fairness in dynamic ad-hoc NeTworks)) 226–236
2. Marti, S., Giuli, T., Lai, K., Baker, M.: Mitigating routing misbehaviour in mobile ad hoc networks. In: 6th International Conference on Mobile Computing and Networking, ACM MobiCom 2000 Conference (2000) 255–265
3. Buttyan, L., Hubaux, J.P.: Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks. Technical Report DSC/2001/001, Swiss Federal Institute of Technology, Lausanne (2001)
4. Zhong, S., Yang, Y., Chen, J.: Sprite: A simple, cheat-proof, credit-based system for mobile ad hoc networks (2002)
5. Lamparter, B., Paul, K., Westhoff, D.: Charging support for ad hoc stub networks. Elsevier Journal of Computer Communications **Elsevier Science** (2003) Special Issue on Internet Pricing and Charging: Algorithms, Technology and Applications.
6. de Laat, C., Gross, G., Gommans, L., Vollbrecht, J., Spence, D. (Technical report)
7. Johnson, D.B., Maltz, D.A.: Dynamic source routing in ad hoc wireless networks. In Imielinski, Korth, eds.: Mobile Computing. Volume 353 of The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers (1996) **http://athos.rutgers.edu/ imielins/book.html.**
8. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems (reprint). Communications of the ACM **26** (1983) 96–99
9. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation **48** (1987) 203-209
10. Riedel, I.: Security in ad hoc networks: Protocols and elliptice curve cryptography on an embedded platform. Diploma thesis at NEC, University of Bochum (2003)

11. Lamparter, B., Paar, C., Weimerskirch, A., Westhoff, D.: On digital signatures in ad hoc networks. IEEE Journal on Selected Areas in Communications **'Wireless Ad Hoc Networks'** (2004) submited.
12. de Laat, C., Gross, G., Gommans, L., Vollbrecht, J., Spence, D.: Generic AAA architecture. Internet Request for Comment RFC 2903, Internet Engineering Task Force (2000)
13. Hickman, K.E.B.: The SSL protocol. RFC draft, Netscape Communications Corp. (1994) Version 1.0.
14. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc on-demand distance vector (AODV) routing for ip version 6. Internet-Draft draft-perkins-aodv6-01.txt (work in progress) (2001)
15. Jelger, C., Noel, T., Frey, A.: draft-jelger-manet-gateway-autoconf-v6-01.txt. Internet-Draft draft-jelger-manet-gateway-autoconf-v6-01.txt (work in progress) (2003)
16. Cha, H.W., Park, J.S., Kim, H.J.: Support of internet connectivity for aodv. Internet-Draft draft-cha-manet-AODV-internet-00.txt (work in progress) (2004)
17. Calhoun, P., Loughney, J., Guttman, E., Zorn, G., Arkko, J.: Diameter base protocol. Internet Request for Comment RFC 2903, Internet Engineering Task Force (2003)
18. : Mgen - the multi-generator toolset. http://manimac.itd.nrl.navy.mil/MGEN/ (2004)
19. : The public netperf homepage. http://www.netperf.org/netperf/NetperfPage.html (2004)

Blind Spontaneous Anonymous Group Signatures for Ad Hoc Groups

Tony K. Chan, Karyin Fung, Joseph K. Liu, and Victor K. Wei

Department of Information Engineering,
The Chinese University of Hong Kong,
Shatin, Hong Kong
{klchan3, kyfung2, ksliu9, kwwei}@ie.cuhk.edu.hk

Abstract. Spontaneous anonymous group (SAG) cryptography is a fundamental alternative to achieve thresholding without group secret or setup. It has gained wide interests in applications to ad hoc groups. We present a general construction of blind SAG 1-out-of-n and t-out-of-n signature schemes from essentially any major blind signature. In the case when our scheme is built from blind Schnorr (resp. Okamoto-Schnorr) signature, the parallel one-more unforgeability is reduced to Schnorr's ROS Problem in the random oracle model plus the generic group model. In the process of our derivations, we obtain a generalization of Schnorr's result [17] from single public key to multiple public keys.

1 Introduction

The popular goals of group cryptography or threshold cryptography are usually:

> Any t members of a group of n members can jointly demonstrate a knowledge concerning the group that no combination of $t-1$ or fewer members can demonstrate.

There are threshold signature schemes that require no less than t members to jointly generate. There are threshold decryption schemes (cryptosystems) that require no less than t members to jointly decrypt. Besides unforgeability, other properties such as robustness, adaptive adversary models, blind signatures, culpability or exculpability, witness hiding, witness indistinguishability (anonymity) are also significant research topics.

Since its inception, group cryptography and threshold cryptography [11] have traditionally been achieved through the secret sharing technique [19, 4]. Also since its inception [9], anonymous (insider-indistinguishable) group cryptography has traditionally been achieved by the technique of blind signatures or other forms of transfer proof-of-knowledge (TPoK). For further details, see [9, 7]

Recently a fundamental alternative has gained wide interests. In the *spontaneity* paradigm to group cryptography, there is no group secret. There is also no setup. Any single entity can arbitrarily and spontaneously conscript $n-1$

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 82–94, 2005.

diversion members to form a group, and complete a signature without the participation, or even knowledge, of the diversion members. The resulting signature can be proven to be from one of the n group members. Yet the actual signer remains anonymous (signer-indistinguishable), with irrevocable, exculpable anonymity. The only requirement is that each group member has a published public key, for the purpose of signature verification. There are also t-out-of-n threshold versions where t entities joint to spontaneously conscript $n - t$ diversion members.

Compared with traditional threshold signature schemes, spontaneous group signatures achieved the definition goal quoted at the beginning of this section. Yet there is no group secret. There is no group setup which requires the participation of non-insider members.

Due to its flexibility and the ease (or lack) of setup, SAG cryptography has been deemed perfectly suitable for applications in ad hoc groups [16, 6, 5].

Compared with traditional privacy (anonymity) protection schemes, spontaneous group cryptography is naturally anonymous. It achieves anonymity without using blinding techniques. Furthermore, the anonymity in spontaneous anonymous group (SAG) cryptography is very strong: in its basic version, the anonymity is unconditional (information-theoretic), irrevocable, and exculpable. The last property means that even if all communication sessions and all secret keys are subpoenaed, the anonymity cannot be revealed. Variants of SAG cryptography achieved different tradeoffs in anonymity based on candidate hard problems and optional revocability and optional culpability.

Our Contributions: In this paper, we present the first blind [8] spontaneous anonymous group (SAG) signature schemes. Based on essentially any major blind signature, we construct ring-type [16, 1] 1-out-of-n blind SAG signatures and CDS-type [10] t-out-of-n blind SAG signatures. The blindness of our SAG blind signature depends on that of its underlying component blind signature. The parallel one-more unforgeability of our SAG signature, when the underlying component is the Schnorr (resp. Okamoto-Schnorr) blind signature, is reduced to Schnorr's ROS Problem [17], in the random oracle model [3] plus the generic group model [14]. In the process, we extend Schnorr's result [17] on single-key parallel one-more unforgeability (p1m-uf) to obtain a reduction of multiple-key parallel unforgeability (mk-p1m-uf) of Schnorr (resp. Okamoto-Schnorr) blind signature to the ROS Problem, in the random oracle model plus the generic group model.

Paper Organization: Background materials in Section 2. Security models and definition of security notions in Section 3. Constructions of blind SAG signatures in Section 4. Security analyses in Section 5. Conclusions in Section 6.

2 Background Materials

We review background results needed subsequently.

2.1 General Background

A **PoK (Proof-of-Knowledge)** is a three-move interactive protocol consisting of (Prover, Verifier). Common input consists of a public key, PK. Prover has the additional input SK. The three moves are $\mathcal{K}$=($\mathcal{T}$,$\mathcal{C}$,$\mathcal{S}$)=(*commit*, *challenge*, *response*). *Completness* means, with all sides honest, results are as they should be. *Soundness* means two random challenge-response pair to the same commitment result in witness extraction. *Special Soundness* means: any two challenge-response pair with the same commitment result in witness extraction.

A **blind signature** consists of the tuple (BlindSigner, Warden, Verifier) where the three components form an interactive protocol as follows:

1. Common input to all three parties: PK. Additional input to BlindSigner: SK.
2. BlindSigner sends t' (commitment) to Warden.
3. Warden sends t to Verifier.
4. Verifier sends message m to Warden.
5. Warden sends c' to BlindSigner.
6. BlindSigner sends s' to Warden.
7. Warden sends s to Verifier.
8. Verifier confirms that (t, s) is a valid signature on m w.r.t. PK.

Typically, Warden is instantiated as a tuple of mappings (f_t, f_c, f_s) and that in various moves do the following:

1. Warden randomly generates Δ_c and Δ_s, computes $t := f_t(PK, t', \Delta_c, \Delta_s)$, and sends t to Verifier.
2. Verifier sends m to Warden.
3. Warden computes $c := H(t, m)$ $c' := f_c(PK, t', \Delta_c, \Delta_s, c)$ and sends c' to BlindSigner.
4. BlindSigner computes s' and sends it to Warden.
5. Warden computes $s = f_s(PK, t', \Delta_c, \Delta_s, t, c, c', s')$ and sends it to Verifier.

If (t', c', s') is a valid PoK, then so is (t, c, s). Some examples below.
Schnorr blind signature[18]: Relation $R = \{(y = g^x, x) | x \in \{1, \cdots, q\}\}$ with $(\mathcal{T}, \mathcal{C}, \mathcal{S} : \mathcal{T} = g^{\mathcal{S}} y^{\mathcal{C}})$ and $\mathcal{T} = \mathcal{T}' g^{\Delta_S} y^{\Delta_C}$,
Okamoto-Schnorr blind signature [15]: Relation $R = \{(g^{x_1} h^{x_2}, (x_1, x_2)) | x_1, x_2 \in \{1, \cdots, q\}\}$. with $(\mathcal{T}, \mathcal{C}, \mathcal{S}) = (\ g^{r_1} h^{r_2},\ c,\ (s_1, s_2) = (x_1 + r_1 c, x_2 + r_2 c)\ )$

Blindness: The signer of a blind signature has no information about the message during and after a blind signature/TPoK protocol. Given any message-signature pair, the signer cannot find out when and for whom it was signed.

2.2 Schnorr's ROS Assumption

Schnorr [17] presented a then-new algorithm to compute the parallel one-more forgery of Schnorr (resp. Okamoto-Schnorr) blind signatures. He showed the equivalence of the parallel one-more unforgery of those two blind signatures and the ROS Problem, in the random oracle model plus the generic group model.

His techique also applied to many other blind signatures. In this paper, we will use the following form of Schnorr's ROS Problem:

The ROS Problem: Given $1 \leq q_B \leq q_H$, typically $q_B << q_H$, and all computations are in Z_q. Compute a $q_H \times q_B$ matrix $\mathbf{A}$, such that the probability of computing the following problem is non-negligible:

> Given random $\hat{c} = [\hat{c}_1, \cdots, \hat{c}_{q_H}]$, compute $J \subset \{1, \cdots, q_H\}$ with $|J| = q_B+1$, $j_0 \in J$, $\{\alpha_j : j \in J\}$ with $\alpha_{j_0} \neq 0$, and β such that $\sum_{j \in J} \alpha_j [\mathbf{A}_j, \hat{c}_j] = \beta$ and $\{\mathbf{A}_j : j \in J \setminus \{j_0\}\}$ are linearly independent.

Note $\mathbf{A}_j$ denote the j-th row vectors of $\mathbf{A}$, and $[\mathbf{A}_j, \hat{c}_j]$ denotes the lengthened vector by one more entry $\hat{c}_j$.

2.3 Background About SAG Signature

First, the definition of SAG signatures.

Definition 1. *Let $L = \{PK_1, \cdots, PK_n\}$ be a list of n public keys, θ be an integer, $1 \leq \theta \leq n$, m be a message, and $\sigma = (t_1, \cdots, t_n, c_1, \cdots, c_n, s_1, \cdots, s_n)$ be a tuple. Let H, H_1, $\cdots$, H_n be full-domain collision-free secure hashing functions. The tuple $(L, n, \theta, m, \sigma)$ is a* ring-type SAG signature [16, 1] *if the following all hold:*

1. $\theta = 1$
2. *For each i, $1 \leq i \leq n$, we have $c_i = H_i(L, n, m, t_{i-1})$ and (t_i, c_i, s_i) is a valid PoK conversation w.r.t. PK_i. (t_0 is interpreted as t_n.)*

The tuple is a CDS1-type SAG signature [10] *if the following all hold*

1. *Each tuple (t_i, c_i, s_i) is a valid PoK conversation w.r.t. PK_i, $1 \leq i \leq n$.*
2. *The polynomial f interpolated from $f(i) = c_i$, $0 \leq i \leq n$, has degree at most $n - \theta$, where $c_0 = H(L, n, \theta, m, t_1, \cdots, t_n)$.*

The tuple is a CDS2-type SAG signature *if the following all hold*

1. *Each tuple (t_i, c_i, s_i) is a valid PoK conversation w.r.t. PK_i, $1 \leq i \leq n$.*
2. *For each $1 \leq j \leq \theta$, $\sum_{1 \leq i \leq n} i^j c_i = H_j(L, n, \theta, m, t_1, \cdots, t_n)$.*

Remark: To conserve bandwidth, the representation of an SAG signature can be shortened. For example, if $(t_1, \cdots, t_n)$ can be efficiently constructed from $(c_1, \cdots, c_n, s_0, \cdots, s_n)$, then it can be omitted from the representation. In ring-type SAG signatures, $(c_2, \cdots, c_n)$ can be further omitted since they can be constructed from $(c_1, s_1, \cdots, s_n)$.

A Construction of Ring-Type SAG Signatures [16]: Given a list of public keys $L = \{PK_1, \cdots, PK_n\}$, a message m, a suitable hash function H, a secret key SK_π corresponding to PK_π, a ring-type SAG signature can be constructed as follows:

1. Randomly generate a commitment $\mathcal{T}_\pi$.
2. For each $i = \pi + 1, \cdots, n, 1, \cdots, \pi - 1$, compute $\mathcal{C}_i = H(L, m, \mathcal{T}_{i-1})$ and then simualate a PoK conversation $(\mathcal{T}_i, \mathcal{C}_i, \mathcal{S}_i)$ w.r.t. PK_i.
3. For $i = \pi$, compute $\mathcal{C}_i = H(L, m, \mathcal{T}_{i-1})$ and then compute a PoK conversation $(\mathcal{T}_i, \mathcal{C}_i, \mathcal{S}_i)$ using the secret key SK_i.
4. Output SAG signature $(L, n, \theta = 1, m, \sigma)$ where $\sigma = (\mathcal{C}_1, \mathcal{S}_1, \cdots, \mathcal{S}_n)$ (thus achieving bandwidth conservation).

A Construction of CDS1-Type (Resp. CDS2-Type) SAG Signature [10]: Given list of public keys $L = \{PK_1, \cdots, PK_n\}$, message m, suitable hash function H. Let $I \subset \{1, \cdots, n\}$, $|I| = t$. Given secret keys $\{SK_\pi : \pi \in I\}$, generate SAG signature as follows:

1. For each $i \notin I$, simulate a PoK conversation $(\mathcal{T}_i, \mathcal{C}_i, \mathcal{S}_i)$.
2. For each $\pi \in I$, randomly pick $\mathcal{T}_\pi$.
3. Compute $\mathcal{C}_0 = H(L, n, \theta, m, \mathcal{T}_1, \cdots, \mathcal{T}_n)$, and solve for $\mathcal{C}_\pi$'s, $\pi \in I$, such that the polynomial f interpolated from $f(i) = \mathcal{C}_i$, $0 \le i \le n$, has degree no more than $n - \theta$. (resp. for CDS2-type, solve for $\mathcal{C}_\pi$'s, $\pi \in I$, such that $\sum_{1 \le i \le n} i^j \mathcal{C}_i = H_j(L, n, \theta, m, t_1, \cdots, t_n)$, $1 \le j \le \theta$.)
4. For each $\pi \in I$, compute a PoK conversation $(\mathcal{T}_\pi, \mathcal{C}_\pi, \mathcal{S}_\pi)$ using SK_π.
5. Output an SAG signature $(L, n, \theta, m, \sigma)$ where $\sigma = (f, \mathcal{S}_1, \cdots, \mathcal{S}_n)$ (achieving bandwidth conservation).

Properties of SAG Signatures: The SAG signature has statistical ZK (zero-knowledge) about its actual signers. Therefore, the signer anonymity is unconditional and exculpable. Furthermore, the SAG signature is a group signature which requires essentially no setup, especially in terms of group key setup or secret sharing of the group key. Any one user can conscript the public keys of another $n-1$ users to form an SAG signature without the participation or even knowledge of the conscripted diversion signers. Such properties make SAG signatures useful in diverse applications including whistle blowing[16], e-voting [13], and ad hoc group cryptography [6].

2.4 Generic Group Model (GGM)

We will use the generic group model of [14, 20, 17]. Some highlights below.

Only a restricted set of operations are allowed. They include random generation of integers and group elements, group computations, exponentiations, equality tests. There are only two data types: *group elements* and *non-group data*.

It is assumed the the discrete logarithm problem is uncomputable in the GGM[14].

We restrict ourselves to a polynomial number of steps. Therefore, there are only a polynomial number of unassociated group elements base g, public keys $y_1, \cdots, y_n$, commitments $t_1, \cdots, t_{q_B}$, randomly generated group elements $u_1, \cdots, u_{q_G}$. The computation transcript at each step τ consists of

$$f_\tau = g^{a_{\tau,-1}} \prod_i y_i^{a_{\tau,i}} \prod_{i'} t_{i'}^{b_{\tau,i'}} \prod_{i''} u_{i''}^{c_{\tau,i''}} \tag{1}$$

Each computation can only depend on parameters in existence before that step, resulting in zero exponent for parameters that come into existence after that step.

Probabilities of hash collisions, discrete logarithm collisions, integer computation collisions are all assumed negligible. (Except those resulting from BlindSign Oracle queries.)

3 Blind SAG Signature and Security Model

3.1 The Real World

A blind SAG signature scheme is a tuple (KeyGen, SAGWarden, $\mathsf{BlindSigner}_{PK_1}$, $\cdots$, $\mathsf{BlindSigner}_{PK_n}$, SAGVerifier) where

KeyGen: Upon input a security parameter 1^λ and generates public-private key pair (PK, SK).

SAGVerifier: Upon input a tuple $(L, n, \theta, m, \sigma)$, outputs ACCEPT or REJECT.

$\mathsf{BlindSigner}_{PK_1}$, $\cdots$, $\mathsf{BlindSigner}_{PK_n}$ are (ordinary) $\mathsf{BlindSigner}_{PK}$ protocols defined in the last Section.

SAGWarden: Upon input L', n', θ', m', it picks $I \subset \{1, \cdots, n'\}$, $|I| = \theta'$, and by invoking $\mathsf{BlindSigner}_{PK_i}$, $i \in I$, produces an SAG signature $(L', n', \theta', m', \sigma')$.

3.2 The Ideal World

1. $\mathcal{SO}$ (Signing Oracle): Upon input a public key PK' and any message m', it outputs a valid signature σ'.
2. SAGSign (SAG Signing Oracle): Upon input public key list L', length n', threshold θ', message m', it outputs a valid SAG signature $(L', n', \theta', m, \sigma')$.
3. BlindSign (Blind Signing Oracle): Upon query, it conducts a 4-move interactive protocol with the querier $\mathcal{Q}$ as follows:
   (a) Move-0: $\mathcal{Q}$ sends PK'.
   (b) Move-1: BlindSign sends a *commitment* t to $\mathcal{Q}$.
   (c) Move-2: $\mathcal{Q}$ sends a challenge c to BlindSign.
   (d) Move-3: BlindSign returns s such that (t, s) forms a valid PoK w.r.t. PK'.
4. Random Oracle: Upon receiving a query, it outputs a random number. All query-reply pairs are kept in record and no same reply for different queries.

3.3 Definitions of Security Notions

Definition 2. *(Completeness) If all parties are honest in following the protocols, then the output of the interactions with various oracles will produce valid signatures.*

Game UF

1. (Setup) Upon input a security parameter 1^λ, generate parameters n, θ, and invoke KeyGen n times to generate key pairs (SK_i, PK_i), $1 \leq i \leq n$. The above, except the secret keys, are published.

2. A forger, $\mathcal{F}$ makes q_B (resp. q_S, q_H, q_A) queries to the BlindSigner (resp. $\mathcal{SO}$, random oracle, SAGSign).
3. $\mathcal{F}$ delivers $> q_B/\theta$ valid SAG signatures $(L_i, n_i, \theta, m_i, \sigma_i)$, $1 \leq i \leq q_B + 1$, none of which coincides with any SAGSign query output.

Remark: For simplicity, we require $\mathcal{F}$ to deliver SAG signatures with the same threshold θ, and each public key used in SAG signatures delivered by $\mathcal{F}$ must have been generated in the Setup Phase of Game UF. In this paper, we restrict ourselves to at most a polynomially many queries in terms of the security parameter.

Definition 3. *(Parallel One-more Unforgeability (p1m-uf)) A blind SAG signature scheme is parallel one-more unforgeable (against adaptive chosen-message, chosen-public-key active attackers) if no PPT adversary can successfully complete Game UF with non-negligible probability.*

Remark: Specializing to $n = \theta = 1$, the above definition is defining p1m-uf of classic blind signatures.

Definition 4. *(Blindness) A blind SAG signature scheme has* blindness *if the probability distribution of the signature produced by Warden is indistinguishable from the probability distribution of the signatures produced by Warden conditioned on the blindsign conversation that produced it.*

Roughly speaking,

$$\Pr\left\{\begin{array}{l|l}\text{SAG signature} & \text{BlindSign Oracle} \\ \text{by Warden} & \text{conversation}\end{array}\right\} = \Pr\left\{\begin{array}{l}\text{SAG signature} \\ \text{by Warden}\end{array}\right\}$$

4 Constructing Blind SAG Signatures

We present the constructions of our blind SAG signatures.

4.1 Blind SAG Signature: CDS-Type [10]

Given a list of n public keys, $L = \{PK_1, \cdots, PK_n\}$, message m, threshold θ, and θ accesses to blind signer w.r.t. public keys from L, the following protocol generates a CDS1-type SAG signature (resp. CDS1-type, CDS2-type) $(L, n, \theta, m, \sigma)$:

1. Select $I \subset \{1, \cdots, n\}$, $|I| = \theta$.
2. For each $i \in \{1, \cdots, n\} \setminus I$, generate PoK triple (t_i, c_i, s_i) w.r.t. PK_i.
3. In θ sessions of the TPoK protocol, one for each $i \in I$, act as Warden equipped with $\mathsf{BlindSigner}_{PK_i}$ w.r.t. PK_i, as follows:
   (a) Obtain commitment t_i' from $\mathsf{BlindSigner}_{PK_i}$, for each $i \in I$.
   (b) For each $i \in I$, compute $\Delta_{s,i}$, $\Delta_{c,i}$, and $t_i = f_t(PK_i, t_i', \Delta_{c,i}, \Delta_{s,i})$.
   (c) Compute $c_0 = H(L, n, \theta, m, t_1, \cdots, t_n)$.

(d) Compute c_i for all $i \in I$ such that the polynomial f interpolated from $f(i) = c_i$, $0 \le i \le n$, has degree at most $n - \theta$. (resp. for CDS2-type, solve for $\mathcal{C}_i$'s, $i \in I$, such that $\sum_{1 \le i \le n} i^j \mathcal{C}_i = H_j(L, n, \theta, m, t_1, \cdots, t_n)$, $1 \le j \le \theta$.)
(e) For each $i \in I$, compute $c'_i = f_c(PK_i, t'_i, \Delta_{c,i}, \Delta_{s,i}, c_i)$, and send c'_i to $\mathsf{BlindSigner}_{PK_i}$.
(f) For each $i \in I$, receive s'_i from $\mathsf{BlindSigner}$ i, and compute $s_i = f_s(PK_i, t'_i, \Delta_{c,i}, \Delta_{s,i}, c_i, s'_i)$.

4. Output $\sigma = (f, s_1, \cdots, s_n)$.

The blind signature for individual index i is referred to as the *underlying blind signature scheme* of the blind SAG signature scheme.

4.2 Blind SAG Signature: Ring-Type [16, 1]

Given a list of n public keys $L = \{PK_1, \cdots, PK_n\}$, message m and accesses once to $\mathsf{BlindSigner}_{PK_i}$ w.r.t. $PK_i \in L$, the following protocol generates a ring-type SAG signature (L, n, m, σ):

1. Select $\pi \in \{1, \cdots, n\}$.
2. Interact as Warden with $\mathsf{BlindSigner}_{PK_\pi}$ to obtain a commitment t'_π, and compute $t_\pi = f_t(PK_\pi, t'_\pi, \Delta_{c,\pi}, \Delta_{s,\pi})$ with randomly generated $\Delta_{c,\pi}$ and $\Delta_{s,\pi}$).
3. Sequentially for each $i = \pi+1, \cdots, n, 1, \pi-1$, compute $c_i = H(L, m, n, t_{i-1})$, and then simulate a PoK triple (t_i, c_i, s_i) w.r.t. PK_i.
4. Finish the interaction with $\mathsf{BlindSigner}_{PK_\pi}$ by
   (a) Compute and send $c'_\pi = f_c(PK_\pi, t'_\pi, \Delta_{c,\pi}, \Delta_{s,\pi}, c_\pi)$.
   (b) Receive s'_π nd compute $s_\pi = f_s(PK_\pi, t'_\pi, \Delta_{c,\pi}, \Delta_{s,\pi}, c_\pi, s'_\pi)$.
5. Output $\sigma = (c_1, \cdots, c_n, s_1, \cdots, s_n)$.

5 Security Analysis

We prove the completeness, the blindness, and the parallel one-more unforgeability of our blind SAG signature schemes. In the process, we also prove an extension of Schnorr's [17] ROS result from single public key to multiple public keys.

5.1 Multi-Key Parallel One-More Unforgeability of Blind Signature

The following results are well-known.

Theorem 1. *[17] The parallel one-more unforgeability (p1m-uf) of Schnorr (resp. Okamoto-Schnorr) blind signature is equivalent to the ROS Problem in the random oracle model plus the generic group model.*

In Schnorr's security model [17], all queries to blindsign are w.r.t. a single public key PK. We generalize it to *multiple-key parallel one-more unforgeability* (mk-p1m-uf) by allowing the Adversary to query blindsign with K different (PK_i) , $1 \leq i \leq n$, a total of q_B times in order to produce a total of $q_B + 1$ signatures each of which is verifiable against some members of the set of public keys $\{PK_1, \cdots, PK_K\}$. We will need this result.

Theorem 2. *The multiple-key parallel one-more unforgeability (mk-p1m-uf) of Schnorr (resp. Okamoto-Schnorr) blind signature is equivalent to the ROS Problem in the random oracle model plus the generic group model.*

Proof in the Appendix.

5.2 Security of Our Blind SAG Signatures

Theorem 3. (Completeness*) Our blind SAG signature has completeness.*

Proof: Trivial.

Theorem 4. (Blindness) *Assume L, n, θ are fixed. Our ring-type (resp. CDS1-type, CDS2-type) blind SAG signature has blindness provided the underlying blind signature also has it.*

Proof Sketch: Denote the SAGBlindSign session communication transcripts by $\mathcal{K}_i = (\mathcal{T}_i, \mathcal{C}_i, \mathcal{S}_i)$, $1 \leq i \leq \theta$, and the SAG signature in question by $(L, n, \theta, m, \sigma)$ where σ=$(t_1, \cdots, t_n, c_1, \cdots, c_n, s_1, \cdots, s_n)$. By the ZK of the underlying blind signatures, (t_i, c_i, s_i) is ZK w.r.t. $(\mathcal{T}_i, \mathcal{C}_i, \mathcal{S}_i)$. Furthermore, (non-blind) SAG signatures are ZK about which secret key actually generated it. Therefore σ is ZK. □

Theorem 5. *(*Unforgeability*) Our ring-type (resp. CDS1-type with $\theta = 1$, CDS2-type with $\theta \geq 1$) SAG blind signature based on Schnorr or Okamoto-Schnorr blind signature is parallel one-more unforgeable (p1m-uf) provided Schnorr's ROS Problem is hard, in the generic group model (GGM) plus the random oracle model (ROM).*

Proof in the Appendix.
Remark: The reduction in Theorem 1 is actually to the ROS Problem or the Discrete Logarithm Problem (DLP). The reduction in Theorem 2 (resp. Theorem 5) is actually to the ROS Problem or the *one-more discrete log (1mDL)* problem. (The 1mDL Problem: compute all discrete logarithms $\log_g y_i$ for $1 \leq y \leq q_{DL}+1$, given g and y_1, $\cdots$, $y_{q_{DL}+1}$ and a total of q_{DL} queries to a Corruption Oracle, which returns the discrete logarithms of qualified query values.) In the GGM, it can be deducted that the probability of computing dicrete log collisions, which include DLP and 1mDL, is negligible for PPT algorithms.

6 Concluding Remarks

We have constructed blind SAG signatures, both ring-type and CDS-type. We have reduced their parallel one-more unforgeability against adaptive chosen-plaintext, adaptive chosen-public-key attackers, to the parallel one-more unforgeability of the component blind signature, and a candidate hard problem, in two cases: where the underlying blind signature is the Schnorr (resp. Okamoto-Schnorr) blind signature.

The security and privacy (anonymity) of the blind SAG signature based on Schnorr blind signature is an interesting topic. The result of Schnorr[17] reduced the security of the Schnorr blind signature to the ROS (Randomized Oversampled Solvable system) Assumption. Recently, Wagner [21] gave a sub-exponential time algorithm to solve the ROS problem. If the array entries are all elements of a binary field, then the ROS Problem can be solved in polynomial time by a method from [2] or [12].

It will be interesting to generalize Schnorr ROS reduction to the Schnorr-based blind SAG signature. Since Schnorr identification scheme does not have zero-knowledge, it will also be interesting to explore the exact zero-knowledge properties of that blind SAG signature.

Acknowledgements. Helpful discussions with Duncan S. Wong are acknowledged.

References

1. M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In *Proc. ASIACRYPT 2002*, pages 415–432. Springer-Verlag, 2002. Lecture Notes in Computer Science No. 2501.
2. M. Bellare and D. Micciancio. a new paradigm for collision-free hasing: incrementality at reduced cost. In *Proc. EUROCRYPT 97*. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1233.
3. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.
4. G. R. Blakley. Safeguarding cryptographic keys. In *Proc. AFIPS National Computer Conference*, volume 48, pages 313–317, 1979.
5. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proc. EUROCRYPT 2003*, pages 416–432. Springer-Verlag, 2003. Lecture Notes in Computer Science No. 2656.
6. E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *Proc. CRYPTO 2002*, pages 465–480. Springer-Verlag, 2002. Lecture Notes in Computer Science No. 2442.
7. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proc. EUROCRYPT 2001*, pages 93–118. Springer-Verlag, 2001. Lecture Notes in Computer Science No. 1294.
8. D. Chaum. Blind signatures for untraceable payments. In *Proc. CRYPTO 82*, pages 199–203. Plenum Press, 1982.

9. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *CACM*, 29(10):1030–1044, 1985.
10. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proc. CRYPTO 94*, pages 174–187. Springer-Verlag, 1994. Lecture Notes in Computer Science No. 839.
11. Y. Desmedt. Some recent research aspects of threshold cryptography. In *Proc. First International Workshop on Information Security, ISW 97*, pages 158–173. Springer-Verlag, 1997. Lecture Notes in Computer Science No 1196.
12. J. K. Liu, V. K. Wei, and D. S. Wong. Cryptanalyzing Bresson, et al.'s spontaneous anonymous group threshold signature for ad hoc groups and patching via updating Cramer, et al.'s threshold proof-of-knowledge. *eprint*, 2004(042), 2004.
13. J. K. Liu, V. K. Wei, and D. S. Wong. Linkable and culpable ring signatures. *eprint*, 2004(027), 2004.
14. V.I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes 55*, pages 165–172, 1994.
15. T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Proc. CRYPTO 92*, pages 31–53. Springer-Verlag, 1993. Lecture Notes in Computer Science No. 740.
16. R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Proc. ASIACRYPT 2001*, pages 552–565. Springer-Verlag, 2001. Lecture Notes in Computer Science No. 2248.
17. C. P. Schnorr. Security of blind discrete log signatures against interactive attacks. In *ICICS*. Springer, 2001. Lecture Notes in Computer Science No. 2229.
18. C.P. Schnorr. Efficient identication and signatures for smart cards. In *Proc. CRYPTO 89*, pages 239–252. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 435.
19. A. Shamir. How to share a secret. In *Communications of the ACM*, volume 22(2), pages 612–613. ACM Press, 1979.
20. V. Shoup. Lower bounds for discrete logarithms and related problems. In *Proc. EUROCRYPT 97*, pages 256–266. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1233.
21. D. Wagner. A generalized birthday problem. In *Proc. CRYPTO 2002*, pages 288–303. Springer-Verlag, 2002. Lecture Notes in Computer Science No. 2442.

A Proof Sketch of Theorem 2

We mimick Schnorr's [17] proof. The generic mk-p1m attacker is as follows:

1. Obtain commitments: $t_{k,i}$, $1 \le k \le K$, $1 \le i \le q_{B,k}$; where $\sum_k q_{B,k} = q_B$.
2. Compute and then send challenges $c_{k,i}$, $1 \le k \le K$, $1 \le i \le q_{B,k}$.
3. Receive responses $s_{k,i}$. Output $q_B + 1$ signatures $(\hat{t}_{\ell,j}, \hat{s}_{\ell,j})$ on messages $\hat{m}_{\ell,j}$ where $\hat{t}_{\ell,j} = g^{\hat{s}_{\ell,j}} y_\ell^{\hat{c}_{\ell,j}}$, $\hat{c}_{\ell,j} = H(\hat{t}_{\ell,j}, \hat{m}_{\ell,j})$; and $1 \le \ell \le K$, $1 \le j \le \hat{q}_{B,\ell}$, $\sum_\ell \hat{q}_{B,\ell} = q_B + 1$

The oracle conversations can be arbitrarily interleaved. The hash query $\hat{c}_{\ell,j} = H(\hat{t}_{\ell,j}, \hat{m}_{\ell,j})$ must have been made.

Let $f_{\tau(\ell,j)} = \hat{c}_{\ell,j}$, for some index mapping τ.

In Eq(1), we can treat $u_i = y_{q_B+i}$. The u_i's can be used as public keys in querying the Signing Oracle. If they are not used as such, then set $q_{B,q_B+i} = 0$.

They cannot be used as public keys in the delivered signatures, if the conditions so require. Therefore, we can omit the u's w.l.o.g. Expanding the subscript of the t's from one to two according to the current convention, we obtain

$$\begin{aligned} f_{\tau(\ell,j)} &= g^{\hat{s}_{\ell,j}} y_\ell^{\hat{c}_{\ell,j}} \\ &= g^{a_{\tau(\ell,j),-1}} \prod_{k'} y_{k'}^{a_{\tau(\ell,j),k'}} \prod_k \prod_i t_{k,i}^{b_{\tau(\ell,j),k,i}} \\ &= g^{a_{\tau(\ell,j),-1}} \prod_{k'} y_{k'}^{a_{\tau(\ell,j),k'}} \prod_k \prod_i (g^{s_{k,i}} y_k^{c_{k,i}})^{b_{\tau(\ell,j),k,i}} \end{aligned}$$

and

$$1 = g^{\Delta_{s,\ell,j}} \prod_{k'} y_{k'}^{\Delta_{c,\ell,j,k'}}, \text{ for each } \ell, j,$$

where

$$\begin{aligned} \Delta_{s,\ell,j} &= -\hat{s}_{\ell,j} + a_{\tau(\ell,j),-1} + \sum_k \sum_i s_{k,i} b_{\tau(\ell,j),k,i} \\ \Delta_{c,\ell,j,k'} &= -\hat{c}_{\ell,j} \delta(\ell, k') + a_{\tau(\ell,j),k'} + \sum_i c_{k',i} b_{\tau(\ell,j),k',i}. \end{aligned}$$

where the Kronecker delta $\delta(u, v) = 1$ when $u = v$ and equals 0 otherwise. Note that the last two Δ-coefficients are computable by the generic adversary, but not by the Simulator. Therefore rewinding will not enable the Simulator to extract any secret key.

Case (1): $\Delta_{s,\ell,j} = \Delta_{c,\ell,j,k'} = 0$ for all ℓ, j, k'. Then the generic adversary has solved the ROS Problem:

$$\hat{c}_{\ell,j} = a_{\tau(\ell,j),\ell} + \sum_i c_{\ell,i} b_{\tau(\ell,j),\ell,i}, \text{ all } \ell, j.$$

where $\hat{c}_{\ell,j}$'s are $q_B + 1$ hash outputs.

Case (2): the opposite. Then the generic adversary has computed a nontrivial linear dependence among discrete logrithms of $y_{k'}$, i.e. the generic adversary has solved the one-more discrete logarithm problem.

Remark: In the generic group model (GGM), the above linear dependence is a form of *discrete logarithm collision*. It can be deducted in GGM that the probability of a PPT algorithm being able to compute a discrete logarithm collision, including the kind above, is negligible. □

B Proof of Theorem 5

We prove for **CDS1-type**, $\theta = 1$, first. The generic attacker in GGM of p1m-uf of blind Schnorr SAG signature is as follows:

1. Input: a list of public keys $L = \{y_1, \cdots, y_n\}$.
2. Receive commitments $t_{k,i}$, $1 \le i \le q_{B,k}$ from $\mathsf{BlindSigner}_{PK_i}$. Note $\sum_k q_{B,k} = q_B$.

3. Send challenges $c_{k,i}$, receive responses $s_{k,i}$.
4. Output SAG signatures $\sigma_j = (\hat{t}_{j,1}, \cdots, \hat{t}_{j,n}, \hat{c}_{j,1}, \cdots, \hat{c}_{j,n}, \hat{s}_{j,1}, \cdots, \hat{s}_{j,n})$ on message $\hat{m}_j$, $1 \leq j \leq q_B + 1$.

The queries $\hat{c}_{j,0} = H(L, n, \theta, \hat{m}_j, \hat{t}_{j,1}, \cdots, \hat{t}_{j,n})$, $1 \leq j \leq q_B + 1$, must have been made. Let the Lagrange interpolation be indicated $\sum_{0 \leq \ell' \leq n} \gamma_{\ell'} \hat{c}_{j,\ell'} = 0$. By GGM, there exists a index mapping τ such that, for $1 \leq j \leq q_B + 1$ and $1 \leq \ell \leq n$,

$$\begin{aligned}
\hat{t}_{j,\ell} &= g^{\hat{s}_{j,\ell}} y_\ell^{\hat{c}_{j,\ell}} \\
&= g^{a_{\tau(j,\ell),-1}} \prod_{\ell'} y_{\ell'}^{a_{\tau(j,\ell),\ell'}} \prod_{k,i} t_{k,i}^{b_{\tau(j,\ell),i}} \\
&= g^{a_{\tau(j,\ell),-1}} \prod_{\ell'} y_{\ell'}^{a_{\tau(j,\ell),\ell'}} \prod_{k,i} (g^{s_{k,i}} y_k^{c_{k,i}})^{b_{\tau(j,\ell),i}} \\
1 &= g^{\Delta_{s,j,\ell}} \prod_{\ell'} y_{\ell'}^{\Delta_{c,j,\ell,\ell'}} \text{ where} \\
\Delta_{s,j,\ell} &= -\hat{s}_{j,\ell} + a_{\tau(j,\ell),-1} + \sum_{k,i} s_{k,i} b_{\tau(j,\ell),k,i}, \text{ all } j, \ell \\
\Delta_{c,j,\ell,\ell'} &= -\hat{c}_{j,\ell} \delta(\ell, \ell') + \sum_{\ell'} a_{\tau(j,\ell),\ell'} + \sum_{k,i} c_{k,i} b_{\tau(j,\ell),k,i}, \text{ all } j, \ell, \ell'
\end{aligned}$$

The negligibility of discrete logarithm collision leads to

$$\begin{aligned}
0 &= -\hat{c}_{j,\ell'} + \sum_{\ell'} a_{\tau(j,\ell'),\ell'} + \sum_{k,i} c_{k,i} b_{\tau(j,\ell'),k,i}, \text{ all } j, \ell \\
-\gamma_0 \hat{c}_{j,0} &= \sum_{\ell'=1}^{n} \gamma_{\ell'} \hat{c}_{j,\ell'} \\
&= \sum_{1 \leq \ell' \leq n} \gamma_{\ell'} (\sum_{\ell'} a_{\tau(j,\ell'),\ell'} + \sum_{k,i} c_{k,i} b_{\tau(j,\ell'),k,i})
\end{aligned}$$

for $1 \leq j \leq q_B + 1$. The generic adversary has solved the above ROS Problem, where $\hat{c}_{j,0}$ are $q_B + 1$ hash outputs.

CDS2-type, $\theta \geq 1$. Similar to the above, with the following modificaitons:

The hash queries are $\hat{c}_{j,0}^{(\theta')} = H_{\theta'}(L, n, \theta, \hat{m}_j, \hat{t}_{j,1}, \cdots, \hat{t}_{j,n})$, $1 \leq j \leq q_B + 1$, $1 \leq \theta' \leq \theta$, have been made. The ROS Problem is

$$\begin{aligned}
-\gamma_0^{(\theta')} \hat{c}_{j,0}^{(\theta')} &= \sum_{\ell'=1}^{n} \gamma_{\ell'}^{(\theta')} \hat{c}_{j,\ell'} \\
&= \sum_{1 \leq \ell' \leq n} \gamma_{\ell'}^{(\theta')} (\sum_{\ell'} a_{\tau(j,\ell'),\ell'} + \sum_{k,i} c_{k,i} b_{\tau(j,\ell'),k,i})
\end{aligned}$$

where $\hat{c}_{j,0}^{(\theta')}$'s are $q_B + 1$ hash outputs expressed in terms of q_B commitments $c_{k,i}$'s.

ring-type: The proof is similar and omitted. □

Security for Interactions in Pervasive Networks: Applicability of Recommendation Systems

Seamus Moloney and Philip Ginzboorg

Nokia Research Center, Itamerenkatu 11-18, 00180 Helsinki, Finland
[seamus.moloney, philip.ginzboorg]@nokia.com

Abstract. Recent growth in the sales of Bluetooth-enabled handsets allows short-lived automated interactions between personal devices to become popular outside the research laboratories. In these new kinds of networks, automated data transfer between devices can now be achieved and there are many use cases, but a missing element is a consistent approach to the problem of risk management in automatic interactions. Access to centralized servers is not feasible, so security management will lie in the hands of end-users. We investigate the features present in these networks that could be used to mitigate risk and present existing research in the areas of ad hoc network security and distributed recommendation systems, discussing their potential for solving these problems.

1 Introduction

We shall use the term "pervasive" for networks built using short-range wireless technologies in unlicensed band. Pervasive networks will be unmanaged, based on local wireless bearers and contain a diverse array of devices. Although wireless LAN and ultra-wideband technologies represent alternatives in the long term, Bluetooth is likely to dominate as the bearer technology in those networks, because of the volume of devices supporting it.

Pervasive networks are likely to carry data traffic between peers that do not have previous first-hand knowledge of each other. In order for small pervasive networks to flourish, there must be a means to mitigate the risk inherent in interactions with unknown entities. Mechanisms are needed to automate the decision of when permitting an interaction is desirable and when it is better to refuse the request. Recommendation based systems have been suggested as a potential solution. Research in this area has tended to focus on the problem of secure routing and resource sharing [4, 9]. The many possible smart phone applications and the increasing number of Bluetooth-enabled devices on the market suggests that more kinds of interactions than just packet routing are likely to be developed for pervasive networks. Although smart phones are capable of networking over multiple bearers and even asymmetric cryptographic operations, their limited size and input possibilities, restrict the complexity of what can be implemented and presented to end-users. Bearers in pervasive networks typically have a very short range, (ten meters for Bluetooth) and thus the availability of third parties to aid in decision-making cannot be relied upon.

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 95–106, 2005.

The paper is structured as follows. Section 2 explains the use cases that pervasive networks could enable and the resulting potential security threats. Section 3 looks at the some risk control measures and in particular the role that Bluetooth pairing can play in dealing with these kinds of threats. Trust-based solutions have been applied to deal with similar problems in larger-scale mobile ad hoc networks and Section 4 summarizes both the research problem of encounters with strangers in short-range radio networks and the solutions promised by Bluetooth technology and distributed trust systems, concluding with challenges to future research.

2 Interactions and Risks

The attraction of automated interactions between mobile devices lies not in the new types of interactions they introduce but in the way that they can enhance the already familiar ones. Rather than require new models of behavior, social interaction in proximity can build on the existing human activities of giving, sharing, greeting, competing and acknowledging others. Certainly it is unlikely that people will forge social relationships with strangers just because they can, but users increasingly create shareable digital content and use applications that support sharing. Pervasive networks are well suited to the sharing of data, because they are free to use and have higher data throughput rates than, e.g. the cellular networks can offer.

Applications possible in pervasive networks include collaboration via white boards, bulletin boards and messaging, games and file sharing. Electronic business card exchange at meetings is an example of how such networks are already in use. In these examples the user is usually involved in selecting the peers with whom communication should take place and deciding what should be shared. However, there are many cases where some form of matchmaking should take place without human selection.

2.1 Example Scenarios

The following list is not exhaustive. It is intended to illustrate the range of possibilities and their impact on device security.

A. Joe gets Fifa 2004 for his N-Gage gaming phone, practices hard and sets his device to find anyone who has the same game and can play to a level similar to him.
B. In a crowded bar Bob notices that someone is sharing a java game he has been seeking for some time now and downloads the game with confidence that the game is all he will get.
C. On a crowded metro, automatic super distribution works very well and Eddie manages to collect most of his favorite cartoons from passers by.
D. Standing in the cinema foyer, Jane uses her phone browser to reserve 2 seats for next week's showing of the latest movie and is happy to have the showing time entered into her calendar without all that tricky typing.

All of these scenarios are about actions occurring under certain conditions. Case A implies Joe's intent that the game should only be playable against a peer who has reached a similar level; case B will only allow the java game to be fetched if it can be verified as a genuine game. In case C, the conditions are looser, the phone may interact with any other phone, but the collected content needs to be assessed against Eddie's filter (favorite cartoons). Case C is interesting also because the phones from which Eddie requests the cartoons may have their own ideas about how much content they wish to share with Eddie. Case D involves access to some private data. In this case, Jane is giving permission to an access point at the cinema to create a calendar entry.

The scenarios above illustrate the complexity of interactions in pervasive networks and the need for automation. They also introduce new threats.

2.2 Threats Caused by the Scenarios

We shall use the following terms from Internet Security Glossary [13] when describing the threats:

- Disclosure: an entity gains access to data for which the entity is not authorized.
- Deception: authorized entity receives false data but believes it to be true.
- Disruption: interrupts or prevents the correct operation of system.
- Usurpation: an unauthorized entity gets control of system services or functions.

A peer might claim to have reached a level that they had not reached in scenario A. Joe would end up playing against someone who was a little too easy to beat. Hardly a catastrophe, but some form of deception has taken place. There are many more serious scenarios, but essentially there needs to be a more secure way of communicating the level achieved by the prospective peer in order to preserve the significance of this information.

All of the threats listed above may materialize in scenario B. The downloaded content may contain viruses which could gain access to messaging APIs etc., making it possible to access personal information and vital system resources. The content may be harmless but still require storage space more than the receiver had anticipated, causing disruption in the operation of the device. In addition, financial loss may occur if the downloaded software sends text messages or opens data connections over the cellular network.

Scenario C has similar considerations in that there is a need to download uncertified content. Pervasive networks supporting automatic download of content will probably also require some form of incentive-based routing to encourage cooperation. This would create security requirements for controlling the incentive system. Eddie would need to share content in addition to just picking content from those in the neighborhood. His device would have to contribute battery resources and storage space, potentially leading to disruption. Deception would occur if a peer in the neighborhood succeeded in corrupting the mechanisms of the incentive system.

In scenario D, access is given for calendar entries to be made. If this operation is not managed correctly, it can lead to disclosure of private information form the device.

2.3 Pervasive Networking Interaction Threats

The above discussion illustrates that when devices interact in pervasive networks, a mechanism is needed to shield the users from the following:

1. Bad content: low quality, e.g. mp3 recording of bad sound quality, offensive content; viruses, trojans, irrelevant, mislabeled content; spam.
2. Bad peers who: have been reported as distributors of bad content; try to gain access to confidential data; do not contribute to the co-operative routing effort; are using fake identities or false advertisements; are known to be tracking devices; are eavesdroppers on confidential exchanges.
3. Battery abuse: the amount of memory and battery power allocated for automated transactions should be controlled.

3 Existing Risk Control Measures

Without a centralized infrastructure and perhaps even with one present, eliminating all threats is likely to be impossible. However, ways to mitigate those threats should be designed to make possible the scenarios presented above.

Recognizing the building blocks available for dealing with these threats means assessing the elements already present in pervasive networks as well as those that can be taken from elsewhere and applied. Bluetooth has a built-in security mechanism for establishing trust between devices and for eavesdrop prevention. This mechanism currently requires user input in the initialization phase. It is most suitable for stable security associations, such as for securing the link between a user's laptop and phone (e.g. for data synchronization).

3.1 Bluetooth Security

This section discusses where the Bluetooth security architecture can fit in pervasive networks. The specification relevant to this discussion is the Bluetooth version 1.1 core specification document, [3]. This information is publicly available and is summarized here with the goal of putting into perspective the value of services provided by the Bluetooth security architecture when attempting to create secure interaction modes for pervasive networks.

Bluetooth applications conform to predefined profiles and in the specification, profiles are defined to obey security modes as follows:

- Security mode 1: non-secure
- Security mode 2: service level enforced security
- Security mode 3: link level enforced security

The difference between Security mode 2 and Security mode 3 is that in Security mode 3 the Bluetooth device initiates security procedures before the link is opened for applications.

Security mode 1 is used for example during the exchange of business cards. Many Bluetooth services require more security than is needed in this example. For instance, services that involve synchronization of personal data between devices need mutual authentication and transmission secrecy.

3.2 Pairing in Bluetooth

Security mode 2 implies that Bluetooth pairing is used in link establishment. Pairing involves the same personal identification number (PIN) being entered into each of the devices being linked, resulting in the generation of a symmetric link key. Thereafter the data traffic between the two devices is encrypted and the devices are capable of authenticating each other.

Jakobsson and Wetzel explain in [7] the weaknesses in the link key setup phase. They demonstrate that the strength of the used PIN is critical in preventing an eavesdropper from obtaining the link key. The cipher used in the link encryption itself is found to be safe from practical attacks.

When digits are used, the PIN would need to be at least 20 characters in length to be safe against cryptographic analysis of the observed encrypted traffic. But the paired devices have such limited input facilities that a short PIN is typically used. For this reason, the pairing procedure is not suited to public spaces. An attacker with access to the link key can potentially impersonate either of the two legitimate parties in the link, although he would also need to perform the non-trivial task of modifying the visible Bluetooth address of his device.

New approaches to the key agreement phase are needed for security mode 2, and thus Bluetooth pairing, to become usable in the above scenarios. For instance, an infrared side channel could be used for the secure exchange of a randomly generated long PIN. Balfanz et al. address the problem of bootstrapping secure connections in ad hoc environments by using physical proximity between the devices [2]. Such approaches can improve both the usability and the security of the pairing procedure.

An added advantage of using mode 2 lies in the fact that the key management functionality is already present in devices that support Bluetooth, so a means already exists for users to manage the security associations with other devices.

The difficulty with pairing is that user involvement is required each time a stranger is encountered, a fact that would seem unacceptable for many of the scenarios discussed here. Device authentication and encryption of transmitted data are features that would certainly be useful, but an automated authorization procedure for some services is also needed.

3.3 Alternatives to Pairing

It seems that the essential new feature of pervasive networks is the absence of a fixed infrastructure for authentication. Without a global PKI, there is a strong possibility that nodes in pervasive networks know very little or nothing about peers they encounter. Yet, for pervasive networks to function the participants

need to be prepared to allow interactions with unknown peers, often without user confirmation.

Automated network formation does not imply a complete lack of security. In real life, people routinely run background checks before deciding to do business with strangers. Clearly the needed checks depend on the nature of the business. An approach in which, before proceeding with further communication, the device performs as accurate a background check as possible on the connecting peer can help. The background check should take into account all information of past transactions between these peers and will need to make a decision about whether to risk the threats the interaction implies.

In recent research work [4, 8] these kinds of threats have been tackled using a trust-based approach. The combination of personal experience and recommendations received from others is used to derive a trust value for the peer requesting interaction or the content being offered.

In the networks referred to in this paper, the potential set of peers is extremely large. This may seem to be an argument against the use of trust-based systems and recommendation sharing, because there would seem to be a small likelihood of meeting the same peer twice. However, the short range of pervasive networks and the fact that people are creatures of habit implies that recommendation information can be highly relevant in the local context. There is a reasonable chance of spontaneous peer-clustering, i.e. that the peers will interact again, now that they are in the same physical space, or that another person in the same area will end up interacting with both of them and benefit from the recommendation information they pass on. Therefore, it makes sense to store and actively distribute and accept information about interaction histories.

4 Trust Based Risk Control

In online auction sites, sellers and buyers need to make a decision about whether they are prepared to pay a bill or to send a product to someone they will probably never meet. In these centrally managed systems, membership is controlled and members can view as well as provide feedback about each other. A feedback can be considered a recommendation that a named peer has behaved in a certain way in a certain context. Thus each member gains a reputation based on past behavior. The principle that past behavior can determine attractiveness to potential peers maps well to pervasive networks.

The recommendation information comes from a third party and is likely to be ignored if any first hand experience is available. It may, however, be the only information a peer has to rely on when deciding whether to authorize another peer for an interaction. Analysis of this information can be performed prior to interactions in order to evaluate the risk involved.

4.1 Features of a Trust Based System

The existence of a pervasive network can be roughly divided into four stages:

- Peer discovery and recognition,
- Building of communication links,

– Communication,
– Disconnection and possibly feedback.

A trust-based system is one of the tools that helps the user in the first stage, deciding on whether or not to communicate with a stranger. The data used by the recommendation system needs to be generated by a person recording an observation about an interaction with the peer. This is done in the feedback phase and it usually involves explicit user involvement - effectively rating the peer. There is clearly scope here for automation of feedback generation. For instance, by monitoring system behavior and usage patterns after an application has been downloaded, e.g. how often it was used, one can decide whether the application is worth recommending.

In the evaluation of Ebay by Resnick et al [11], the following characteristics are seen as compulsory for a reputation-based system to succeed:

1. Identities of buyers and sellers need to be long lived in order for future interaction to be likely.
2. Feedback about transactions and interactions must be available for others to read.
3. The ratings participants have must be reliable for them to be used.

4.2 Effect of Distribution

When no fixed infrastructure exists, new issues related to entity recognition and reputation data integrity need to be taken into account.

As Resnick points out, the identities of peers must be long-lived for the recommendations made by them or about them to be meaningful. A centralized member database may provide more security and better ease-of-use than a distributed system can provide.

The data needs to be shared between peers, many of which will have storage restrictions and lack a secure platform for protecting the integrity of the information.

Reputation-based systems rely on the right information being available at the right time. It is unlikely that a system without a central server can guarantee information availability and freshness, meaning that there is an increased margin for error in the calculation of an "average" recommendation value, i.e. reputation, for a peer or for some content.

In summary, in the case of distributed reputation systems there are increased difficulties in three areas:

1. How to tie reputations to peers: identity needs to come from some external source.
   As we mention later in section 4.3, the Bluetooth device address or a credential from the cellular network could be a solution here.
2. How to guarantee the integrity of the passed recommendations.
   Because the devices in question here are capable of generating key pairs and running cryptographic operations, digital signatures could be used for

protecting the integrity of recommendation data. The keys need to be tied to some identifiers for successful verification of signatures. Although this generally implies PKI, the Poblano system [15] illustrates that more flexible approaches are possible. In that solution, the peers pass recommendations to each other which detail a level of confidence in the fact that a certain peer uses a certain public key, thus forming a PGP-like PKI. In [1], the public key information is stored in a decentralized storage system, built so that the integrity of trust data can be checked efficiently.

3. How to guarantee the needed recommendation information is available in the right context.
   As discussed earlier, peer-clustering can mean that recently received recommendation information can be highly relevant - the peers are in the same area.

Point 3 raises the question of how the trust data is managed and shared. On the one hand, the connections in pervasive networks may be short lived (e.g. the cartoon exchange from scenario C). But on the other hand, the data speeds are quite high (more than 400 Kb/s for Bluetooth) and the transmission does not cost anything, so transmission bandwidth is not an obstacle to the exchange of trust-related data.

The portability of the trust data must be considered in a decentralized system: are there standardized ways to represent the recommendation data that would allow different device types to share trust data? A recommendation is essentially an assertion of belief. Therefore it would make sense to adopt the Security Assertion Markup Language (SAML) [10], an OASIS standard, for representing recommendations.

4.3 Known Problems of Trust-Based Systems

In [5], the functioning of the Ebay reputation system was studied over time and the following "misbehaviors" were observed. In each case a workaround was proposed for minimizing the effect of these security threats:

Ballot Stuffing. A seller colludes with a group of buyers to gain unfairly high ratings from them. The resulting increase in seller's reputation allows that seller to receive more orders and at a higher price than she deserves.

The countermeasure is to discard scores from the most frequent reviewers.

Shilling. When two sellers sell similar products, one seller can log in with a number of different identities and "hijack" the other seller's auction, making artificially high bids which are then canceled. This drives authentic buyers to the other auction.

The countermeasure is to make it difficult to have multiple identities in reputation system. In pervasive networks the usage of a single identifier for each device based on that device address is a way to implement this countermeasure.

Bad-Mouthing. This involves collusion among a number of buyers to lower the reputation of a seller by submitting very negative ratings. The opposite can also be true where very high recommendations are given.

The workaround recommended is to reduce the effect of unfair ratings by taking the median rather than the mean of recommendations received.

Problems have also been identified in systems that are more distributed as compared to Ebay. A lot of research has been done in this area and many different trust models have been proposed. When a centralized server is not an option, the following issues arise.

Reputation Initialization and Identity. In [12], the authors point out that for pervasive networks, bootstrapping should involve giving a new peer a small measure of trust so that it can begin collaborating, even though technically this gives it permission to do things it should not be doing. With such an approach, however, it makes very little sense for a peer to attempt any collaborations when it's rating goes below this bootstrap threshold value. It is more advantageous to obtain a new identity and begin with the threshold trust value.

This is the dilemma of many systems built on reputation. To prevent this identity changing, there needs to be some incentive for users to retain their current identity. This could mean that stricter forms of authentication (e.g. based on digital certificates or real-world identifiers) are used to tie reputations to identities, thus increasing the effort required to obtain a new identity. In some scenarios this will obviously conflict with privacy requirements. Allowing devices to use any of a limited set of pseudonyms could be a good compromise solution to this problem.

How Transitive Should Recommendations Be? Often it is necessary to take into account the source of a recommendation rather than handling all recommendations equally. If Alice receives a recommendation from her friend Bob about Dave, she might be more inclined to believe it than the recommendation she received from a stranger Carla about Dave. This introduces complexity to the interaction decision that may be difficult to hide from an end user. In small devices, the effort required to interpret such chains accurately could be greater than the benefit gained, so a simplified approach may be preferable.

Semantics of a Generic Reputation: Accounting for the Desired Action. The nature of the action being requested by the peer should be taken into account when the interaction decision is made. This again introduces complexity, which can make the system difficult for end-users. Who should decide how much risk is evident in each requested action and how can the requested action be described in a manner that makes it universally recognizable? The approach which seems most feasible here is to leave the decision about the risk involved in an action to the developer of the application which implements that action and to require them to provide this information.

Difficulty in Measurement of Performance. A potential problem in evaluating reputation-based systems is that the behavior of the system needs to be

observed over very long periods of time to build up an accurate picture of its performance. Results will be very dependent on the number of unknown entities encountered and rate at which new recommendation information is gained. Small world modeling [14] is a promising approach for simulating interactions in pervasive networks. It has been shown to model social networks accurately and allows for the possibility of controlling the level of network nodes mobility. Our initial experiments with this approach suggest that when recommendations are exchanged prior to interactions, peers who cheat become isolated, particularly when nodes are highly mobile. More research is needed to estimate how well small-world modeling fits pervasive networks and in modeling the various cheating behaviors and their effects.

Recommendations Can Be Copied Easily. As reported in [8], a tendency can develop for people to present the recommendations of others as their own. The copied recommendations can thus lose their significance. This is avoidable if all recommendations are integrity protected and contain an indicator of freshness, like timestamps. Making sure that recommendations really do come from the peers that had the experience means that there is less incentive to switch identities: the recommendations cannot be carried over to the new identity.

Requesters Attach Themselves to Trustworthy Recommenders. There may be a tendency for lockout in pervasive networks if there is no incentive for requesters to believe that they have anything to gain by interacting with people other than their trusted acquaintances. Again an incentive system should take this into account.

Big Ticket Item Cheat. One behavior pattern reported on Ebay is to build up a "solid" reputation by making many small trades, before causing havoc by cheating in a number of transactions involving items of significant value. The way to avoid this is to link the value of the transaction made to corresponding change in reputation. The mapping of this method to non-financial transactions requires more investigation. In the simplest approach, recommendations would have a context that relates them to a specific action and a good reputationwould not lead to actions in higher-risk contexts becoming allowed.

The Sybil Attack. Docouer sets out his claim in [6] that it is not possible to guarantee the distinctness of entities in a distributed system without the aid of a central authority. The Sybil attack is the forging of multiple identities for malicious intent. It undermines the assumed mapping between identities and entities and hence introduces a number of false entities. When a central authority is not available, a peer must use either direct validation of identities or accept indirect validations such as a security assertion that a device A vouches for the identity of a device B. Docouer reasons that these kinds of validations require the issuing of computationally expensive challenges and verification of the responses. In networks where some peers are more powerful than others, this cannot be relied upon. Also challenges need to be issued to all those who have vouched for

the peer simultaneously in order to be sure that they are not themselves faulty entities. This is claimed to be impossible when network size increases.

Docouer dismisses the use of IP addresses as "helpers" in identification because the Internet Corporation For Assigned Names and Numbers (ICANN) cannot be relied on either. In the networks described in this paper, the Bluetooth device address would be the logical identifier choice. Another interesting topic for research is the fact that the devices in pervasive networks may have a parallel connection to the cellular network and use the authentication infrastructure it provides in to prove their identities.

5 Conclusion

Short-range wireless technologies in unlicensed band enable pervasive networks that are composed of mobile devices owned by consumers and in which there is no centralized management. There are many usage scenarios based on spontaneous interaction between devices and there will be encounters between complete strangers in such networks.

Bluetooth represents the most widely used bearer for pervasive networks. Although Bluetooth device pairing can provide to applications the important building blocks of authentication, encryption and authorization, the requirement for automated formation of communication links (also with strangers) makes it difficult to use.

Dealing with unknown entities exposes the user to threats, notably exposure to non-trusted content or misbehaving entities can lead to disruption of the user's device, disclosure of personal data or usurping of device resources. If Bluetooth pairing is not feasible and a centralized authority cannot be relied on to manage interactions, an attractive alternative is disseminating information about experiences with both peers and content, which would enable their trustworthiness to be calculated. In short- range networks, this information is likely to have real relevance in the local context. Thus it effectively provides a means to warn others in the same neighborhood about a potential threat. Moreover, people are creatures of habit and there is a chance of encountering somebody a second time or encountering somebody who has already been in the same network as a future peer. Because of this peer-clustering it makes sense to store information about previous encounters and also to exchange this information with other people.

Decentralized trust-based systems will have to be very simple to be acceptable to end-users. There are challenges in guaranteeing the integrity of trust information in the absence of a secure platform. Systems for mobile devices with limited storage capabilities must also be optimized to ensure the relevant information is present in the right context. The issue of how entities recognize each other without central management is a complex one with many of the known flaws in these systems resulting from the ease with which peers can change identities. Although using trust-based systems in these scenarios is promising, the effectiveness of such systems should also be verifiable, meaning research should also focus on performance measurement.

References

1. Anwitaman D., Hauswirth M., Aberer K.: Beyond "web of trust": Enabling P2P E-commerce Proceedings of the IEEE Conference on Electronic Commerce (CEC'03), June 24-27 2003, Newport Beach, California, USA.
2. Balfanz D., Smetters D., Stewart P.,ChiWong H.: Talking to strangers - Authentication on ad-hoc wireless networks. In Symposium on Network and Distributed Systems Security, San Diego, CA, USA (2002)
3. Bluetooth Special Interest Group: Bluetooth 1.1 Core specification, available from http://www.bluetooth.org/spec/ (2002)
4. Buchegger S., Le Boudec J.-Y., Nodes Bearing Grudges: Towards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks. In Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, Canary Islands, Spain, (2002)
5. Dellarocas C.: Building Trust Online - Design of Robust Reputation Reporting Mechanisms for Online Trading Communities. Avaialable from http://ccs.mit.edu/dell/papers/ideabook.pdf (2002)
6. Docouer, J.: The Sybil Attack, Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02) Cambridge, MA, USA.
7. Jakobsson M,. Wetzel S.: Security Weaknesses In Bluetooth, RSA Conference (2001), San Francisco, CA, USA.
8. Kinateder, M. and Rothermel, K.: Architecture and Algorithms for a Distributed Reputation System, In Proceedings of the First International Conference on Trust Management,Crete, Greece. (2003)
9. Marti S., Giuli T., Lai K. and Baker M.: Mitigating Routing Misbehavior in Mobile ad hoc networks. Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking, Boston, MA, USA (2000)
10. Oasis Security Services TC, Security Assertion Markup Language (SAML) v1.1, available from http://www.oasis-open.org/home/index.php (2003)
11. Resnick, P., Zeckhauser, R., Swanson, J., and Lockwood, K.: The Value of Reputation on eBay: A Controlled Experiment. Working paper presented at the ESA conference, Boston,MA,USA (2002)
12. Seigneur J, Jensen C., S. Farrell S., Gray E., Chen Y.: Towards security auto-configuration for smart appliances, Smart Objects Conference, Grenoble, France (2003)
13. Shirey R.: RFC 2828, Internet Security Glossary, (2000)
14. Watts D., and Strogatz S.: Collective dynamics of small world networks. Nature, 393, 440-442. (1998)
15. Yeager W., Chen R.: Poblano, A distributed Trust Model for JXTA, Sun Microsystems (2001)

Pseudonym Generation Scheme for Ad-Hoc Group Communication Based on IDH

Mark Manulis and Jörg Schwenk

Network and Data Security Group,
Department of Electrical Engineering & Information Sciences,
IC 4/158, Ruhr-Universität Bochum, D-44801, Germany
{mark.manulis, joerg.schwenk}@rub.de

Abstract. In this paper we describe the advantages of using iterative Diffie-Hellman (IDH) key trees for mobile ad-hoc group communication. We focus on the Tree-based Group Diffie-Hellman (TGDH) protocol suite, that consists of group key agreement protocols based on IDH key trees. Furthermore, we consider the anonymity of members during group communication over a public broadcast channel that provides untraceability of messages. The main goal of the proposed pseudonym generation scheme is to allow group members to generate their own pseudonyms that can be linked to their real identities only by a democratic decision of some interacting group members. The real identities are bound to public keys used in the group key agreement. The communication and computation costs as well as the security of the scheme can be optimized with respect to the characteristics of involved mobile devices.

1 Introduction

In this paper we describe the communication and computation advantages of the iterative Diffie-Hellman (IDH) key agreement for ad-hoc group communication scenarios and propose a new pseudonym generation scheme with threshold revocation which can be embedded in the IDH process. We are using IDH key agreement protocols of the Tree-based Group Diffie-Hellman (TGDH) suite, proposed recently in [7]. As communication infrastructure we consider a public broadcast channel between mobile devices that provides untraceability of messages, where only the attached identity values serve as identification of the sender. In our scenario members agree on a group key using the TGDH protocols and then compute their own pseudonyms using our pseudonym generation scheme, which is an extension to TGDH. These pseudonyms are unlinkable to the real identities of members used in the group key agreement. The real identity can be revealed only upon interaction of a certain number of group members in a democratic decision. Besides that, we propose a new communication and computation costs optimizing strategy given by the structure of the key trees established by the iterative Diffie-Hellman key agreement. It allows to choose the optimization cost factor that is common for all mobile devices that take part in ad-hoc communication. The strategy is based on the fact that keys assigned to IDH-tree nodes are shared between members assigned to leaves of the subtree rooted at that node. The optimization of costs is very

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 107–124, 2005.

important for ad-hoc communication, since mobile devices are often limited in their power ressources.

Motivation. Different appliance scenarios can be considered for the ad-hoc group communication with pseudonyms, e. g. members of directing board of a company might want to communicate securely and anonymously in order to perform an anonymous election process, without having to trust into a third party. If at least one of members breaches the communication rules by broadcasting some misleading information, then other members might want to reveal her identity. The decision whether such dispute case has been occured is democratic since none of group members is obliged to take part in the revealing process. This is the main difference to communication scenarios with a designated group manager that decides when dispute case has occured. To achieve such democratic decision our scheme allows a subset of k members to trace any pseudonym to its holder, with k being a power of 2. Another example is an ad-hoc analogon to the GSM TMSI (Temporary Mobile Subscriber Identity) that would allow users to hide the real identity of their mobile devices by generating pseudonyms. In this case users are interested in generating such pseudonyms, since otherwise it would be possible to track their mobile devices.

Organization. The rest of the paper is organized as follows. Section 2 describes the IDH key agreement protocols of the TGDH suite. Section 3 outlines computation and communication costs of the TGDH protocols and shows their advantages in a mobile ad-hoc environment. Section 4 specifies the communication model and requirements defined for the proposed pseudonym generation scheme, that is presented in Section 5. We describe first a special case, when all group members have to interact in order to link a pseudonym to its real identity. Further, we give a generalized pseudonym generation scheme, that uses the structure of the IDH key tree, and mention additional optimizations for the limited power ressources of the mobile devices.

2 Iterative Diffie-Hellman (IDH) Key Agreement

Iterative Diffie-Hellman key agreement was originally proposed in [2] and later more specific in [5] and [7]. In [7] Kim *et. al.* introduce the Tree-based Group Diffie-Hellman (TGDH) protocol suite which allows group members to establish and maintain a group key through a *contributory* agreement, where each member contributes her own share to the common group key. There is no group manager or any other trusted authority required for this agreement. In the following, we give a brief description of the main protocols of the TGDH suite.

2.1 IDH Key Tree

The IDH key tree used in the TGDH protocol suite is a logical binary tree, referred to as $\mathcal{T}$. It consists of nodes $\langle l, v\rangle$, the v-th node at level l, $0 \leq v \leq 2^l - 1$ and $0 \leq l \leq h$ where h is the height of $\mathcal{T}$. Group members are represented by leaf nodes. Node $\langle 0, 0\rangle$ is the root of $\mathcal{T}$. Each node $\langle l, v\rangle$ is associated with a key $K_{\langle l,v\rangle}$ and a blinded key

(bkey) $BK_{\langle l,v\rangle} = f(K_{\langle l,v\rangle})$, where f is an exponentiation function $f(k) = g^k$ in a multiplicative cyclic prime order group $\mathbb{G}$ with generator g. If we assume that leaf node $\langle l, v\rangle$ hosts member M, then M knows the keys of all nodes on the path from node $\langle l, v\rangle$ to $\langle 0, 0\rangle$, referred to as the *key path*. Keys associated with leaves are chosen by the group members. The keys $K_{\langle l,v\rangle}$ in $\mathcal{T}$, where $\langle l, v\rangle$ is not a leave, are computed iteratively using the Diffie-Hellman key exchange as follows:

$$\begin{aligned} K_{\langle l,v\rangle} &= (BK_{\langle l+1,2v+1\rangle})^{K_{\langle l+1,2v\rangle}} \\ &= (BK_{\langle l+1,2v\rangle})^{K_{\langle l+1,2v+1\rangle}} \\ &= g^{K_{\langle l+1,2v\rangle} K_{\langle l+1,2v+1\rangle}} \\ &= f(K_{\langle l+1,2v\rangle} K_{\langle l+1,2v+1\rangle}) \end{aligned}$$

Member M can compute key $K_{\langle l,v\rangle}$ of any node $\langle l, v\rangle$ on her key path only if she knows the key of its preceeding node $\langle l + 1, 2v\rangle$ or $\langle l + 1, 2v + 1\rangle$ and the bkey of the sibling node of that preceeding node ($\langle l + 1, 2v + 1\rangle$ or $\langle l + 1, 2v\rangle$, respectively). Every blinded key $BK_{\langle l,v\rangle}$ is sent over the broadcast channel after being computed by one of the group members in the subtree of a node. Every member M knows her position in $\mathcal{T}$ and can therefore determine which blinded keys are needed to compute the keys on her key path. The computation of a key path is finished with the calculation of the group key $K_{\langle 0,0\rangle}$. For security reasons $K_{\langle 0,0\rangle}$ should not be used directly for the purpose of encryption, authentication or data integrity, but some derived key (e.g. $K_{group} = h(K_{\langle 0,0\rangle})$, where h is a strong hash function).

For example, in Fig. 1, M_2 can compute $K_{\langle 2,0\rangle}$, $K_{\langle 1,0\rangle}$ and $K_{\langle 0,0\rangle}$ using $K_{\langle 3,1\rangle}$, $BK_{\langle 3,0\rangle}$, $BK_{\langle 2,1\rangle}$ and $BK_{\langle 1,1\rangle}$.

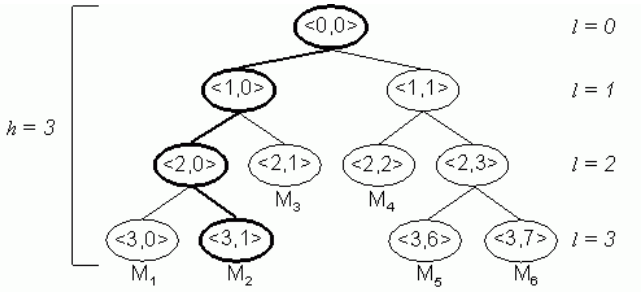


Fig. 1. IDH Key Tree of TGDH Protocol Suite

To establish and maintain such an IDH key tree the TGDH suite provides JOIN, LEAVE, PARTITION and MERGE protocols. Due to space limitations we describe only JOIN and LEAVE.

2.2 TGDH Protocol Suite: JOIN

For a set of group members $\mathcal{M} = \{M_1, \ldots, M_n\}$ and a new member M_{n+1} Fig. 2 describes the JOIN protocol of the TGDH suite. The insertion point of the new member

1. M_{n+1} chooses her own key K_{n+1}, computes bkey BK_{n+1} and broadcasts the latter as request for join
2. Every member M_i, $1 \leq i \leq n$
   - calculates the insertion point of M_{n+1} in $\mathcal{T}$
   - updates $\mathcal{T}$ by adding two leaf nodes for M_{n+1} and for the sponsor at the insertion point node
   - removes (invalidates) all keys and bkeys along the key path of the new leaf nodes

   The *sponsor* M_s additionally
   - computes all (key, bkey) pairs on her key path
   - broadcasts an updated tree $\hat{\mathcal{T}}$ including bkeys of all its nodes
3. Every member M_i, $1 \leq i \leq n+1$ updates her copy of $\mathcal{T}$ with the new bkeys and computes new keys for all the nodes of her key path intersecting the sponsor node key path, including the group key $K_{\langle 0,0 \rangle}$

Fig. 2. TGDH JOIN Protocol

is the rightmost node, where (if possible) JOIN does not increase the height of $\mathcal{T}$ (and else in a completely balanced tree simply the rightmost node). The sponsor is the group member initially located at the insertion point; he will move to the left leave node which will be added to $\mathcal{T}$ during the JOIN operation. The right leave node will be inhabited by the new group member. The sponsor shares the first Diffie-Hellman key with the new member, updates the tree and broadcasts the changed blinded key(s).

Fig. 3 shows an example of member M_4 joining a group $\mathcal{M} = \{M_1, M_2, M_3\}$ at insertion point $\langle 1,1 \rangle$. Member M_3 is hosted by node $\langle 1,1 \rangle$ and is therefore the sponsor. She renames her leaf node $\langle 1,1 \rangle$ to $\langle 2,2 \rangle$, then adds new intermediate node $\langle 1,1 \rangle$ and new member's leaf node $\langle 2,3 \rangle$ to $\mathcal{T}$, computes keys $K_{\langle 1,1 \rangle}$ and $K_{\langle 0,0 \rangle}$ on her key path using her own key $K_{\langle 2,2 \rangle}$ and bkeys $BK_{\langle 2,3 \rangle}$ and $BK_{\langle 1,0 \rangle}$. After M_3 broadcasts the updated tree $\hat{\mathcal{T}}$ every member can compute keys on her key path too.

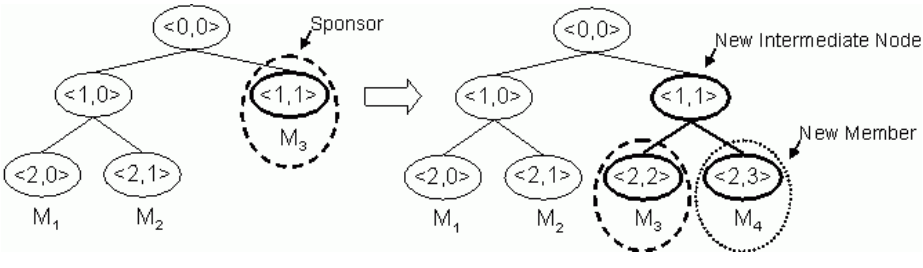


Fig. 3. Example of JOIN Protocol

2.3 TGDH Protocol Suite: LEAVE

For a set of group members $\mathcal{M} = \{M_1, \ldots, M_n\}$ let M_d be a member who leaves the group. Fig. 4 shows the LEAVE protocol of the suite. The sponsor is the rightmost leaf node of the subtree rooted at the sibling node of M_d. Fig. 5 shows an example of member M_3 leaving the group. Every remaining member deletes nodes $\langle 2,2 \rangle$ and $\langle 1,1 \rangle$ from $\mathcal{T}$.Sponsor M_5 computes changed keys $K_{\langle 1,1 \rangle}$ and $K_{\langle 0,0 \rangle}$ on her key path using her own

1. Every member M_i, $1 \leq i \leq n$, $i \neq d$
   - updates $\mathcal{T}$ by removing the member node of M_d and replacing its parent node with the sibling node of M_d
   - removes (invalidates) all keys and bkeys along the key path of the leaving member node

   The *sponsor* M_s additionally
   - computes all (key, bkey) pairs on her key path
   - broadcasts an updated tree $\hat{\mathcal{T}}$ including bkeys of all its nodes
2. Every member M_i, $1 \leq i \leq n-1$ updates her copy of $\mathcal{T}$ with the new bkeys and computes new keys for all the nodes of her key path intersecting the sponsor node key path, including the group key $K_{\langle 0,0 \rangle}$

Fig. 4. TGDH LEAVE Protocol

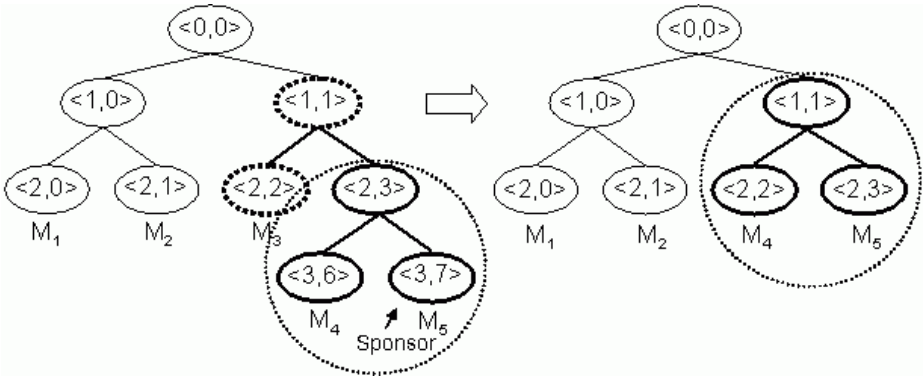


Fig. 5. Example of LEAVE Protocol

key $K_{\langle 2,3 \rangle}$ and bkeys $BK_{\langle 2,2 \rangle}$ and $BK_{\langle 1,0 \rangle}$. After M_5 broadcasts the updated tree $\hat{\mathcal{T}}$ every remaining member can compute keys on her key path too.

3 IDH Key Agreement in Ad-Hoc Networks

Let us summarize some properties of the TGDH protocols that make them suitable for mobile ad-hoc group communication scenarios. When designing protocols for wireless ad-hoc communication, several factors should be considered.

- The relationship between communication participants in mobile networks is highly dynamical and temporary. Communication can be interrupted at any time due to the mobility of nodes. Especially for secure group communication in such environments fault tolerance and efficient dynamic maintenance are required for the group key agreement.
- Another property of mobile ad-hoc networks is the absence of trusted nodes. Any node can be a potential adversary. Therefore a decentralized and cooperative participation of all nodes having equal rights must be provided to achieve desired security for the group key establishment.

– Mobile stations are limited in their computational capability compared to other stationary devices.

Protocols of the TGDH suite fulfill the described requirements. The dynamical and temporary nature of the interaction of group members prohibits the use of symmetric key cryptography, which needs a secure channel to establish cryptographic keys. Amongst the published solutions for public key group key agreement schemes, the TGDH protocols offer the best performance for JOIN and LEAVE operations. Computation and communication costs of the specified protocols are shown in Table 1. The height of the current IDH key tree, number of merging groups and leaving members are denoted by: h, k and p, respectively.

Table 1. Computation and Communication Costs of the TGDH Protocols

	Computation			Communication	
	Exponentiations	Signatures	Verifications	Rounds	Messages
JOIN	$\frac{3h}{2}$	2	3	2	3
LEAVE	$\frac{3h}{2}$	1	1	1	1
MERGE	$\frac{3h}{2}$	$\log_2 k + 1$	$\log_2 k$	$\log_2 k + 1$	$2k$
PARTITION	$3h$	$min(\log_2 p, h)$	$min(\log_2 p, h)$	$min(\log_2 p, h)$	$2h$

Kim *et. al.* [7] compare the TGDH protocols to other existing contributory group key agreement protocols like GDH.3 [8], BD (Burmester-Desmedt) [3], and STR [6]. Despite the higher cost of partition, the TGDH suite shows the best performance on low and medium delay networks. In [1] is shown, how delay in wireless ad-hoc networks can be kept low without trading off the throughput while capacity of a network increases. This confirms the possibility of efficient usage of the TGDH protocols for wireless ad-hoc communication. Mobile peers with limited power can perform the protocols due to their low computation costs. The contributory nature of the protocols allows to compute the group key in a decentralized and cooperative manner without trusted parties. The sponsor in the protocols is not a privileged entity. She is chosen according to the actual tree structure and her messages can be verified by other participants, so that no compromisation of the key agreement is possible. Additionally, we outline some points about fault tolerance.

Suppose that a new member M_{n+1} wants to join the group and looses her connection after sending her blinded key BK_{n+1} in step 1 of the JOIN protocol. Despite that, remaining group members can still perform steps 2 and 3 and agree on a new group key. If M_{n+1} restores her connection to the group, she would only need to verify the knowledge of her corresponding own key K_{n+1}. After receiving the updated tree, she can compute the group key according to step 3 of the protocol without additional interaction.

Network failures that lead to more partitions of the group can be managed using the PARTITION protocol. With the MERGE protocol different groups can be efficiently merged to a common group.

4 Communication Model and Problem Specification

In this section we specify the communication model and define some requirements for our pseudonym generation scheme.

COMMUNICATION MODEL: A set $\mathcal{M} = \{M_1, \ldots, M_n\}$ of n network participants, called *members*, communicates over a public broadcast channel, that provides untraceability of messages. The sender of the message can only be identified upon some identity value id attached to a message.

We consider that members communicate in order to form a dynamic group whithout having any privileged members or trusted parties. Especially, there is no group manager with extended rights. Each member has a unique real identity, which she attaches to messages of the key agreement protocol. Our main goal is to give members pseudonyms, that can be used as id values in messages in order to hide the real identities once the group key is established. Pseudonyms should fulfil the following requirements:

- Each pseudonym can be linked to the real identity of M_i only through the efficient interaction of k members, where k is a fixed value
- The pseudonym of any member M_i must be computed efficiently with respect to the given computational and communicational ressources
- Any member must be able to choose her own pseudonym

5 Pseudonym Generation Scheme Based on IDH Key Trees

5.1 Pseudonym Generation Scheme (Special Case $k = n$)

Our main aim is to embed pseudonym generation in the IDH key agreement. Therefore computations of our scheme are done in the same group $\mathbb{G}$, that is used by the TGDH protocols. Recall that $\mathbb{G}$ is a multiplicative cyclic prime order group with generator g and keys of the IDH key tree are computed using some exponentiation function $f(k) = g^k$.

Remark 1. We remark at this point, that our scheme works also in any other cyclic group, where the *Computation Diffie-Hellman Problem* is hard. Therefore $\mathbb{G}$ can also be a group of points of an elliptic curve E over a finite field $\mathbb{F}_p$ with prime p or $\mathbb{F}_{2^m}$. Such groups are appropriate for use in mobile ad-hoc networks because of the low computation costs of the group operations.

First we describe how our pseudonym generation works using a given IDH key tree $\mathcal{T}$ for a set of current group members $\mathcal{M} = \{M_1, \ldots, M_n\}$. Pseudonyms generated by our scheme can be linked to the real identities only upon the interaction of k members. In this section we describe a special case of $k = n$. In Section 5.2 we then show how k and computational costs can be optimized with respect to the limited ressources of given ad-hoc devices using the structure of the IDH key tree.

As mentioned in Section 2 a member M is represented by a leaf node $\langle l, v \rangle$ with private key $K_{\langle l,v \rangle}$ and blinded key $BK_{\langle l,v \rangle}$. For simplicity we denote them as K_i and

BK_i respectively, for a member $M_i \in \mathcal{M}, i \in \{1, \ldots, n\}$. Blinded keys BK_i are public and computed as

$$BK_i = g^{K_i}$$

Let M_s perform our pseudonym scheme. M_s computes the *shared keys*

$$skey_{sj} = BK_j^{K_s} = g^{K_j K_s}$$

for $j \in \{1, \ldots, n\}$, $j \neq s$. For a chosen pseudonym ps_s and each shared key $skey_{sj}$, M_s constructs the *secret share*

$$s_{sj} = skey_{sj}^{ps_s} = g^{K_j K_s ps_s}$$

and uses it to compute the *public share*

$$S_s = g^{\sum_j s_{sj}} = g^{\sum_j g^{K_j K_s ps_s}}$$

M_s broadcasts the public share and uses pseudonym ps_s as her identity value id in her further group messages.

Remark 2. Although, our scheme allows any value to be chosen for the pseudonym ps, a better choice is to chose some secret random value xs and compute $ps = g^{xs}$. This would allow the holder of the pseudonym to use (xs, ps) as a public-key pair during the communication, e. g. to sign or encrypt messages using own pseudonym.

In the following, we describe how all n group members interact to link the pseudonym ps_s to its holder.

Every member M_i computes secret shares

$$s_{ij} = skey_{ij}^{ps_s} = g^{K_j K_i ps_s}$$

for $j \in \{1, \ldots, n\}$, $j \neq i$ and broadcasts *hidden secret shares*

$$r_{ij} = g^{s_{ij}} = g^{g^{K_j K_i ps_s}}$$

M_i can verify whether pseudonym ps_s is linked to member M_s by computing

$$R_s = \prod_j r_{js} = \prod_j g^{g^{K_s K_j ps_s}}$$

using hidden secret shares r_{js}, $j \in \{1, \ldots, n\}$, $j \neq s$ and comparing the product to the public share S_s of M_s.

Theorem 1 (Special Case $k = n$). *If pseudonym ps_s was correctly used in the construction of S_s by member M_s then $R_s = S_s$.*

Proof.

$$S_s = g^{\sum_j s_{sj}} = g^{\sum_j g^{K_s K_j ps_s}} = \prod_j g^{g^{K_s K_j ps_s}} = \prod_j r_{js} = R_s$$

□

Member M_i does not know in advance who owns the pseudonym ps_s. Therefore, the above computations should be done for all members simultaneously.

5.2 Generalized Pseudonym Generation Scheme

Our first requirement on a pseudonym in Section 4 is, that it can be linked to the owner only by interaction of k members, where k is fixed. So far, we have presented the special case $k = n$, that is all group members must exchange their hidden secret shares in order to link a pseudonym. In the following we give the generalized form of the scheme. Recall from Section 2, that members are hosted by leafs of the IDH key tree $\mathcal{T}$. Using the structure of $\mathcal{T}$, k can be varied by choosing different levels for the public keys to be used in the construction of secret shares. Let M_s perform the generalized scheme for a chosen level l and pseudonym ps_s. M_s first computes the shared keys

$$skey_{s\langle l,v\rangle} = (BK_{\langle l,v\rangle})^{K_s} = g^{K_{\langle l,v\rangle}K_s}$$

for $0 \leq v \leq 2^l - 1$ and the corresponding secret shares

$$s_{s\langle l,v\rangle} = skey^{ps_s}_{s\langle l,v\rangle} = g^{K_{\langle l,v\rangle}K_s ps_s}$$

and then builds her public share

$$S_s = g^{\sum_v s_{s\langle l,v\rangle}} = g^{\sum_v g^{K_{\langle l,v\rangle}K_s ps_s}}$$

and broadcasts it. The linking process requires the interaction of k members, referred to as set $\mathcal{K} \subseteq \mathcal{M}$. $\mathcal{K}$ should consist of at least one representative member from all subgroups, that are rooted at nodes $\langle l, v\rangle$, $0 \leq v \leq 2^l - 1$, so that there is at least one member, who knows the corresponding private key $K_{\langle l,v\rangle}$ of each secret share. Set $\mathcal{K}$ can be generally defined as

$$\mathcal{K} = \{\{M_{K_{\langle l,0\rangle}}, \ldots, M_{K_{\langle l,2^l-1\rangle}}\} \mid M_{K_{\langle l,v\rangle}} \in \mathcal{M} \text{ is a member of a subgroup rooted at node } \langle l, v\rangle\}$$

The following dependency is given between $k = |\mathcal{K}|$ and chosen level l:

$$k \leq 2^l\,, \quad 0 \leq l \leq h$$

In order to link pseudonym ps_s to its owner M_s, every member $M_{K_{\langle l,v\rangle}} \in \mathcal{K}$ must compute secret shares using private key $K_{\langle l,v\rangle}$ and public key BK_i of every other member $M_i \in \mathcal{M}$ as follows

$$s_{\langle l,v\rangle i} = skey^{ps_s}_{\langle l,v\rangle i} = BK_i^{K_{\langle l,v\rangle}ps_s} = g^{K_i K_{\langle l,v\rangle}ps_s}$$

and broadcast the hidden secret shares

$$r_{\langle l,v\rangle i} = g^{s_{\langle l,v\rangle}\, i} = g^{g^{K_i K_{\langle l,v\rangle}\, ps_s}}$$

for $0 \leq v \leq 2^l - 1$. Any member $M_i \in \mathcal{M}$ can prove whether pseudonym ps_s is owned by member M_s upon computing the product

$$R_s = \prod_v r_{\langle l,v\rangle s} = \prod_v g^{g^{K_s K_{\langle l,v\rangle}\, ps_s}}$$

using hidden secret shares $r_{\langle l,v\rangle s}$, $0 \leq v \leq 2^l - 1$ and comparing it with the public share S_s of M_s.

Theorem 2 (Generalized Case $k \leq 2^l$). *If pseudonym ps_s was correctly used in the computation of S_s by member M_s, then $R_s = S_s$.*

Proof.

$$S_s = g^{\sum_v s_{s\langle l,v\rangle}} = g^{\sum_v g^{K_{\langle l,v\rangle} K_s ps_s}} = \prod_v g^{g^{K_s K_{\langle l,v\rangle} ps_s}} = \prod_v r_{\langle l,v\rangle s} = R_s$$

□

The generalized pseudonym scheme is useful in ad-hoc networks, because the choice of level l and that is of k can be used to optimize costs, with respect to limited ressources of given mobile devices. In order to keep down computation costs, a lower level l should be chosen. There is more on computation and communication costs in Section 5.5.

Fig. 6 shows an example of the IDH key tree $\mathcal{T}$ for a set of five members $\mathcal{M} = \{M_1, M_2, M_3, M_4, M_5\}$. Assume, that with respect to their computation ressources members

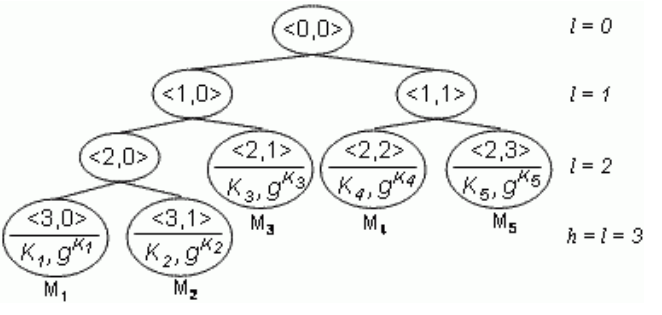


Fig. 6. Example: $\mathcal{T}$ ready for Pseudonym Generation

agree for pseudonyms generation to use public keys of level 1, that are $BK_{\langle 1,0\rangle}$ and $BK_{\langle 1,1\rangle}$. Public shares S_i are computed and broadcasted by members according to Table 2. Pseudonym ps_i can then be used as identity value for the messages of M_i.

To link pseudonym ps_i set $\mathcal{K}$ should consist of at least 2 members. One of them should know $K_{\langle 1,0\rangle}$ and another $K_{\langle 1,1\rangle}$. Set $\mathcal{K}$ is defined as

$$\mathcal{K} = \{\{M_{K_{\langle 1,0\rangle}}, M_{K_{\langle 1,1\rangle}}\} | M_{K_{\langle 1,0\rangle}} \in \{M_1, M_2, M_3\}, M_{K_{\langle 1,1\rangle}} \in \{M_4, M_5\}\}$$

We assume that $\mathcal{K} = \{M_1, M_4\}$ and pseudonym ps_2 should be linked to its owner. As shown in Table 3, members of $\mathcal{K}$ compute and broadcast hidden secret shares $r_{\langle 1,v\rangle i}$ for $v = \{0, 1\}$.

Members M_1, M_2, and M_3 belong to the subgroup routed at node $\langle 1, 0\rangle$, thus they all can compute hidden secret shares $r_{\langle 1,0\rangle i}$ for every $M_i \in \mathcal{M}$. Therefore, if M_1 sends falsified hidden secret shares, this would be noticed by other subgroup members. For the

Table 2. Example: Computation of public share S_i

Member $M_i \in \mathcal{M}$	Public share S_i
M_1	$g^{BK_{\langle 1,0\rangle}^{K_1 ps_1} + BK_{\langle 1,1\rangle}^{K_1 ps_1}}$
M_2	$g^{BK_{\langle 1,0\rangle}^{K_2 ps_2} + BK_{\langle 1,1\rangle}^{K_2 ps_2}}$
M_3	$g^{BK_{\langle 1,0\rangle}^{K_3 ps_3} + BK_{\langle 1,1\rangle}^{K_3 ps_3}}$
M_4	$g^{BK_{\langle 1,0\rangle}^{K_4 ps_4} + BK_{\langle 1,1\rangle}^{K_4 ps_4}}$
M_5	$g^{BK_{\langle 1,0\rangle}^{K_5 ps_5} + BK_{\langle 1,1\rangle}^{K_5 ps_5}}$

Table 3. Example: Broadcasted hidden secret shares $r_{\langle 1,v\rangle i}$

Member $M_i \in \mathcal{K}$	$r_{\langle 1,v\rangle 1}$	$r_{\langle 1,v\rangle 2}$	$r_{\langle 1,v\rangle 3}$	$r_{\langle 1,v\rangle 4}$	$r_{\langle 1,v\rangle 5}$
M_1	$g^{BK_1^{K_{\langle 1,0\rangle} ps_2}}$	$g^{BK_2^{K_{\langle 1,0\rangle} ps_2}}$	$g^{BK_3^{K_{\langle 1,0\rangle} ps_2}}$	$g^{BK_4^{K_{\langle 1,0\rangle} ps_2}}$	$g^{BK_5^{K_{\langle 1,0\rangle} ps_2}}$
M_4	$g^{BK_1^{K_{\langle 1,1\rangle} ps_2}}$	$g^{BK_2^{K_{\langle 1,1\rangle} ps_2}}$	$g^{BK_3^{K_{\langle 1,1\rangle} ps_2}}$	$g^{BK_4^{K_{\langle 1,1\rangle} ps_2}}$	$g^{BK_5^{K_{\langle 1,1\rangle} ps_2}}$

same reason members M_4 and M_5, that belong to the subgroup rooted at node $\langle 1,1\rangle$, can prove hidden secret shares being broadcasted by each other.

For computation of the product R_i every member $M_j \in \mathcal{M}$ has to multiply broadcasted hidden secret shares $r_{\langle 1,0\rangle i}$ and $r_{\langle 1,1\rangle i}$. After M_j builds the products R_1, R_2, R_3, R_4, and R_5, she compares them to public shares. Comparing $R_2 = S_2$, M_j can figure out, that member M_2 owns the pseudonym ps_2.

5.3 Embedding the Pseudonym Generation Scheme in TGDH

In this section we show how the generalized pseudonym scheme can be embedded in group key agreement protocols of the TGDH suite. Every new member generates her pseudonym upon joining the group. Every former group member should change her pseudonym, otherwise the pseudonym of the new member can be easily figured out. We change the last point of step 2 and extend step 3 of the original JOIN protocol of Section 2 to be conform with our pseudonym generation scheme, as described on Fig. 7. Our modifications are marked bold. Our protocols do not specify how to agree on value k in advance, however one possibility is to calculate an optimal value k for each participating mobile device and then to choose the lowest in order to allow the least powerful device to take part in communication. Another possibility is to publish optimal values for k and corresponding mobile device properties so that members can look up and set values in advance. Step 4 prevents any member from having more than one pseudonym and specifies which pseudonyms are valid for communication. If less or more than $n+1$ pseudonyms are received, pseudonym generation can be repeated without changing the group key. A member, who permanently compromises the generation, can be figured out upon revealing pseudonyms of all members. The compromising member can then be excluded from the group communication.

2. ...
   The *sponsor* M_s additionally
   - computes all (key, bkey) pairs on her key path
   - broadcasts an updated tree $\hat{\mathcal{T}}$ including bkeys of all its nodes
3. Every member M_i, $1 \leq i \leq n+1$
   - updates her copy of $\mathcal{T}$ and computes new group key $K_{\langle 0,0 \rangle}$
   - **chooses new pseudonym ps_i, computes new secret shares $s_{\langle l,v \rangle i}$ and broadcasts public share S_i with attached real identity**
   - **builds keyed-hashing value of ps_i using new group key $K_{\langle 0,0 \rangle}$ and broadcasts it anonymously**
4. **Every member verifies received hash values and ensures that there are $n+1$ pseudonyms**

Fig. 7. GENERATION Protocol as Extension to TGDH JOIN Protocol

1. Every member $M_{K_{\langle l,v \rangle}} \in \mathcal{K}$
   - computes n hidden secret shares $r_{\langle l,v \rangle i}$, using ps
   - combines all $r_{\langle l,v \rangle i}$ to a single message and builds its keyed-hashing value using $K_{\langle 0,0 \rangle}$
   - broadcasts the message and its keyed-hashing value with attached real identity
2. Every member $M_s \in \mathcal{M}$
   - verifies keyed-hashing values of received messages and extracts values $r_{\langle l,v \rangle i}$
   - computes product R_i for every other member $M_i \in \mathcal{M}$, $i \neq s$
   - finds public share $S_i = R_i$ and determines member M_i as owner of ps

Fig. 8. Basic LINKING Protocol

In order to link pseudonym ps to its holder, members perform the Basic LINKING protocol from Fig. 8. This protocol uses set $\mathcal{K}$, that consists only of one member of each subgroup rooted at node $\langle l, v \rangle$, which public key was used in the computation of the secret share $s_{\langle l,v \rangle i}$. The Basic LINKING protocol has higher computation costs than the GENERATION protocol (see Section 5.5). However, that can be acceptable, because the linking of a pseudonym is a rare case.

Anyway, there is a possibility of optimizing the Basic LINKING protocol. The optimization uses the fact, that all members of a subgroup rooted at node $\langle l, v \rangle$ can compute the same secret shares $s_{\langle l,v \rangle i}$. Therefore, the computation of n hidden secret shares $r_{\langle l,v \rangle i}$ in the first step of the Basic LINKING protocol can be done in parallel by several members of the same subgroup. The possible computation gain is linear proportional to the number of involved members of each subgroup, that we denote as t. This number depends on the chosen level l and height h of $\mathcal{T}$, and is calculated as

$$t \leq 2^{h-l}$$

Therefore, the possible maximal computation gain varies between 1, for $l = h$, and n, for $l = 0$. This optimization can be applied upon the redefinition of the set $\mathcal{K}$ from the previous section, thus

$$\mathcal{K} = \{\mathcal{M}_{K_{\langle l,0 \rangle}} \cup \ldots \cup \mathcal{M}_{K_{\langle l,2^l-1 \rangle}} \mid \mathcal{M}_{K_{\langle l,v \rangle}} \subseteq \mathcal{M} \text{ is a set of } t \text{ members, who all belong to a subgroup rooted at node } \langle l, v \rangle\}$$

The number of members, that participate in the Optimized LINKING protocol is $|\mathcal{K}| = k \cdot t$, where k is the number of participants in the Basic LINKING protocol. Each of these members sends her own hidden secret shares during the linking process. This means, that during the optimized protocol every member of $\mathcal{M}$ receives t times more messages, compared to the basic protocol. However, the total length of these messages remains the same, because overall n hidden secret shares must be sent in both protocols. Fig. 9 shows the Optimized LINKING protocol. The computation costs are optimized by factor t.

1. Every member $M_{K_{\langle l,v\rangle}} \in \mathcal{M}_{K_{\langle l,v\rangle}}$
   - computes $\frac{n}{t}$ hidden secret shares $r_{\langle l,v\rangle i}$, using ps
   - combines all $r_{\langle l,v\rangle i}$ to a single message and builds its keyed-hashing value using $K_{\langle 0,0\rangle}$
   - broadcasts the message and its keyed-hashing value with attached real identity
2. Every member $M_s \in \mathcal{M}$
   - verifies keyed-hashing values of received messages and extracts values $r_{\langle l,v\rangle i}$
   - computes product R_i for every other member $M_i \in \mathcal{M}$, $i \neq s$
   - finds public share $S_i = R_i$ and determines member M_i as owner of ps

Fig. 9. Optimized LINKING Protocol

5.4 On Security of the Pseudonym Generation Scheme

In this section we consider different attacks on the Pseudonym Generation Scheme and show its resistance against them. In order to link pseudonym ps_i to the real identity of its holder, adversary must be able to recompute the public share S_i or to compute the corresponding product R_i. Recomputing S_i requires knowledge of all secret shares $s_{\langle l,v\rangle i}$. The security of each secret share is however based on the *Computational Diffie-Hellman Problem* (CDH), since only members, who know the corresponding private key $K_{\langle l,v\rangle}$ or K_i can recompute $s_{\langle l,v\rangle i}$, that is except for M_i only $M_{K_{\langle l,v\rangle}}$ can recompute $s_{\langle l,v\rangle i}$. Thus, passive adversary is not able to compute all secret shares unless the CDH problem can be solved in polynomial time. To compute product R_i, one must know all hidden secret shares $r_{\langle l,v\rangle i}$. Hidden secret shares can only be computed through exponentiation of g with the corresponding $s_{\langle l,v\rangle i}$, that is protected as stated above. Therefore, only shareholders can compute secret shares, thus interaction between all shareholders is required, to be able to link ps_i to its owner M_i.

A possible security risk is given, when adversary M_a knows a secret share $s_{\langle l,v\rangle i}$ between $M_{K_{\langle l,v\rangle}}$ and M_i for some pseudonym ps. M_a can compute the inverse value ps^{-1} and the shared key $skey_{\langle l,v\rangle i} = s_{\langle l,v\rangle i}^{ps^{-1}}$. Using this shared key, M_a can compute secret share $s_{\langle l,v\rangle i}$ for any other pseudonym ps'. As already mentioned, the security of single secret shares is given by the hardness of the CDH problem.

An adversary that aims to forge a pseudonym of the member requires all her shared keys in order to compute the correct public share. As stated above any shared key can only be computed by members that share it. Therefore, the only way for the adversary to forge a pseudonym of the member is to collude with k other members that share keys with the victim.

During the GENERATION protocol every member M_i computes and broadcasts her public share S_i. In the next round she broadcasts her pseudonym ps_i anonymously. Since authentication of broadcasted pseudonyms is not provided, any group member adversary can broadcast a false value for S_i or ps_i. In the linking process none of calculated products R_j would match such S_i. Thus it would be not possible to find out which member has broadcasted the false public share or pseudonym, unless all members reveal their pseudonyms and prove the correctness of their signed shares. Any member, who fails to provide this proof, is an adversary and should be excluded from the communication. In this case every member decides on her own whether to reveal her pseudonym to the others and get free of accusations or to continue hiding her pseudonym and not allow other members to identify her previously broadcasted messages. Therefore, the proposed scheme is applicable in scenarios where members are interested in the linking of their pseudonyms at the end of communication, e. g. auctions or polls.

This weakness of the scheme can be repaired if every member M_s would prove the knowledge of a corresponding link ($ps_s \leftrightarrow S_s$) without revealing it to other group members. In Appendix A we describe a protocol that can be used to prove the knowledge of such link.

At last we discuss some risks given by the collusion of some group members. If all group members that know the secret key $K_{\langle l,v\rangle}$ collude then they can bar the group from the linking process for a pseudonym ps_i by broadcasting the same false value for the hidden secret share $r_{\langle l,v\rangle i}$. In this case other group members are not able to compute the correct value of the product R_i, thus link ps_i to S_i. Our scheme allows optimization against the collusion risk by choosing a lower level l. It is trivial that the lower l is the more group members know the secret key $K_{\langle l,v\rangle}$, thus more group members have to collude.

5.5 On Complexity of the Pseudonym Generation Scheme

In this section we discuss computation and communication costs of our scheme without considering the complexity of the TGDH suite protocols. Let $\mathcal{M} = \{M_1, \ldots, M_n\}$ be a set of members participating in the TGDH key agreement protocol, $\mathcal{T}$ the associated IDH key tree with height h, and k, $k = 2^l$ and $0 \leq l \leq h$, a number of required interacting members in the linking process of a pseudonym. The complexity of our scheme for each member is summarized in Table 4.

Table 4. Computation and Communication Costs of the Pseudonym Generation Scheme

	Computation	Communication	
	Exponentiations	Rounds	Messages
GENERATION	$k+1$	2	2
Basic LINKING	$2n$	2	1
Optimized LINKING	$\frac{2n}{t}$	2	1

GENERATION is performed every time a join event occurs or pseudonyms have to be changed after the linking process. In order to save computational power k shared keys $skey_{s\langle l,v\rangle}$ should be computed and stored by M_s after any dynamic event that

causes GENERATION protocol to start. Each member has to perform k group exponentiations to compute all secret shares, and one more to compute her public share. Public shares and pseudonyms should be sent in two separate messages, because the latter requires sender anonymity. Table 4 shows costs of both LINKING protocols. The basic protocol from Fig. 8 requires $2n$ exponentiations, done by every of k participating members. Every member $M_{K_{\langle l,v\rangle}} \in \mathcal{K}$ first computes n hidden secret shares $r_{\langle l,v\rangle i}$, $1 \leq i \leq n$, that require two exponentiations each, and then sends them combined in one message. Every $M_j \in \mathcal{M}$ receives these k messages from members of $\mathcal{K}$ and computes the product R_i. The optimized protocol from Fig. 9 requires $\frac{2n}{t}$ exponentiations, where $1 \leq t \leq n$. Every of t members in $\mathcal{M}_{K_{\langle l,v\rangle}} \subseteq \mathcal{K}$ computes only $\frac{n}{t}$ hidden secret shares and sends them combined in one message. Every $M_j \in \mathcal{M}$ receives these $k \cdot t$ messages from members of $\mathcal{K}$ and computes the product R_i.

The computation costs of the GENERATION protocol vary upon choosing appropriate values for k, according to power limitations of the mobile devices. Because $\mathcal{T}$ is a binary tree, k may only be chosen as a power of two.

6 Conclusion

This paper contains two different contributions concerning the usage of the computation and communication efficient iterative Diffie-Hellman (IDH) protocol for dynamic group key agreement in mobile ad-hoc networks that we have shown on the example of the Tree-based Group Diffie-Hellman (TGDH) protocol suite [7]. The first main result is the proposed communication efficient pseudonym generation scheme that allows members to generate pseudonyms without trusting in any third parties. This scheme can be used in different mobile scenarios, like in director board meetings or spontaneous auctions. Another major contribution is the proposed method of optimization by choosing an appropriate level of the IDH key tree with respect to power limitations of involved mobile devices. This method can be used separately from the pseudonym generation scheme for optimization in other security schemes that are based on threshold revocation.

References

1. N. Bansal and Z. Liu. Capacity, delay and mobility in wireless ad-hoc networks. In *Infocom 2003 (IEEE)*, 2003.
2. C. Becker and U. Wille. Communication complexity of group key distribution. In *ACM Conference on Computer and Communications Security*. ACM Press, NY, November 1998.
3. M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology (EUROCRYPT '94),Lecture Notes in Computer Science*, volume 950, pages 275–286. Springer-Verlag Berlin, May 1994.
4. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology, Lecture Notes in Computer Science*, volume 1294, pages 410–424. Springer-Verlag, 1997.

5. Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *ACM Conference on Computer and Communications Security '00*, pages 235–244. ACM Press, NY, 2000.
6. Y. Kim, A. Perrig, and G. Tsudik. Communication-efficient group key agreement. In *Information Systems Security, Proc. of the 17th International Information Security Conference, IFIP SEC'01*, 2001.
7. Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Transactions on Information and System Security*, 7(1):60–96, 2004.
8. M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(8), 2000.

A Proving Public Shares for Pseudonym Generation Scheme

After GENERATION protocol is completed every member M_s knows the set of signed public shares $\mathcal{S} = \{S_i | \ 1 \leq i \leq n\}$ and the set of broadcasted pseudonyms $\mathcal{PS} = \{ps_i | \ 1 \leq i \leq n\}$. Only M_s knows the corresponding link ($ps_s \leftrightarrow S_s$). In order to provide the authenticity of broadcasted pseudonyms, for every $ps_i \in \mathcal{PS}$ it has to be shown that it was used to compute one of known public shares $S_i \in \mathcal{S}$ without revealing that share.

Protocol in Fig. 10 is a proof for this relation. The idea behind the proof is that prover P makes all known public shares $S_i \in \mathcal{S}$ indistinguishable by building corresponding blinded public shares $S'_i = (S_i)^{r^{ps_p}}$ using randomly chosen value r and broadcasting them as a random permutation.

During one run of the protocol P proves depending on the challenge of the verifier V with the probability of $\frac{1}{2}$ either that all S'_i were built as described above using all known S_i or that she knows how a certain pseudonym ps_p relates to one of the broadcasted S'_i. The role of the prover should be taken by a member M_s, that knows the relation between her pseudonym ps_s and public share S_s. The role of the verifier V should be taken separately by all existing group members except for M_s. All verifiers have to agree on the same challenge that P has to respond to, or P has to prove her knowledge to every verifier in a separate protocol run. All messages that P broadcasts are signed using her pseudonym ps_s. Function $\pi : \ \mathbb{N} \rightarrow \mathbb{N}$ specifies a random permutation. The protocol uses zero-knowledge proof techniques $SKROOTLOG$ and $SKLOGLOG$ described by Camenisch *et. al.* in [4]. $SKROOTLOG[\alpha : \ y = g^{\alpha^e}]$ proves the knowledge of an e-th root of the discrete logarithm of y to the base g, where e, g and y are commonly known values. $SKLOGLOG[\alpha : \ y = g^{(a^\alpha)}]$ proves the knowledge of a double discrete logarithm of y to the bases g and a, where a, g and y are commonly known values.

Conjecture 1. *Protocol in Figure 10 fulfils requirements on completeness and soundness.*

Sketch of Proof. Completeness: We consider that both P and V are honest. The completeness of the protocol is obvious for case $c = 0$, since r^{ps_s} was used in step 1 as exponent to compute blinded public shares S'_i in the broadcasted permutation. Therefore, in step 4 the verifier computes the correct values. In order to show the correctness for the case $c = 1$ we examine first zero-knowledge proofs given by values $\mathcal{Z}_1$ and $\mathcal{Z}_2$.

1. P chooses random $r \in_{\mathcal{R}} \mathcal{G}$, computes
$$\{S'_1, \ldots, S'_n\} = \{S_1^{r^{ps_s}}, \ldots, S_n^{r^{ps_s}}\}$$
and broadcasts $\mathcal{S}' = \{S'_{\pi(1)}, \ldots, S'_{\pi(n)}\}$.
2. V broadcasts chosen challenge $c \in_{\mathcal{R}} \{0, 1\}$.
3. Case $c = 0$: P broadcasts
$$\mathcal{X}_0 = r$$
Case $c = 1$:
   - P computes
$$\mathcal{X}_1 = g^{r^{ps_s}},\ \mathcal{X}_2 = \{x_{2_v} |\, x_{2_v} = \mathcal{X}_1^{\left(g^{K_{\langle l,v\rangle}{}^{ps_s}}\right)^{K_s}}, 0 \le v \le 2^l - 1\}$$
$$\mathcal{Z}_1 = SKROOTLOG[r : \mathcal{X}_1 = g^{r^{ps_s}}]$$
$$\mathcal{Z}_2 = \{z_{2_v} |\, z_{2_v} = SKLOGLOG[K_s : x_{2_v} = \mathcal{X}_1^{\left(g^{K_{\langle l,v\rangle}{}^{ps_s}}\right)^{K_s}}], \forall x_{2_v} \in \mathcal{X}_2\}$$
   - broadcasts $(\mathcal{X}_1, \mathcal{X}_2, \mathcal{Z}_1, \mathcal{Z}_2)$
4. Case $c = 0$: V checks
$$\{S_1^{\mathcal{X}_0^{ps_s}}, \ldots, S_n^{\mathcal{X}_0^{ps_s}}\} \stackrel{?}{=} \mathcal{S}'$$
Case $c = 1$:
   - V checks the correctness of $\mathcal{Z}_1$ and computes
$$\mathcal{X}_3 = \{x_{3_v} |\, x_{3_v} = g^{K_{\langle l,v\rangle}{}^{ps_s}}, 0 \le v \le 2^l - 1\}$$
   - checks the correctness of every $z_{2_v} \in \mathcal{Z}_2$
   - checks
$$\prod_{x_{2_v} \in \mathcal{X}_2} x_{2_v} \stackrel{?}{\in} \mathcal{S}'$$

Fig. 10. Proof of S_i in Pseudonym Generation Scheme

The correctness of $\mathcal{Z}_1$ shows that ps_s is the root of the discrete logarithm of $\mathcal{X}_1$ to the base g. Set $\mathcal{Z}_2$ consists of hidden secret shares $r_{\langle l,v\rangle s}$ that are blinded with the random value r^{ps_s}. In order to prove the correctness of each $z_{2_v} \in \mathcal{Z}_2$ verifier has to compute set $\mathcal{X}_3$ using known public keys of all nodes at level l. The correctness of $\mathcal{Z}_2$ shows that every public key $g^{K_{\langle l,v\rangle}}$ at level l was used in the computation of blinded hidden secret shares. At last verifier builds a product of all values $x_{2_v} \in \mathcal{X}_2$. This product corresponds to the blinded public share S'_s of the prover as shown in the following identity:

$$\begin{aligned}
\prod_{x_{2_v} \in \mathcal{X}_2} x_{2_v} &= \prod_v \mathcal{X}_1^{\left(g^{K_{\langle l,v\rangle}{}^{ps_s}}\right)^{K_s}} \\
&= \prod_v \mathcal{X}_1^{g^{K_{\langle l,v\rangle}{}^{K_s ps_s}}} \\
&= \prod_v g^{r^{ps_s} g^{K_{\langle l,v\rangle}{}^{K_s ps_s}}} \\
&= g^{\sum_v \left(g^{r^{ps_s}} g^{K_{\langle l,v\rangle}{}^{K_s ps_s}}\right)}
\end{aligned}$$

$$= (g^{\sum_v (g^{K_{\langle l,v \rangle} K_s ps_s})})^{r^{ps_s}}$$
$$= (S_s)^{r^{ps_s}}$$
$$= S'_s$$

Soundness: We consider information-theoretical soundness of the protocol and show that a cheating prover P^* cannot convince a correct verifier V with more than an exponentially small probability. Consider P^* broadcasting a set of blinded public shares $\{S'_1, \ldots, S'_n\}$ for some chosen pseudonym ps_s. In order to respond to challenge $c = 1$ correctly P^* should be able to compute the sets $\mathcal{X}_2$ and $\mathcal{Z}_2$. This can be done if P^* knows the private key K_s used in the computation of secret shares for the corresponding public share S_s. Since P^* does not have this knowledge he can convince V only on challenge $c = 0$, that is with probability at most $\frac{1}{2}$. Hence, the probability with that P^* convinces V after σ iterations is at most $2^{-\sigma}$. □

The protocol in Figure 10 does not have a zero-knowledge property since the simulator guessing case $c = 1$ should output set ${\mathcal{S}'}_1^{SIM}$ that must be computationally indistinguishable to the set ${\mathcal{S}'}_0^{SIM}$ in case $c = 0$. This is obviously impossible unless the simulator can recompute at least one public share S_i and provide correct values for $\mathcal{X}_2$ and $\mathcal{Z}_2$.

Secure Overlay for Service Centric Wireless Sensor Networks

Hans-Joachim Hof, Erik-Oliver Blaß, and Martina Zitterbart

Institut of Telematics, University of Karlsruhe
{hof, blass, zit}@tm.uka.de

Abstract. Sensor networks consist of a potentially huge number of very small and resource limited self-organizing devices. Those devices offer different services and use services provided by other sensor nodes. To give sensor nodes the possibility to offer services and to network-wide search for available services, some kind of lookup facility is needed. Several possibilities exist to realize service lookup in traditional networks and ad-hoc networks [ALM03, GOL99, GUT99, PRE02, SAL99, ZHU03]. In this paper we present Secure Content Addressable Networks Version 2 (SCANv2), a secure overlay focusing especially on wireless sensor networks. The paper describes how this secure overlay can be used among other things to offer lookup functionality in sensor networks. The design of the overlay focuses on secure service lookups. The overlay is part of the Karlsruhe Sensor Network Platform K-SNeP, a modular and flexible architecture for service centric sensor networks. Key areas of application of the architecture are gradually extendable service centric sensor networks where sensors and actuators jointly perform various user defined tasks, e.g. in the field of an office environment or health care.

1 Introduction

As computer miniaturisation continues and small devices become cheaper and cheaper, more and more computers are embedded into the user's environments to offer pervasive services which enhance the surrounding of the user with intelligence. Sensor networks are special occurrences of networks formed by those small devices. The nodes of those networks have low-power, are self-organising and have little computation capabilities. Such devices typically use "peanut CPUs" [STA02]. Sensor nodes usually offer services (for example the data which they acquire) and use services provided by other sensor nodes to construct even more complex services (for example a services which controls an actuator based on the output of a sensor). The goal of sensor networks is to benefit of synergy effects of a huge number of sensor nodes in a network which has a high node density.

Security issues are very important in sensor networks as sensors invade the personal environment of a user and at the same time are harder to recognise ("disappearing computer"). In many scenarios, like assisted living or health care, sensor networks have critical functions. Therefore if sensor networks should ever be widely used, it is crucial that they have high security standards.

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 125 – 138, 2005.

Services are the building blocks of the functionality of any sensor network. Secure access to services and secure and robust service lookups are therefore vitally important for the security of the whole network.

Given the limited resources (low memory, little CPU power, low bandwidth) and other characteristics of sensor networks (maintenance-free operation, frequent node failures) it stands to reason that traditional security methods do not work satisfyingly in the context of sensor networks. Especially asymmetric cryptography seems computationally too complex for these nodes, even with latest techniques like elliptic curve algorithms [KOB94]. Therefore, security architectures for sensor networks benefit from focusing on symmetric cryptography which can be used efficiently even on peanut CPUs.

This paper focuses on secure and robust service lookups and describes a secure overlay for sensor networks that is used in the "Karlsruhe Sensor Network Platform" (K-SNeP) to realise among other things a distributed service directory. The security mechanisms of the overlay take into consideration the limited resources of sensor network nodes. Therefore, only symmetric cryptography is used on sensor nodes. The key idea behind the use of an overlay for wireless sensor networks is to create a simple distributed system that easily scales with its growth, is considerably failsafe, and does not require any centralised component during normal operation. Hence, a user can sequentially add nodes to the network without bothering to provide enough capacity. A special device, the so called Master Device, is used to bring sensor nodes into the network. This is the only device which may utilise asymmetric cryptography.
A typical scenario where K-SNeP can be advantageously deployed is an intelligent office environment with dozens or even hundreds of sensors and actuators that act jointly within a sensor network to accomplish various tasks. Another scenario is health care in a hospital where wireless sensors are used to replace wires and therefore make the work of doctors easier and the patients more mobile. In this scenario, sensors may also be build into a prosthesis to control correct movement of a patient. Sensors can even be used to record life signs of a patient over a long period of time even if the patient is not in the hospital. Assisted Living is yet another scenario which benefits from the "Karlsruhe Sensor Network Platform".

The paper is structured as follows: Chapter 2 presents Content Addressable Network which is the basis of the secure overlay presented in this paper. Chapter 3 shows the context in which the virtual overlay will be used: the Karlsruhe Sensor Network Platform. Chapter 4 presents the proposed secure overlay network: Secure Content Addressable Network Version 2 (SCANv2). This chapter also describes how SCANv2 can be used to build a distributed service directory. It also presents Clustered SCANv2, a solution for heterogeneous sensor networks which enables very low power devices to uses the distributed service directory. Chapter 5 summarises the paper and gives an outlook on future work.

2 Content Addressable Networks

Overlays are an emerging issue in wired networks. Filesharing tools like the open source project emule [EMU04] use those overlays for their services. Several overlays like Chord [STO01], Pastry [ROW01], Tapestry [ZHA01], Content Addressable Network [RAT01] and Kademlia [MAY02] are available. The Secure Content

Addressable Network Version 2(SCANv2) presented in this paper is based on Content Addressable Network (CAN) which gets enhanced with security features. SCANv2 is an advancement of SCAN [HOF04]. CAN has been chosen because of its solid structure where neighbors in CAN space have a special relationship which can be easily secured. Another reason why we chose CAN is the low and constant memory overhead of CAN which is not the case with most of the other overlays. This chapter gives a short overview of CAN. Please refer to [RAT01] for more details.

CAN utilizes a d-dimensional Cartesian coordinate space on a d-torus. This virtual space will be called "CAN space" in the rest of this paper. The coordinate space is completely logical and has no relation to any physical coordinate system or to the structure of the network. CAN forms an overlay network which lies above the network layer and utilizes the communication abilities offered by it. Hence, CAN can only be functional as long as the underlying network layer is still usable. At any time, the entire CAN space is divided into zones which are administrated by the nodes participation in the CAN. Every node "owns" a distinct zone within the overall space hence such a node is called zone owner in the rest of this paper. New zones emerge from existing zones by a split considering an ordering of the dimensions. CAN realizes a distributed hash table. The virtual coordinate space is used to store *(key,value)* pairs as follows: *key* is mapped on a point *P* in CAN space using a hash function. The *(key,value)* pair is then stored on the zone owner of the zone within which *P* lies. To retrieve an entry corresponding to a key *K*, any node can apply the public hash function to map *K* on a point *P* and then retrieve the corresponding value from the zone owner of the zone in which *P* lies. The request is routed through the CAN space until it reaches the node which owns the zone including *P*. Routing is done by greedily forwarding a message to the neighbor which coordinates are closest to the destination. Each node keeps a list of neighbors that abut their own zone. This list is used for forwarding and acts as a routing table. Periodic update messages between neighbors help to recognize failed devices and abandoned zones. Please note that many different paths exist between two points in the CAN space. This means, that even if one or more of a node's neighbors crash, a node would automatically route along the next best available path. If a node loses all its neighbors, it can do an expanding ring search to get connected to the CAN again. CAN describes some repair mechanisms for restoring a consistent state. Those are of no interest for this paper and will get careful attention in further work.

As mentioned earlier CAN space is partitioned among a number of nodes. If a new node decides to participate in CAN an existing zone is split into half and the joining device gets one of the two resulting zones. A uniform distribution of join points is eligible to maintain zones of equal size which is important because effective routing depends on a similar zone size of all zones. CAN offers some optimizations of routing and robustness. Routing can be improved using multiple CAN spaces (called multiple realities) with different hash functions on each node. Reaching a point in CAN then translates to reaching this point in any reality. Robustness can be improved by zone overloading. This means that multiple nodes own one zone and all owners are permitted to issue answers for requests on this zone. If one owner fails, there are still other zone owners which may answer requests.

3 Karlsruhe Sensor Network Platform

The proposed secure overlay is embedded in the "Karlsruhe Sensor Network Platform" (K-SNeP) which is a flexible and general architecture for wireless sensor networks. K-SNeP was designed for a special occurrence of sensor networks: service-centric sensor network. Service-centric sensor networks focus on services and realization of services in a network. In service-centric sensor networks, data flows between sensors and actuators, one delivering data and the other executing control tasks. Both, data delivery and execution of tasks may be expressed as a service. Service-centric sensor networks are therefore contradictory to traditional data-centric sensor networks where the data flow is between a huge number of sensors and a small number (typically only one) of control stations. Data-centric sensor networks are typically used for environmental observation projects, for example Great Duck Island [GRE04].

Figure 1 gives an overview of the "Karlsruhe Sensor Network Platform". In the following, its modules will be described in detail.

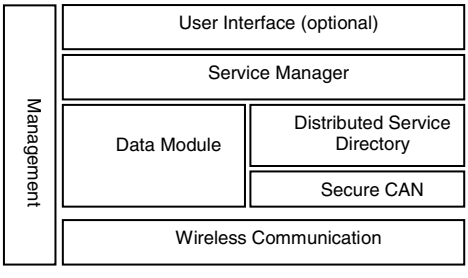


Fig. 1. Karlsruhe Sensor Network Platform

3.1 Distributed Service Directory

As explained above, services are the building blocks of a sensor network. Complex services rely on the output of other services in the network and use services on actuators to execute control tasks. A flexible use of services can only be possible if a robust lookup functionality is present in the network. The Service Directory module implements this functionality. The interface and administrational functions of the Service Directory are independent of the concrete implementation. At the moment, K-SNeP may use one of two methods: In a totally decentralised scenario, the SCANv2 presented in this paper is used. In the presence of gateways on the other hand the functionality of the Distributed Service Directory will either be implemented on those gateways with help of services in the network to which the gateway connects or, in a more general case, a clustered SCANv2 (see below) can be used where gateways are clusterheads. The service directory is used to store service records holding attributes of services which are available in the network. Attributes include, but are not limited to, address of service provider, address of data replication point, physical location of service, validity of service, quality category, needed input to provide service, output

format etc. Attributes are service-dependant. A special query language allows the user to specify the attributes the sought-after service should have.

To deal with node failure it is necessary to demand constant refresh of service entries. Services can also have a limited lifetime after which their service records are removed from the service directory.

3.2 Secure Content Addressable Network

The Secure CAN (SCAN) module implements the secure overlay described in this paper. It realises a secure and robust distributed hash table. A Content Addressable Network (CAN) is used as base of the secure hash table and gets enhanced with security features. See chapter 2 for more information on CAN and chapter 4 for details on SCANv2.

3.3 Service Manager

The Service Manager pairs actuators and sensors to execute a user task. It supervises service execution. The pairing of actuators and sensors could be temporary or permanent, access could be simultaneous, competitive or exclusive. The Service Manager Module is also responsible to determine the sensor node on which a task should be executed. Services are described using a service description language and may be executed on extern nodes. The Service Manager defines and registers new complex services which are based on other services. The Service Manager is responsible for registering services in the Service Directory. It uses the Service Directory to find the needed services. As services may require results of other services as input, the Service Manager Module may need to do cascading lookups in the service directory. Please note that the Service Manager has only to do lookups at the time of pairing. Later on, the paired actuators and sensors do not need any more service lookups as they know each other.

3.4 Management

The Management Module realises all those functions which need access to all layers or provide information needed on all layers. The Management Module includes software updates and localisation. Software updates are vital for unattended sensor networks. They allow error correction, adaptation to unforeseen environmental conditions and recalibration of sensors. Location information can be retrieved using different methods on different layers. Nearly all layers need information about the position of the node.

3.5 Data Module

The Data Module is responsible for service-centric data aggregation, data replication and, on sensor nodes, for data acquisition.

The basic idea of service-centric data aggregation is that some requester defines an aggregation rule and finds an aggregation point, where the aggregation should take place. Aggregation is offered as a service and sensor nodes capable of executing aggregations register themselves and can be found by other nodes using the distributed service directory. The requesting sensor node defines the aggregation rule

and a set of sensors which should be used for the aggregation. The aggregating node executes the aggregation rule repeatedly and registers in the service directory a new service which offers the aggregated data. Other nodes which need the same aggregated data may now use this service. If the aggregation rule is stored somewhere in the network, for example using the overlay presented in this paper, a node can request the sensor data and the aggregated values and verify the aggregation thus making it more secure as cheating may be easily noticed.

Data replication is used to cache frequently used data pieces. This can result in energy saving, if sensor values have a certain lifetime for which the sensor would acquire always the same or very similar data. Data replication may also be useful in data aggregation to deal with timing problems if sensor values are not acquired synchronously. Therefore, in the data module the data replication module is directly connected to the data aggregation module.

Data Acquisition is initiated by the Service Manager Module on sensors. Data acquisition may be initialised by another sensor node (pull) or may be executed based on an intern scheduler and published in regular periods to other sensor nodes (push)

3.6 User Interface

The User Interface enables the user to communicate with the sensor network. It gives her the possibility to issue commands to the sensor network and it provides a nice way to get results back. The User Interface is accountable for execution of commands and returns (aggregated) overall results of quests given to the sensor network. It may represent a gateway between the user's network (like the Internet) and an off-site sensor network. The User Interface may also be used for security issues. For example hardware of our testbed (called BlueEgg) has a build-in wheel to enter number sequences which can be used for authentication between two devices.

3.7 Wireless Communication

The proposed architecture requires some underlying wireless communication ability. Our testbed uses Bluetooth as communication layer. However, the architecture itself is independent of the actual used communication technology and can be implemented on most of the recently available sensor network platforms.

4 Secure Content Addressable Networks

The proposed protocol for Secure Content Addressable Networks is based on Content Addressable Networks presented in Chapter 2. However, CAN has some security flaws:

An obvious attack on a distributed hash table is to overtake a certain part of the hash table. In CAN, this can easily be done if an attacker can use an arbitrary join point. If an attacker would like to overtake a certain hash value h1 which lies in the zone (x1,y1,x2,y2) it would choose h1 as its join point. The owner of the zone would split its zone and hand over one part of the zone to the attacker (it is not necessarily the zone where h1 lies in). This can be done multiple times until the attacker gets the zone in which h1 lies.

Another attack may be performed if it is possible to claim to own an arbitrary zone of CAN. An attacker may than claim to own the whole CAN space or a huge part of it and therefore draw joining nodes into a fake network.

Communication between neighbors in CAN space ("one hop") usually takes multiple hops on network layer. If neighbors want to communicate securely, for example to achieve a common purpose like isolating an attacker, neighbors need to have a symmetric key in common. Some information like the continuous update messages need not be secret but authenticated and integrity checked.

Those issues can be solved with an extension to CAN: Secure Content Addressable Networks (SCAN). The basic idea is to use a Master Device as trust anchor for the virtual overlay. The Master Device is not part of the sensor network itself. It is a device more powerful than the typically sensor node and has especially the ability to perform public key operations. However, the Master Device is designed to have only one hard-wired key and no other state information. As the Master Device has no state information which it collects during the lifetime of the sensor network, the Master Device can be easily replaced when lost or damaged. It may also be duplicated to give multiple users the possibility to bring new nodes into the network. The Master Device is only present when a new node joins the sensor network and is under total control of the user. The user authenticates the joining device by the use of a location limited channel [BAL02], e.g. physical contact or infrared. For an actual implementation of SCAN, the master device should be a small gizmo which the user can carry with him all the time. [MAX04] is perfect for our implementation. Given the properties of the Master Device, the first security flaw can be eliminated by letting the Master Device select the join point and ensuring, that no other join point can be used. To protect against the second security flaw, the Master Device issues zone certificates during the join of a node. Those certificates get checked later by the Master Device when another node joins. The Master Device is used to establish a common secret between neighbors.

The first version of Secure Content Addressable Network was presented in [HOF04]. However, this version has some problems:

First of all, there is no way for the Master Device to notice attacks. More specific, there is no feedback during the join of a new device.

Secondly, the Master Device must be able to order other nodes to act upon a possible attack.

Thirdly, in the first SCAN proposal an attacker had the possibility to use old and no longer valid zone certificates. This is especially a problem when the hardware is not tamper-proof because in this case, it is likely that it would be possible to reconstruct an old zone certificate from the memory of an attacked sensor nodes.

Fourthly, in the first version of SCAN , symmetric keys are constructed on sensor nodes. However, those nodes are likely to have no good random number generator and the keys may therefore be insecure.

To deal with all those problems we present in the following a second version of the Secure Content Addressable Network protocol called SCANv2.

4.1 Assumptions on the Attacker

Sensor networks typically cover environments where an attacker has physical access to sensor nodes. The sensor nodes are naturally not supervised in an public

environment. Sensor networks should consist of a large number of very cheap devices. Therefore we do not expect sensors to be tamper-proof.

Considering those properties of a sensor network, an attacker can remove any sensor from the network and impersonate any sensor in the network if he has physical access. An attacker can also clone sensor nodes and add fake nodes to the network at his will. However, he can not add nodes to SCANv2 as the join process includes an interaction with the Master Device.

An attacker may eavesdrop any local communication.

There may be special devices which can not be tampered with. In our case, the Master Device is considered tamper-proof and as it is under surveillance of the user (it may be a key fob which the user carries with him all the time), chances are low that an attacker gets access to this device.

An attacker can not easily distinguish the physical position of a node knowing only the administrated zone in the SCANv2 as the structure of the overlay is unrelated of the actual network structure.

We expect it impossible to attack a sensor node before it is integrated in the sensor network by the user. The sensor node may be sealed by the vendor and the vendor guarantees that the sensor has not been infiltrated at the point of sealing.

4.2 Secure Construction and Preservation of Structure in SCANv2

The description of the proposed protocol uses the following syntax:

Channel | A -> B: Message

Channel denotes the channel a message is send in. In the present case, this can be the location limited channel, for example physical contact (*PHY*), the main communication channel like Bluetooth Scatternets (*MC*) or the CAN overlay (*CAN*). *A* denotes the sender and *B* denotes the receiver of *Message*. In the case under consideration, sender and receiver can be Master Device (*MD*), Joining Device (*JD*), owner of the zone *JD* wants to join into (*ZO*) and neighbors of *ZO* (N_i).

The Protocol consists of ten steps:

1.) PHY | MD -> JD: JP, E_{SK}*(JP,"temp")*

First, the Master Device sends *JD* over the Location Limited Channel (physical contact in this case) the join point (*JP*) at which JD will join the CAN. This point is selected randomly by the Master Device. Random selection is very important to achieve a equal distribution of zone sizes in CAN to ensure that routing in the CAN is ideal. It also ensures that it is not possible to overtake a specific part of the distributed hash table by a specially chosen join point. The Master Device also sends the join point encrypted with its super key (*SK*), which is the private key of the Master Device. The encrypted join point is used as a shared symmetric key between the node and the Master Device. It will be denoted with *k* in the further description of the protocol. The string "temp" indicates the temporary character of the key. It is important to always ensure that the join point of a device lies in its owned zone and

there are cases where it is necessary for a joining device to get another join point in a later step to ensure this characteristic.

2.) CAN | MD -> ZO: "who is responsible for JP", Address

Next, the Master Device sends a message into the CAN to find out, who owns the zone *JP* lies in. This message is not broadcasted but send in CAN space to the address *JP*. The message will therefore reach the zone owner which answers immediately. The message also includes the network layer address of the Master Device to enable the receiver to communicate with the Master Device outside the CAN on network layer. This is done for performance reasons as communication on network layer is more efficient than communication in CAN space. As stated earlier, it is not necessary for the Master Device to have the ability to communication with the network because it could otherwise use the communication abilities of the trusted joining device over the location limited channel. However, for simplification of the protocol, we assume the Master Device to be able to communication with the CAN in our further protocol description.

3.) MC | ZO -> MD: JP_{ZO},E_k(certificate, time, NeighborJPList)

The zone owner answers the request of the Master Device with its join point and the certificate of its zone (see below), encrypted with the key shared between the Master Device and the zone owner. The encrypted message also contains a timestamp to prevent reply attacks. The Master Device derives *k* using its private key as described earlier. It checks the zone owners certificate. The size of the owned zone is included in the certificate. The message from the Zone Owner also includes a list of join points of the neighbors of the zone owner in the formate $((n_1,JP_1),(n_2,JP_2),..., (n_d,JP_d))$ where n_i is the network layer address of a neighbor and JP_i the join point of the corresponding node. This is a major chance in SCANv2 as the knowledge of the neighbors join points and network layer addresses enable the Master Device to communicate with the neighbors of the zone owner. The join points are used to derive the keys the Master Device has in common with the neighbors. A feedback mechanism can now be constructed to make the Master Device aware of possible attacks during a join operation and the Master Device has a way to give orders to the neighbors of the zone owner.

4.) PHY | MD -> JD: JP, E_{SK}(JP,"perm")

The Master Device evaluates if the joining node needs a new join point taking in consideration the join point of the zone owner and the future split line in CAN. The goal is to ensure that the join point of a zone owner always lies in its zone. See chapter 2 for a description of zone splits. In all cases, the joining device gets a new symmetric key shared with the Master Device, which is constructed in the same way as described earlier. This time, key and join point are flagged as permanent. For any further interaction of the joining node with the Master Device, only the permanent key and therefore only the permanent join point is valid. From now on the sensor node uses the symmetric key for authentication and secure communication with the master device.

5.) PHY | MD -> JD: $key_{JD,ZO}$,neighborKeyList,, E_{ZO}("JP,JD", $key_{JD,ZO}$, $certificate_{ZO}$, $E_{JD}(certificate_{JD})$, neighborInformationList)

In the next step, the Joining Device gets a symmetric key ($key_{JD,ZO}$) for secure communication with the zone owner and also a neighbor key list in the format (neighbor address, symmetric key, join point). The symmetric keys are used for secure communication between the new neighbors after the zone split. Unlike the first version of SCAN, in SCANv2 the symmetric keys are constructed on the master device because it is not ensured that the neighbors all have a good random number generator. The join points are stored in the neighbor table of the Joining Device. The Master Device also hands out a "ticket" to *JD* which enables it to initiate the zone split and to get its new zone from *ZO*. The ticket is encrypted with *ZO*'s symmetric key and includes address (*JD*) and join point of *JD*, a symmetric key for communication between *JD* and *ZO*, a certificate for *ZO*'s zone after the split operation ($certificate_{ZO}$) and a certificate for *JD*'s new zone ($certificate_{JD}$) encrypted with the symmetric key shared between the Master Device and the Joining Device. The encrypted part includes some information for the neighbors of the Zone Owner which are affected by the zone split. The neighbor information list consists of several tuples in the format (zone size of zone owners zone, symmetric key for communication with Joining Device, symmetric group key for communication with all neighbors of the zone owner, Join Point of Joining Device). These tuples are encrypted for each neighbor with the symmetric key the Master Device has in common with each neighbor. Those keys can be derived by the Master Device with the help of the neighbors join points which have been transmitted in step 3 of the protocol (see there for details).

6.) MC | JD -> ZO: E_{ZO}("JP,JD", $key_{JD,ZO}$, $certificate_{ZO}$, $E_{JD}(certificate_{JD})$, neighborInformationList)

JD hands the ticket to *ZO*. *ZO* starts the zone split.

7.) MC | ZO -> JD: E_{key}(data of hash table)

ZO starts the zone split with the transmission of data stored in the part of the distributed hash table, which will belongs to *JD*. Transfer of data is encrypted with $key_{JD,ZO}$ which *JD* and *ZO* got handed out by the Master Device. This ensures, that no intermediate node is able to alter the data stored in *ZO*'s former hash table. Encryption is important to ensure integrity of the distributed hash table. All received data is acknowledged by *JD*. Those messages are not included in our protocol discussion for simplification.

8.) MC | ZO -> JD: $E_{key}(E_{JD}(certificate_{JD}))$

When the data transfer successfully ended, *ZO* hands out *JD*'s zone certificate. With the certificate being handed over at the end of the data transfer, *ZO* could test some values it formerly stored. This does not prevent the joining device to drop all data anyway, but it makes cheating more difficult. From the moment of certificate handover forth, *ZO* is no longer responsible for the half of its former zone. *ZO*

immediately and thoroughly destroys its old zone certificate and stores the new one which was included in the last message from the Master Device. Unlike in version 1 of Secure Content Addressable Networks, in SCANv2 it is not that vital to thoroughly destroy old certificates because now the neighbors double check the zone size and report to the Master Device. Therefore, even if an attacker could reconstruct a deleted certificate, it has no value to him.

9.) $MC \mid ZO \rightarrow N_i : E_{Ni}(E(\text{zone size}, key_{JD,Ni}), JD\_Address)$

As the zone split is now official in effect, *ZO* notifies all affected neighbors about the split and transfers them the neighborhood information, which was encrypted by the Master Device and can be only decrypted by the neighbors. This message includes the claimed zone size of the zone certificate of the zone owner and a symmetric key shared between the neighbor and the Joining Device. The message from the zone owner also includes the network layer address of the Joining Device. The message is encrypted with the symmetric key *ZO* shares with each of its neighbors. The neighbors check if the zone size is correct from their local view.

10.) $MC \mid N_i \rightarrow MD: JP, E_{Ni}(ACK, certificate)$ *or* $JP, E_{Ni}(NACK, certificate)$

If the zone size is correct from the local view of a neighbor, it sends an acknowledge to the master device. If the zone size is not correct, the neighbor sends a negative acknowledge. In all cases, the zone certificate of the neighbor is included and gets checked by the master device. The master device now has the prove that something is wrong and either the zone owner or the neighbor uses an old certificate. It is the challenge of the master device to resolve this issue. This feedback mechanism is new to SCANv2 and very helpful. The MD has several possibilities to find out who is cheating:

All affected neighbors either send ACK or NACK. A majority vote can be used to find out who is cheating. The master device could also query neighbors of the neighbor to verify the neighbors zone certificate. This however results in a high overhead. The age of a certificate can be used to estimate trust in combination with the knowledge of the ordering of dimensions during a zone split and the usual proportion of zone size (for example in the two dimensional case a zone has always a proportion between x and y out of the set {1:1, 1:2 and 2:1}).

To resolve the problem, the MD randomly selects one neighbor to become new zone owner (ideally the one which owned the zone earlier) and declares this node to be the new zone owner. The MD issues a new certificate and notifies all neighbors about the change. Additionally the MD can order the neighbors to add the zone owner to their ignore list.

4.3 Node Failure

Neighbors notice node failure when the constant update messages between neighbors are missing. They inform the other neighbors about this using the symmetric group key shared among them. They then start an election process and declare one node new zone owner. The zone owner gets the correct election receipted by all neighbors, encrypted with the key each neighbor has in common with the Master Device. This

receipt also includes the (encrypted) zone certificate of the neighbor. At the next join which involves the zone owner (either as neighbor or as zone owner whose zone gets split) the Master Device checks the receipts and issues a new certificate. This process again needs communication with the neighbors of the zone owner to ensure that the claimed zone size is the actual zone size.

4.4 Usage of SCANv2

In the beginning of our work, the Secure Content Addressable Network Version 2 was mainly used to realise a secure distributed service directory. As stated earlier, K-SNeP is an architecture for service-centric sensor networks. A basic lookup-functionality is therefore crucial for the network to ensure, that service executors are able to find matching sensors and actuators. However, SCANv2 may also be used for other purposes.

The above described service directory can be extended to act as a security anchor for secure services. For example, a commitment to a public key may be included in a service description, similar to self-certifying path names [MAZ99]. SCANv2 may also be used by the data module: in the process of data aggregation, aggregation rules and data descriptions may be stored in SCANv2 to be accessible for everybody who wants to get aggregated data making therefore the data aggregation transparent and verifiable.

However, for all usage of SCANv2, it is important to remember that one hop in the SCANv2 space can be multiple hops on network layer. Communication in SCANv2 is therefore energy expensive and should only be used in rare cases like in the cases above, where a node typically needs only one time contact to SCANv2 and this contact is followed by a long period in which the returned data is used.

4.5 Clustered SCANv2

Sensor networks are usually very heterogeneous in performance of the sensor nodes. Some sensors may not be able to perform all the task related with SCANv2. Because SCANv2 as basis of the distributed service directory is vital for service provisioning and usage of services, even those devices need to have a way to get information out of SCANv2. We propose to cluster those low-power devices around a more powerful sensor node (called cluster head) which provides all the information the cluster member needs. We expect that there are enough powerful sensor nodes regarding the complex tasks some sensors fulfil. Communication in clusters is usually one-hop communication. A hash-chain or a symmetric key is used to secure communication between cluster head and cluster members. Security is also set up by the Master Device. It may for example issue an up-to-date hash value to the joining node or hand it a symmetric key for communication with the cluster head. The Master Device could either hand out a personal symmetric key or a cluster key which is used in the whole cluster. A special case of a clustered SCANv2 is a scenarios where gateways are present which connect to other networks. There, gateways are cluster heads and the only SCANv2 members. Another special case is a clustered SCANv2 where the task of cluster head and therefore SCANv2 member changes from time to time between the mass of cluster members. This may be done for energy loss balancing between a group of sensors.

4.6 Redundancy and Load Balancing

The distributed service directory uses SCANv2 to store service records under the hash of the name of the service. This can be a problem if there is a huge amount of sensors which offer the same service because in this case, all the service records are stored in the same zone (on the same node). This uses resources of just one node and makes this node to a single point of failure and therefore a good attack target. The original CAN paper suggests to use "zone overloading" or "multiple hash functions". With zone overloading active, one zone may be owned by more than one node. Multiple hash functions can be used to calculate more than one hash value under which service records are stored. Both methods add redundancy but they distribute exactly the same amount of data on a higher number of nodes. However, communication will be less on the individual nodes. As communication consumes the main part of the energy reserves of a sensor node, both methods are used in K-SNeP. To achieve a better distribution of hash values for a service, each service may use an arbitrary but well-known parameter which is added to the service name before hashing. Typically, a discreet and often-used parameter is used because this makes searches more efficient. For example if a sensor network is used in an office environment and there is a special service which is used in context of a room, the room number may be appended to the service name thus distributing load and making searches more efficient as no compare operation is needed in the distributed service directory module on the zone owner.

5 Summary and Outlook

This paper presented a secure overlay for wireless sensor networks and how it can be used to realise a distributed service directory. The overlay is embedded in the Karlsruhe Sensor Network Platform for which this paper gave a work-in-progress status. The paper showed how common secrets are distributed to SCANv2 neighbors during the join of a new device and how zone certificates are used to prevent a node from claiming ownership of an arbitrary zone size. No asymmetric cryptography is used on any sensor network node. The paper presented a feedback mechanism to deal with problems during the join. It also presented how a group key is established between neighbors of a zone.

CAN in its basic implementation does not take into consideration the location of devices. This means, that eventually, communication of two physical neighbors takes plenty of hops in CAN. There are some enhancements of the basic proposal which we plan to integrate into our protocol.

Further research is needed to use SCANv2 and the service directory as trust anchors for secure services.

References

ALM03 Florina Almenárez, Celeste Campo: "SPDP: A Secure Service Discovery Pro-tocol for Ad-hoc Networks", 9th Open European Summer School and I-FIP Workshop on Next Generation Networks, Budapest, Ungarn, 2003

BAL02 Dirk Balfanz, D. K. Smetters, Paul Stewart and H. Chi Wong: "Talking To Strangers: Authentication in Ad-Hoc Wireless Networks", Symposium on Network and Distributed Systems Security (NDSS'02), Xerox Palo Alto Research Center, Palo Alto, USA, 2002

EMU04 http://www.emule-project.net/ , accessed on 09.03.2004

GRE04 http://www.greatduckisland.net/, accessed on 09.03.2004

GOL99 Y. Goland, T. Cai, P. Leach, Y. Gu, and S. Albright: "Simple service discovery Protocol/1.0 operationg without an arbiter" , Internet Draft, IETF, 1999

GUT99 E. Guttman, C. Perkins, J. Veizades, M. Day: „Service Location Protocol, Version 2", IETF RFC2608, 1999

HOF04 Hans-Joachim Hof, Erik-Oliver Blaß, Thomas Furhmann, Martina Zitterbart: "Design of a Secure Distributed Service Directory for Wireless Sensornetworks", First European Workshop on Wireless Sensor Networks (EWSN), Berlin, 2004

KOB94 Neal Koblitz, "A course in number theory and cryptography, 2nd edition", Springer Verlag, Berlin, 1994

MAX04 Maxime: "Java-Powered Cryptographic iButton", http://www.ibutton.com/ ibuttons/java.html, accessed on 09.03.2004

MAY02 Petar Maymounkov and David Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric", In Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, March 2002

MAZ99 David Mazières, Michael Kaminsky, M. Frans Kaashoek and Emmett Witchel: "Separating key management from file system security", in 17th Symposium on Operating Systems Principles (SOSP'99), Kiawah Island, 1999

PRE02 Stephan Preuß: "JESA Service Discovery Protocol: Efficient Service Discove-ry in Ad-Hoc Networks", 2nd International IFIP-TC6 Networking Conference, Pisa, Italien, 2002

RAT01 Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp and Scott Shenker, "A Scalable Content-Addressable Network", In Proceedings of ACM SIGCOMM 2001, August 2001

ROW01 Antony Rowstron and Peter Druschel: "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems", 18th Conference on Distributed Systems Platforms, Heidelberg, Germany, 2001

SAL99 The Salutation Consortium: "Salutation Architecture Specification Version 2.0c", http://www.salutation.org/spec/Sa20e1a21.pdf, 1999

STA02 Frank Stajano, "Security for ubiquitous computing", John Wiley & Sons, West Sussex, England, 2002

STO01 Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan: "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", Technical Report TR-819, Massachusetts Institute of Technology, Cambridge, USA, March 2001

ZHA01 B.Y. Zhao, K.D. Kubiatowicz and A.D. Joseph: „Tapestry: An Infrastructure for Fault-Resilient Wide-Area Location and Routing", Technical Report UCB//CSD-01-1141, Computer Science Division, U. C. Berkeley, Berkeley, USA, April 2001

ZHU03 F. Zhu, M. Mutka, L. Ni: "Splendor: A secure, private and location-aware service discovery protocol supporting mobile services", 1st IEEE International Conference on Pervasive Computing and Communications, 2003

IKE in Ad-Hoc IP Networking

Kaisa Nyberg

Nokia Research Center, Finland
kaisa.nyberg@nokia.com

Abstract. As the Internet Protocol (IP) is becoming the ubiquitous networking protocol, the benefits of using IP-based security technology in different networking environments become clear. IPsec is already widely exploited in different networks and terminals. It is therefore expected that also IKEv2 will be required to adapt itself to a wide range of requirements posed by different key management environments. The problem studied in this paper is, how IKEv2 could be adapted for use in IP-based ad hoc networking. As a standard authentication and key agreement method IKEv2 would be available over different radio technologies, support secure IP networking and offer solid basis for multiple applications. We conclude that the problem can be solved and demonstrate it by presenting two different protocols, which integrate ad hoc authentication methods to standard IKEv2 exchanges.

Keywords: peer-to-peer communication, ad hoc networking, ad hoc access, key management, security association, IKEv2, ad hoc authentication, message authentication code.

1 Introduction

1.1 IKEv2

The initial exchange protocol of Internet Key Exchange version 2 (IKEv2) [7] has two essentially different forms. The first one, which in this paper is called the basic form, consists of four messages, and supports scenarios where the communication parties authenticate to each other using public key certificates or strong shared secret keys. The second form makes use of the Extensible Authentication Protocol (EAP) [8] for client authentication, and potentially accommodates any EAP type. In particular, authentication of the initiator (client) can be based on a human memorable passkey by selecting such an EAP type for use. The responder (server) is authenticated using either public key methods and public key certificates, or a strong shared secret.

IKEv2 does not specify how the key management is performed, but assumes that the specified types of authentication keys are available for the initial exchanges. In particular, when an application specification tells to use IKEv2, it must also specify what is the key management procedure that is required for IKEv2 to run succesfully and securely. In most cases the complexity of key management supersedes the complexity of IKE. Verification of public key certificates requires the root key to be securely distributed to and stored in the devices. A digital signature algorithm must be implemented in the devices and a digital signature must be created and/or verified for each new IKE. If the

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 139–151, 2005.

authentication of an IKE security association (IKE SA) is based on shared secret, this secret must somehow be established for the devices, and moreover, securely stored in the devices for later use to establish an IKE SA. In particular, all types of authentication keys currently specified for IKEv2 are difficult to configure manually or using other ad-hoc means.

The Internet Protocol (IP) itself has become the ubiquitous networking protocol. With IP, also the security solutions offered by IPsec are widely exploited in various different network environments. It is therefore expected that also IKEv2 will be required to adapt itself to a wide range of requirements posed by different networking environments and devices.

1.2 Independent and Ad-Hoc Networking

The task of exploiting IKEv2 as the ubiquitous authentication and key agreement protocol becomes very challenging if key management prerequisites as required by IKE SA authentication are not in place. In this paper we are concerned with application of IKEv2 to ad-hoc networking in general, and to ad-hoc networking of small devices with limited computing power and constrained operating interfaces, in particular. In such an environment one or more of the following prerequisities are missing at least for one of the devices:

- access to a service network,
- preliminary security association for key agreement,
- secure storage, and/or
- management interface for strong keys.

Independent networking configurations are specified for IEEE 802.11 WLAN [5] and for Bluetooth [2] wireless connectivity technologies. Currently only Bluetooth has specified a key management mechanism, using which the users of two Bluetooth devices can establish a shared secret link key. This mechanism, also called pairing, makes use of a user generated passkey that is entered to a pair of Bluetooth devices and nonces exchanged by the devices over the radio link. The cryptographic algorithms are symmetric key algorithms, from where it follows that the resulting link key is at most as strong as the initial passkey. But the passkey is handled by the user and therefore adequate level of security is difficult to achieve. In Bluetooth, the standard security mechanisms are usually implemented in the Bluetooth radio module, which makes difficult to upgrade them to meet current and future security requirements.

IP networking in WLAN independent basic service set (IBSS) [5] and Bluetooth personal area network (PAN) are becoming commonplace radio technologies for ad-hoc networking. In addition to the link layer security mechanisms the security is enhanced using IPsec. Using IKE for authentication and key agreement also in ad-hoc network becomes a naturally recommended option. Having been designed for IP, IKE offers not only the basic authentication and key agreement between the parties but also a multiple of other security support for IP networking including key derivation for descendant (child) security associations.

1.3 Ad-Hoc Authentication Interfaces for IKEv2

In this paper, we describe two different approaches to open up an interface to the IKEv2 initial exchanges that would allow for ad-hoc peer-to-peer authentication. In both ap-

proaches, the responder is authenticated based on a short piece of information that the responder gives to the initiator before the IKE exchanges can start. This short piece of information is either a unkeyed hash code or a message authentication code using a shared secret key. The latter have previously been discussed and called as MANA certificates in [10]. MANA certificates were designed to minimize the length of the authentication information that is handled manually by users. In many applications about 10 decimal digits provides sufficient security. An example construction of MANA certificates is given in Section 3.

The first protocol is described in Section 4. It is based on the basic IKEv2 and has been designed to minimize computations of public key cryptography algorithms. This is achieved by authenticating the responder's Diffie-Hellman key share in advance using a MANA certificate. In such a manner, it suffices to authenticate the IKEv2 exchanges using a short shared secret. While no changes are required to the IKEv2 protocol exchanges, this approach may be considered controversial and to violate IKEv2 design principles as discussed in Section 5. Therefore, a second, more conventional (from IKEv2 point-of-view) protocol is proposed.

The second protocol is based on the extended authentication version of IKEv2. The responder authenticates itself based on a raw RSA key certificate that is verified by the initiator based on a MANA certificate or a hash code. The initiator is authenticated based on a short passkey using some EAP method. The second protocol is much more complex than the first one, since it requires that both Diffie-Hellman key exchange and RSA signatures are implemented, as well as the entire EAP authentication framework. The second protocol is described in Section 6.

Finally, in Section 7 we discuss some variations to the ad-hoc authentication methods such as replacing the MANA certificates by unkeyed hash codes. Such methods are previously widely in use for example in fingerprinting of the PGP public keys [9]. Application of hash code based authentication to various ad-hoc and proximity scenarios are presented in [1].

2 IKEv2 and Options

The initial exchanges of the basic IKEv2 as given in [7], Section 1.2, is depicted in Figure 1. The first pair of messages (IKE_SA_INIT) negotiate cryptographic algorithms, exchange nonces, and do a Diffie-Hellman exchange. The second pair of messages (IKE_AUTH) authenticate the previous messages, exchange identities and certificates, and establish the first security association (CHILD_SA).

IKEv2 offers a number of options to accommodate special needs of different use scenarios. For example, reuse of Diffie-Hellman exponentials is allowed although a health warning is given in [7], Section 2.12, that it may affect the property of "perfect forward secrecy".

The AUTH payload can be generated using two essentially different methods, either using a digital signature or a message authentication code based on a shared secret key. As noted in the IKEv2 memo, this authentication method is not secure if the shared secret is short or otherwise weak, such as a human memorable passkey. For applications using passkey-based authentication for bootstrapping an IKE security association IKEv2

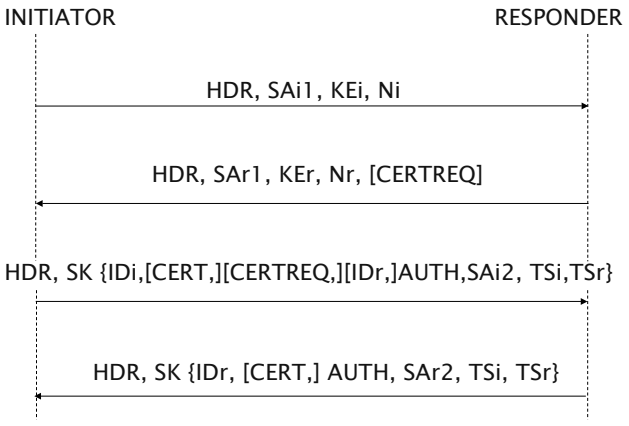


Fig. 1. Initial exchanges of IKEv2

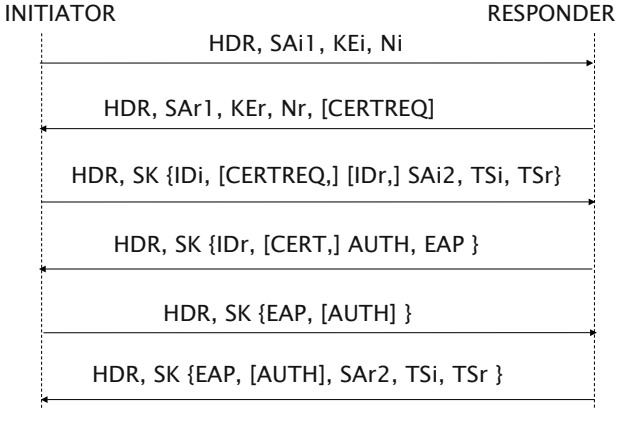


Fig. 2. Extensible authentication method in IKEv2

offers another authentication method, which is designed to prevent off-line dictionary attacks of weak passkeys. In this method, the initiator (client) is authenticated in the EAP framework. But authentication of both parties based on a short passkey is not secure within IKEv2.

The use of extensible authentication protocol in IKEv2 context is specified in [7], Section 2.16, and depicted in Figure 2. The first pair of messages are the same as in the basic protocol, see Figure 1. The IKE_AUTH exchanges are different. In the third message the client indicates by not including the AUTH payload that it wants to be authenticated using an EAP method. The next two messages indicate the EAP request and response exchanges that follow and ended by the EAP_success sent by the responder. In the first of them the responder sends its AUTH payload to allow the initiator to authenticate the responder.

The authentication method used to authenticate the responder may vary a lot. For example, in a case of an EAP method that offers mutual authentication, the initiator may even ignore the responder's CERT and AUTH fields. If responder's authentication by the initiator is based on a certificate, a number of different options are given. From the given options, the raw RSA key, with no certificate to be verified, suits best for our purposes.

3 MANA Certificates

MANA certificates consist of security related information that is authenticated using an unconditionally secure message authentication code (MAC). Such methods were first suggested to improve the security of the Bluetooth pairing procedure in [4] in combination with the Diffie-Hellman key exchange. They were further developed by the IST-SHAMAN project in [10], and are currently being standardised by ISO/IEC [6]. For a recent overview, see [3]. In ad-hoc key agreement certificates are created on-the-fly, then used once, after which they are discarded. Therefore, they need be neither human memorable nor computationally secure. But short length is an advantage when the certificate is transferred from one device to another. The transfer needs to take place over a confidential and authenticated communication channel, which is separate from the open communication channel, over which the IKEv2 protocol messages are communicated. They are often proximity-based and constrained (narrow-band) physical communications channels, and typically based on one or more of the following technologies:

- Fixed connection such as cable, USB interface, bar-code reader, smart-card reader;
- Human involvement, communication of passkeys, entering passkeys; and/or
- Other proximity based technology (low power channel)

The use of physical security technologies requires some human involvement. For better security, as well as for user convenience, it is desirable to minimise the human involvement, and to make it as easy and robust as possible.

Let D be some security related data generated by one device (A) and sent to a second device (B) at some later point. Then the MANA certificates are generated and handled as follows:

1. Device A generates a random key K, where K is suitable for use with a MAC generation function shared by the two components. Using K, device A computes a MAC code C as a function of data D. The MANA certificate consists of the pair $K||C$. The certificate is then given to the user by the output interface of A. The user now reads $K||C$ from the output of A.
2. The user enters $K||C$ to the second component using the input interface of device B. The $K||C$ value is stored in B.
3. When B at some later time receives data D, it can verify the authenticity of the data using the stored value of the MANA certificate $K||C$. Device B uses the key K to recompute the C value as a function of the received data D. If the two MAC values agree then B accepts D and outputs a success signal to the user. Otherwise it gives a failure signal.

An unconditionally secure MAC is used only once or a small number of times. Then its security can be quantified using the probability of forging the MAC. There exist two types of attacks, substitution attacks and impersonation attacks. The MANA certificates are kept confidential, which implies that the substitution attack is not possible. The only attack that remains is to guess the MAC correctly in one or a small number of trials.

A typical example of an unconditionally secure MAC function can be constructed using a Reed-Solomon code as proposed by the SHAMAN project [10]. The forgery probabilities can be computed for this code and are given here as reference. The probabilities are optimised for equal lengths of K and C, and vary within the given interval depending on how much off-line analysis the attacker can do.

Table 1. Forgery probabilities of MANA certificates using Reed-Solomon codes

Length of D	Length of $K\|\|C$ in bytes	Forgery probability
128	2	$2^{-4} - 2^{-8}$
128	3	$2^{-8} - 2^{-12}$
128	4	$2^{-13} - 2^{-16}$
256	4	$2^{-12} - 2^{-16}$
128	5	$2^{-17} - 2^{-20}$
256	5	$2^{-16} - 2^{-20}$

The forgery probabilities depend also on the length of D. For longer data the recommended practise is to hash the data first using a computationally secure hash function. It can be seen from the table that the probability decreases rapidly as the length of K and C increase. For K and C one byte each, the forgery probability is at most 2^{-4}. For many applications, MANA certificates with length of four bytes provide reasonable security level. Then the forgery probability is at most 2^{-12}. Transformed to decimal digits, which is an appropriate form of handling data over user interfaces, the length of MANA certificate $K||C$, as displayed to the user of the responder device, is 10 decimal digits.

4 Protocol I

In Protocol I four separate phases can be identified:

1. Pre-authentication
2. IKE_SA_INIT exchanges
3. MANA verification
4. IKE_AUTH exchanges

The same phases can be identified also in Protocol II, Section 6.

Phases 2 and 4 comprise of the standard IKE exchanges that are transmitted over the insecure connection between the initiator and the responder. Phase 1 makes use of some authenticated and confidential channel which is available online or offline in beforehand, while in phase 3 no communication takes place.

Authentication of the parties in the basic IKEv2 is based on public key methods (digital signatures) or a shared secret key. The CERT and CERTREQ fields can be omitted if authentication is based on the shared secret key. Protocol I makes use of the shared secret key option for authentication. Moreover, the shared secret key used by IKEv2-MANA is relatively short.

The use of identities in this protocol as well as in Protocol II depend on the application environment. Sometimes they can be omitted, most often the identity can be replaced by a connection sequence number or a similar identifier known to both parties. The use of identies in the protocols described below is just one example.

Let us now describe the four phases of this protocol.

1. Pre-authentication
   The pre-authentication step is depicted in Figure 3 and the exchange makes use of a different channel than the other steps of the protocol. The idea is that the responder generates a MANA certificate and a passkey for the client, and transfers this information to the initiator using some confidential and authenticated interface. If a man-machine interface is used, then typically the information is displayed on the responder device, from where the user reads it and enters it to the initiator device.

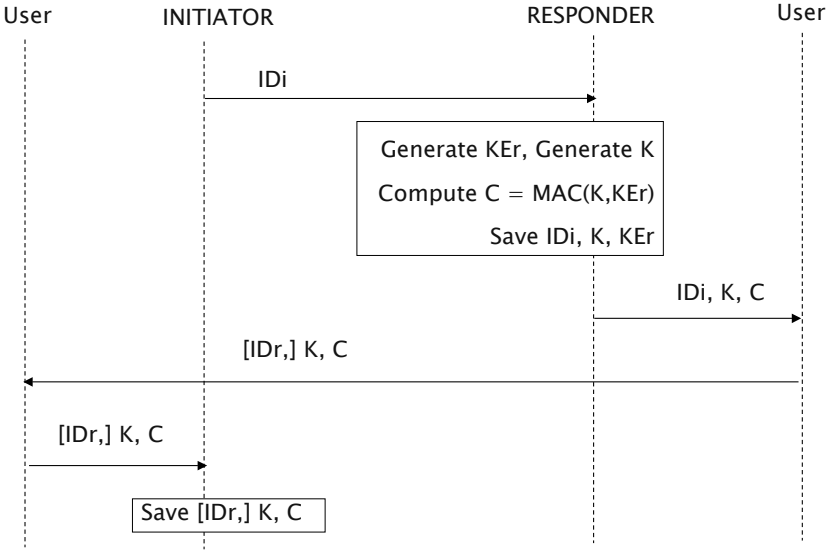


Fig. 3. Pre-authentication for Protocol I

2. IKE_SA_INIT exchanges of IKEv2
   To initialize the IKE protocol, the standard IKE_SA_INIT messages are exchanged. The optional CERTREQ field in the second message from the responder to the initiator is not used.

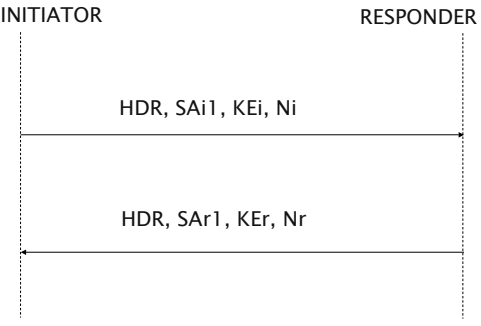


Fig. 4. IKE_SA_INIT exchanges for Protocol I

3. MANA verification
   After the initiator receives KEr from the responder, it retrieves the saved K and C, computes $C' = MAC(K, KEr)$. If $C' = C$ the initiator accepts KEr and performs the remaining steps of IKEv2. If $C' \neq C$ the initiator may retransmit the IKE_SA_INIT request a small number of times. If $C' \neq C$ repeatedly, then the establishment of the IKE security association must be interrupted.
4. IKE_AUTH exchanges
   The IKE_AUTH exchanges as specified for IKEv2 in the shared secret case are depicted in Figure 5. The AUTH value in the third and fourth exchange is computed as specified in IKEv2:

   AUTH = prf(prf(Shared Secret,"Key Pad for IKEv2"), <message octets>)

   where Shared Secret = K and the message octets are as specified in IKEv2 containing the information from the sender's previous message and the other party's nonce.

5 Rationale Behind Protocol I

According to the design principles of IKEv2 [7], Section 2.12, the Diffie-Hellman values KEi and KEr are of ephemeral nature, which reflects the fact that the IKE SA is ephemeral. On the other hand, the security associations that are used to authenticate the IKE SA are assumed to be long term and well established in beforehand. In particular, certificate creation is considered expensive in a conventional PKI and is therefore used only to certify long term public keys. Therefore, beforehand agreed certificates are not issued on Diffie-Hellman ephemeral keys.

In ad-hoc key authentication and key agreement everything is ephemeral. If the device interfaces allow transport of a strong shared secret from one device to another, and the devices can provide a secure storage for the shared secrets, then the future IKE SAs between these devices can be authenticated based on the shared secret key. However, such key agreement interfaces if they exist are not standardised, and can be used only in special environments, and are expensive. MANA certificates offers a solution using only the standard man-machine interfaces by reducing the length of the shared secret

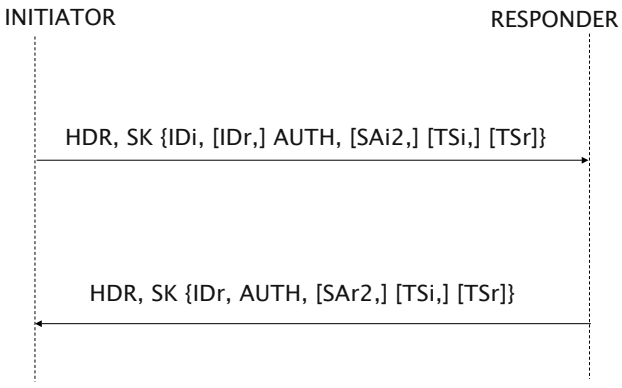


Fig. 5. IKE_AUTH exchanges for Protocol I

to a manageable size. MANA certificates are used for one key agreement as a proof of physical identification of the networking devices. The current structure of the initial exchanges of the basic IKEv2 would suggest the use of another public key mechanism to authenticate the IKE SA. This suggestion is followed in the design of Protocol II described below. But ad-hoc certificates such as MANA certificates are ephemeral, and therefore the public keys authenticated using it are also ephemeral. Therefore it means only duplication of effort and waste of resources if another public key mechanism is taken to use to authenticate the IKE SA.

In Protocol I, an ad-hoc certificate is created on the responder's Diffie-Hellman exchange value. The initiator verifies the certificate. If the verification passes, then the secret key SK is authenticated by the initiator. In this manner, a secure tunnel from the initiator to the responder has been established, through which the initiator can now securely authenticate itself to the responder using a short shared secret. This principle of secure tunnelling is well-known, and also used in IKEv2 for extensible authentication.

Protocol II to be described next follows more closely the existing structure and design principles of IKEv2. It is based on the extensible authentication mode of IKEv2, which implements the tunneling principle from the initiator to the responder. The authentication of the tunnel is mandatory in the course of the protocol. This requirement is fulfilled in Protocol II by using a raw RSA key for this purpose. Another approach would be to omit the tunnel authentication in the course of the protocol exchanges, and do as in Protocol I and create the ad-hoc certificate on the responder's Diffie-Hellman exchange value directly.

6 Protocol II

Protocol II explained in this section integrates manual authentication to the IKEv2 protocol using an Extensible Authentication Protocol for client authentication. The standard form of this protocol has (at least) six messages as depicted in Figure2.

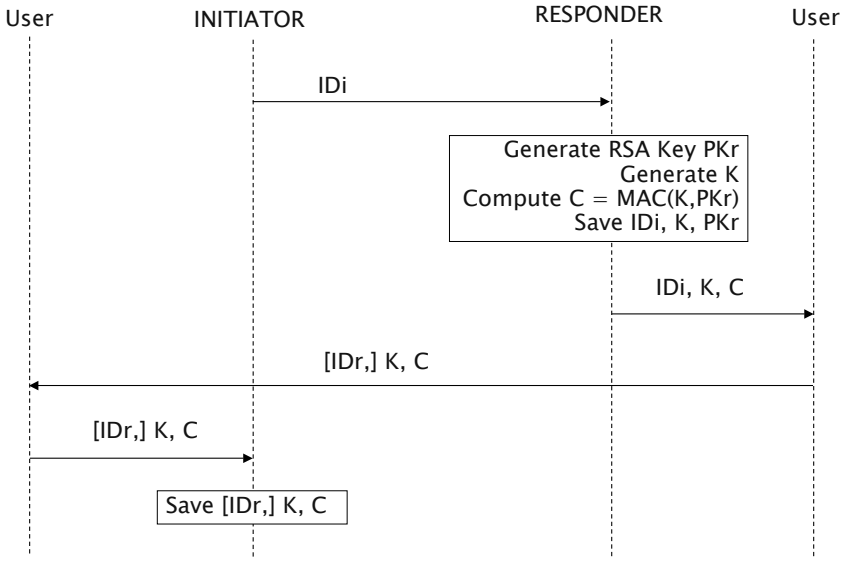


Fig. 6. Pre-authentication for Protocol II

Protocol II makes use of the "raw RSA key" option as the certificate in the fourth message. Hence the AUTH payload in the fourth message is authenticated using RSA signatures, on which the authentication of the responder is based on. The EAP method used to authenticate the initiator can be any client authentication method using a short password.

Let us now describe the four phases of this protocol.

1. Pre-authentication
   The pre-authentication step is depicted in Figure 6. The responder generates a MANA certificate and a passkey for the client, and transfers this information to the initiator using some confidential and authenticated interface. If man-machine interface is used, then typically the information is displayed on the responder device, from where the user reads it and enters it to the initiator device.
2. Initial IKE exchanges
   At this step the first four messages of the EAP authenticated IKEv2 protocol are exchanged as depicted in Figure 7. The optional CERTREQ field in the second message is not used. In the fourth message the CERT payload contains the raw RSA public key PKr of the responder.
3. MANA verification
   The initiator extracts PKr from CERT and retrieves the saved K and C. Then the initiator computes $C' = MAC(K, PKr)$. If $C' = C$ the initiator accepts PKr and performs the remaining IKE authentication steps. If $C' \neq C$ the initiator may

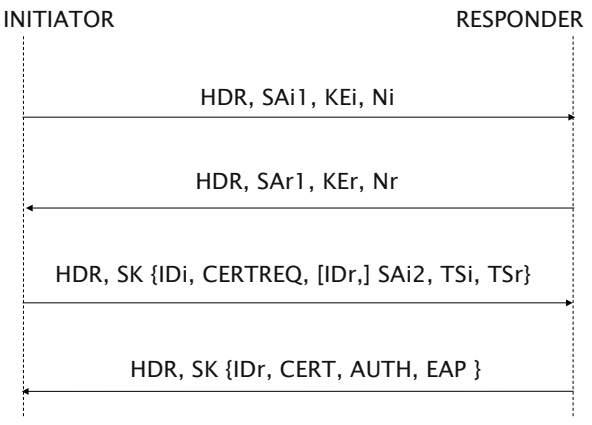


Fig. 7. Initial IKEv2 exchanges for Protocol II

retransmit the IKE_SA_INIT request a small number of times. If $C' \neq C$ repeatedly, then the establishment of the IKE security association must be interrupted.

4. EAP exchanges
   The EAP exchanges, see the last two exchanges in Figure 2, can now be transmitted. The EAP method uses the passkey K to authenticate the initiator to the responder. Hence the AUTH field is not used in the EAP exchanges sent from responder to initiator.

7 Hash Functions and Other Options

In the protocols as described above the short key K is used two different times, first to generate the MANA certificate, and later to authenticate the initiator to the responder. In some scenarios it may be desirable to use two separate short keys K_1 and K_2. Let K_1 be the key that is used to compute the ad-hoc certificate in pre-authentication, and again to verify it by the initiator. Let K_2 be the key used to authenticate the AUTH payloads and/or as the initiators passkey in the EAP method. For example, if Protocol II can be used for ad-hoc network access in such a way that the network access point is authenticated based on MANA certificate computed using K_1, and the EAP method uses some back-end server beyound the access point, then clearly the keys should be separated. In this case, the EAP passkey K_2 may be a long term passkey of the initiator.

Another option would be to use an unkeyed hash code as the ad-hoc certificate by selecting $K_1 = \emptyset$. Authentication of public keys based on computationally secure hash codes have previously become known as fingerprints in the PGP context [9]. Recently Balfanz et al. discussed a number of ad-hoc authentication scenarios using public keys authenticated by hash codes [1]. Hash code based certificates need not be ephemeral, and can be communicated using a non-confidential channel. But their inegrity and authenticity must be protected by physical means. Also they must be significantly longer than

the MAC of the MANA certificate. For example, ad-hoc authentication of an accessory device such as an printer in public area would be based on a long term hash code. But then also the public key that is authenticated using the hash code must be long term, and therefore only Protocol II is suitable in this case.

8 Conclusion

Authentication and key agreement for ad-hoc networking has been lacking a standard solution that would be available over different radio technologies, support secure IP networking and offer solid basis for multiple applications. In this paper, we selected IKEv2 as the candidate protocol and presented a number of arguments why IKE would be a good choice. The problem is that IKEv2 was not designed to be authenticated using ad-hoc means, but rather conventional pre-established security associations. Therefore we investigate possible approaches to adapt IKEv2 for the special authentication requirements of ad-hoc networking. We conclude that the problem can be solved and demonstrate it by presenting two different protocols, which integrate ad-hoc certificates to standard IKEv2 exchanges. In the first protocol the cryptographic machinery needed to establish a secure and authenticated IKE is brought to its minimum. The second protocol uses heavier machinery, and assumes that the EAP is available.

Acknowledgment

I wish to thank Pasi Eronen for reading an earlier version of this paper and suggesting the option using raw RSA key.

References

1. Dirk Balfanz, D. K. Smetters, Paul Stewart, and H. Chi Wong. Talking To Strangers: Authentication in Ad-Hoc Wireless Networks. In *Network and Distributed System Security Symposium Conference Proceedings: 2002 (NDSS 2002)*, http://www.isoc.org/isoc/conferences/ndss/02/proceedings/papers/balfan.pdf
2. Bluetooth SIG. *Specification of the Bluetooth System*, Volume 1, v 1.1 , June 2003.
3. C. Gehrmann, C. J. Mitchell and K. Nyberg. Manual authentication for wireless devices. RSA Cryptobytes, Spring 2004.
4. C. Gehrmann and K. Nyberg. Enhancements to Bluetooth baseband security. In *Proceedings of Nordsec 2001*, Nov. 1 - 2, 2001, Technical University of Denmark, Lyngby, Denmark
5. IEEE P802.11i/D10.0, IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements, April 2004
6. ISO/IEC FCD 9798-6, Information technology — Security techniques — Entity authentication — Part 6: Mechanisms using manual data transfer, ISO/IEC JTC 1/SC 27 N 3961, 2004-05-18.
7. C. Kaufman, Ed. Internet Key Exchange (IKEv2) Protocol, IETF ipsec working group draft (work in progress), Obsoletes: 2407, 2408, 2409, draft-ietf-ipsec-ikev2-13.txt, March 22, 2004 Expires: September 2004.

8. RFC 2284, PPP Extensible Authentication Protocol (EAP). L. Blunk, J. Vollbrecht. March 1998.
9. P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, Cambridge, MA, USA, 1995.
10. IST-2000-25350 SHAMAN, Security for Heterogeneous Access in Mobile Applications and Networks, Deliverable D13 Annex 2, http://www.ist-shaman.org or http://www.isrc.rhul.ac.uk/shaman/docs/d13a2v1.pdf (2003)

Advanced Detection of Selfish or Malicious Nodes in Ad Hoc Networks

Frank Kargl, Andreas Klenk, Stefan Schlott, and Michael Weber

University of Ulm, Dep. of Multimedia Computing, Ulm, Germany

Abstract. The fact that security is a critical problem when implementing mobile ad hoc networks (MANETs) is widely acknowledged. One of the different kinds of misbehavior a node may exhibit is selfishness. A selfish node wants to preserve own resources while using the services of others and consuming their resources. One way of preventing selfishness in a MANET is a detection and exclusion mechanism. In this paper, we focus on the detection phase and present different kinds of sensors that can be used to find selfish nodes. First we present simulation results that show the negative effects which selfish nodes cause in MANET. In the related work section we will analyze some of the detection mechanisms proposed in literature so far. Our new detection mechanisms described next are called *activity-based overhearing*, *iterative probing*, and *unambiguous probing*. Simulation-based analysis of these mechanisms show that they are highly effective and can reliably detect a multitude of selfish behaviors.

1 Misbehaving Nodes in Ad Hoc Networks

Mobile ad hoc networks (MANETs) rely on the cooperation of all the participating nodes. The more nodes cooperate to transfer traffic, the more powerful a MANET gets. But supporting a MANET is a cost-intensive activity for a mobile node. Detecting routes and forwarding packets consumes local CPU time, memory, network-bandwidth, and last but not least energy. Therefore there is a strong motivation for a node to deny packet forwarding to others, while at the same time using their services to deliver own data.

In table 1 we analyze different possibilities for a selfish node to save resources in a MANET based on the DSR routing protocol [1, 2]. It uses the attack-tree notation proposed by Bruce Schneier [3] that allows the categorization of attacks that all lead an attacker to reach a specific goal. Alternatives to reach this goal are denoted with OR, multiple steps that are necessary with AND. Using the numbers in the table, we can easily describe different attacks. For example, attack 3.1 stands for "Drop data packets".

Whereas most of the attacks based on manipulations of routing data can be detected by the use of a secure routing protocol like Ariadne [4], SRP [5, 6, 7, 8, 9], ARAN [10], or SAODV [11, 12], there remain two attacks in the attack tree that cannot be detected this easily. When nodes simply drop packets (case 1.1 and 3.1 in the attack tree), all of the secure routing protocols fail, as they focus only

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 152–165, 2005.

Table 1. Attack Tree: Save own resources

Attack tree: Save own resources
OR 1. Do not participate in routing
OR 1. Do not relay routing data (case A)
OR 1. Do not relay route requests
2. Do not relay route replies
3. Set hop limit or TTL value in route request/reply to smallest possible value
2. Modify routing data/topology
OR 1. Modify route request
OR 1. Insert additional hops
2. Modify route reply
OR 1. Replace own ID in returned route with detour leading through neighboring nodes
2. Return completely wrong route, provoking RERR and salvaging
3. Insert additional hops
4. Declare own ID in source route as external
2. Stop participation in current route
AND 1. Provoke route error
OR 1. Create arbitrary RERR messages
2. Do not send ACK messages (causing RERRs in other nodes)
2. Do not participate in following route request (A.1)
3. Do not relay data packets
OR 1. Drop data packets (case B)
2. Set hop limit/TTL to 0/1 (causing a RERR)

on the detection of modifications to routing data but not on the concealment of existing links.

In order to study how this behavior affects a MANET, we have done a number of simulations where we modeled a varying number of selfish nodes according to case A and B from table 1. The simulations were done using ns-2.1b8a and the DSR routing protocol. The scenario included 50 nodes moving in an area of 1500x300m according to the random waypoint model at speeds of $1\frac{m}{s}$ and $20\frac{m}{s}$ with no pause time. Twenty of the nodes were CBR sources sending 4 packets per second. Details of the simulation parameters are given in table 2. These parameters are typical for MANET simulations (see e.g. [13]) and are used for all following simulations.

Figure 1 shows the results of these simulations. We have varied the number of selfish nodes from 0 to 50 (the total number of nodes in the network). It is obvious that this number has a significant effect on the rate of packets that are successfully delivered in the network. In addition the movement rate has a clear effect. The faster nodes move, the lower the delivery ratio becomes. Finally we see that at lower speeds nodes of case B are more detrimental to the network

Table 2. Simulation parameters

Parameter	Value
Number of Nodes	50
Area X (m)	1500
Area Y (m)	300
Traffic Model	cbr
Sending rate (packets/s)	4.0
Max. number of connections	20
Packetsize (byte)	512
Simulationtime (s)	900

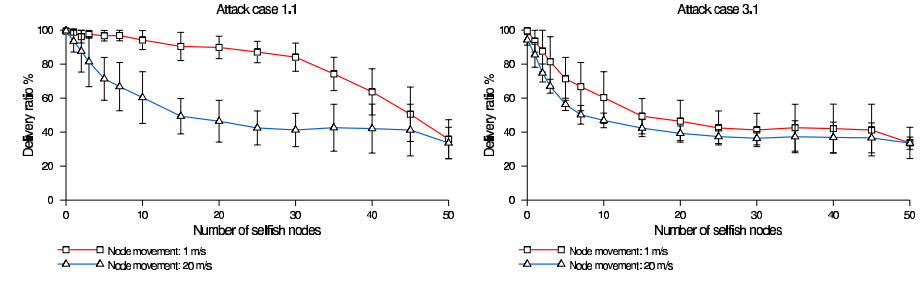


Fig. 1. Selfish attack simulation

than those of type case A whereas at higher speeds there are no big differences. Finally, when all the 50 nodes are selfish, we still get a delivery ratio of around 50%. This is due to the direct node-to-node traffic which does not need relaying. In this case the sender can reach the receiver directly.

What explanations can be found for this behavior? When the number of case A nodes rises in a network, there are fewer nodes available for building up routes. So if no alternative route can be established, there is no route to the destination which means that packets have to be discarded. That reduces the delivery rate. When movement speed rises, the delivery ratio also diminishes as the network in general gets more fragile. But the network still has a reasonable chance of routing around the selfish nodes. This changes with type case B. Here the nodes behave correctly during the route discovery phase. Thus they can be included in regular routes, but then they start to drop all packets. This isn't detected by DSR and no countermeasures are taken. So at a movement speed of $20\frac{m}{s}$ only 10% of the selfish nodes push the probability of a successful packet delivery below 50%.

Our simulations with AODV have revealed a similar behavior. This demonstrates clearly that an effective protection against selfish and malicious nodes is absolutely mandatory for ad hoc networks.

2 Preventing Selfish Nodes: Motivation Versus Detection and Exclusion

There are two approaches of dealing with selfish nodes. The first approach tries to give a motivation for participating in the network function. A typical system representing this approach is Nuglets by Hubeaux et al. [14, 15]. The authors suggest to introduce a virtual currency called Nuglets that is earned by relaying foreign traffic and spent by sending own traffic. The major drawback of this approach is the demand for trusted hardware to secure the currency. There are arguments that tamper-resistant devices in general might be next to impossible to be realized [16, 17]. A similar approach without the need of tamper proof hardware has been suggested by Zhong et al. in [18]. There exist also other unresolved problems with virtual currencies, like e.g. nodes may starve at the edge of the network because no one needs them for forwarding etc.

Most of the existing work in this field concentrates on the second approach: detecting and excluding misbehaving nodes. The first to propose a solution to the problem of selfish (or as they call it "misbehaving") nodes in an ad hoc network were Marti, Giuli, Lai and Baker in [19]. Their system uses a watchdog that monitors the neighboring nodes to check if they actually relay the data the way they should do. Then a component called pathrater will try to prevent paths which contain such misbehaving nodes. As they indicate in their paper, their detection mechanism has a number of severe drawbacks. Relying only on overhearing transmissions in promiscuous mode may fail due to a number of reasons. In case of sensor failure, nodes may be falsely accused of misbehavior. The second drawback is that selfish nodes profit from being recognized as misbehaving. The paths in the network are then routed around them, but there is no exclusion from service. We will later present more advanced sensors that will allow a better detection of selfish nodes.

In [20, 21] the authors describe a distributed intrusion detection system (IDS) for MANETs that consists of the local components "data collection", "detection" and "response" and of the global components "cooperative detection" and "global response". Whereas their architecture is very promising and similar to the one we use in our project, they neglect the aspect how their local data collection should find out on incidents like dropped packets, concealed links, etc.

Another system is the "Collaborative Reputation Mechanism" or CORE [22, 23]. It is similar to the distributed IDS by Zhang et al. and consists of local observations that are combined and distributed to calculate a reputation value for each node. Based on this reputation, nodes are allowed to participate in the network or are excluded. In their work, the authors specify in detail how the different nodes should cooperate to combine the local reputation values to a global reputation and how they should react to negative reputations of nodes. For the actual detection of selfish nodes, they only refer to the work of Marti.

A similar approach is conducted by Buchegger et al. with the CONFIDANT system [24, 25]. Again, they only marginally describe their detection mechanism and rely mostly on promiscuous overhearing.

3 The Mobile Intrusion Detection System (MobIDS)

We have developed a *Mob*ile *I*ntrusion *D*etection *S*ystem (*MobIDS*) that has a similar structure like some of the systems mentioned above. Because most of the other systems use overhearing, we tried to focus on enhancing this mechanism and develop additional sensors that can be used in parallel to have a higher detection accuracy. As you can see in figure 2, different sensors collect data from the network. As MobIDS is embedded in a complete security architecture called SAM [26], data from the routing protocol SDSR is also taken into account. SAM provides also node authentication based on public/private key pairs that are also used for authentication when distributing ratings (see below). SAM also includes a secure routing protocol called SDSR that negotiates secret session keys between the endpoints or a route and each of the inner nodes.

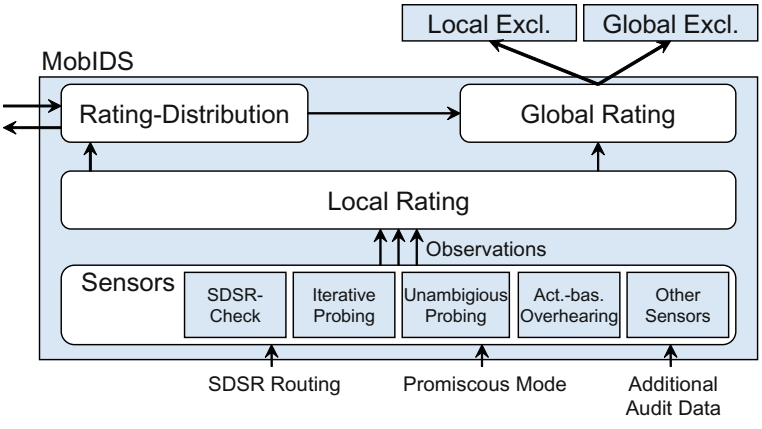


Fig. 2. Overview of MobIDS

The sensors generate *observations*. $\sigma_n^s \in [-1;1]$ represents the n^{th} observation of sensor s. Positive values represent a positive behavior whereas a negative value expresses non-cooperative behavior. All local observations of a node k_i and a sensor s regarding another node k_j at time t lead to a *sensor rating* $r_{k_i}^t(k_j|s) \in [-1;1]$:

$$r_{k_i}^t(k_j|s) = \left(\sum_{\forall n} \rho(t, t_n) \cdot \sigma_n \right) / n$$

where

$$\rho(t, t_n) = 1 - \left(\frac{t - t_n}{T} \right)^x$$

t_n is the time when a specific observation σ_n^s was made. The function ρ makes older observations less important than newer once, observations older than $t-T$ are ignored and can be discarded. x controls the degradation of older observations.

Finally all sensor ratings $r^t_{k_i}(k_j|s)$ are combined into a *local rating* $r^t_{k_i}(k_j) \in [-1;1]$ that expresses the judgment of node k_i regarding node k_j at time t:

$$r^t_{k_i}(k_j) = \sum_{\forall s} w_s \cdot r^t_{k_i}(k_j|s)$$

The local ratings are then *distributed* to neighboring nodes by flooding them periodically in a certain diameter surrounding a node. The distribution of ratings is secured by a simple acknowledge and retransmit mechanism. A node averages all received local ratings (including his own) which results in the *global rating* $gr^t_{k_i}(k_j)$.

As the initial observations are often based on statistical sensors, no node can prove that his rating is actually accurate. So when distributing ratings, these are signed by private keys of each node, but no further attempt is made to prove the credibility of a rating. Instead global ratings are only accepted when at least N nodes have contributed to the rating. This prevents alliances of less than N nodes from excluding other nodes from the network.

Based on the global rating, nodes may be excluded from the current network. MobIDS defines different thresholds t_t, t_e and t_r where t_e is the *exclusion threshold*. If the rating of a node k_i regarding a node k_j sinks below t_e, k_i will invalidate all routes containing k_j and will ignore all packets related to k_j. After some time, old negative observations will expire, so the rating of k_j will eventually increase again. As soon as the global rating exceeds the *rehabilitation threshold* t_e, k_j will be serviced again.

There is one problem: as the distribution process takes some time to deliver the local ratings to all nodes, the global ratings of different nodes regarding k_j may differ by a certain amount ϵ. If $r^t_{k_i}(k_j) < t_e < r^t_{k_l}(k_j)$ then node k_i will stop servicing k_j whereas k_l will still regard k_j as a cooperating node. So when k_i stops forwarding packets to k_j, sensors of k_l may detect this and punish k_i.

Therefor the system contains a third threshold t_t, the so called *tolerance threshold* where $t_e < t_r < t_t - \epsilon$. When $r^t_{k_l}(k_j)$ is below t_t, k_l will tolerate any node to deny service to k_j without deducing negative ratings from this.

In addition to local exclusion the security architecture contains a mechanism that allows global exclusion of nodes from MANETs by invalidating their cryptographic identity. The certified key pairs representing the identities of misbehaving nodes can be revoked, when enough incidents of selfish behavior are recorded. As this is outside the scope of this paper, please refer to [26] for details. We also will not go into any details on performance costs of the MobIDS system as we want to focus on the sensor part for the remaining part of the paper. A detailed description and analysis of the complete system can again be found in [26].

Another question is how the different thresholds should be chosen. Up to now we have adjusted them manually by running different simulations, testing the results and modifying the thresholds. In the final section we will outline future research on how to adjust them automatically.

It is obvious that without good sensors all the following steps (local and global rating, exclusion) will fail to deliver good results. So the rest of the paper focuses on this aspect of MobIDS.

4 MobIDS Sensors

4.1 Activity-Based Overhearing

We already mentioned that there are a number of problems when a node wants to determine whether another node actually relays its packet by using promiscuous mode and listening for the transmission. There are a lot of cases where a relay-node actually forwards a packet but the node overhearing the relay-node's activity will fail to realize that. If e.g. the overhearing node is currently transmitting or receiving data in an IEEE 802.11 network at a lower wire speed (e.g. 5.5 or 2 Mbps) then it won't be able to capture transmissions that happen at other speeds. Other problems include collisions, cooperating selfish nodes and many more.

We tried to improve the classical overhearing sensor, that simply tries to detect missing forwarding like described in [19], to avoid some of these drawbacks. We call the result *activity-based overhearing*. Here a node also tries to overhear forwarding of data packets by its next hop. A node constantly monitors its neighbors' traffic activity for regular data packets sent out by the neighbor nodes. The date of the last regular activity of a neighbor is stored in a table. When it sends a packet to another node and cannot detect a forwarding of the packet by the relaying node, this is esteemed a selfish behavior only when there has been a recent regular activity by this node. This way, the likelihood of a false detection in any of the cases described above is reduced. On the other hand a selfish node can evade our sensor only if it does not generate own traffic. But then it may as well leave the ad hoc network altogether.

Using this mechanism we can significantly improve the detection accuracy. Additionally our architecture introduces a detection threshold . The monitoring node will only trigger an alarm when it detects a certain number of packets being dropped within a certain timeframe. This way a small number of false detections will not lead to any actions against the assumed selfish node.

In order to verify our claim, we have performed a number of simulations which compare traditional and activity-based overhearing. The simulation setup was identical to table 2. Figure 3 shows simulation results at a movement speed of $1m/s$. It verifies the better performance of the activity-based overhearing mechanism. The left graph shows the detection rate of MobIDS in the presence of a specific number of selfish nodes that operate according to case B in the attack tree (forward routing traffic, but drop subsequent data traffic). All values are taken as the average of 10 different simulation runs. Let us assume a network with 2 selfish nodes. Then each of the two nodes is (on average) detected by 1.1 monitoring nodes using traditional overhearing. When we use activity-based overhearing there are 4.5 nodes detecting each selfish node which is 4 times better than the classical overhearing sensor. When the number of selfish nodes gets higher (from 5 to 10), the traditional overhearing performs slightly better than activity-based overhearing. We can use both approaches when the results of both sensors are combined. The combined overhearing sensor always considers a missing forwarding activity and devalues the rating of the corresponding node.

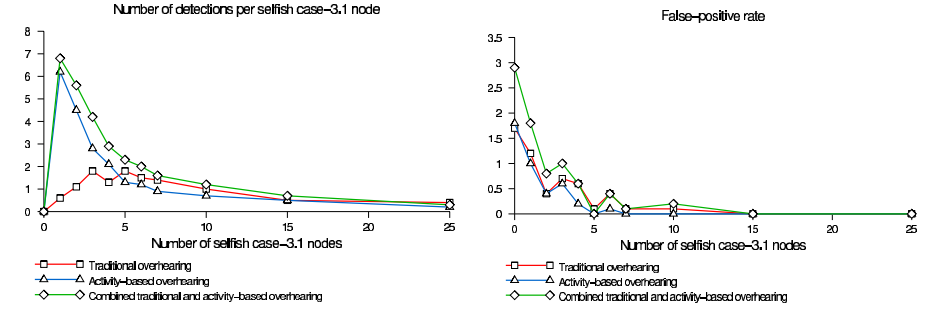


Fig. 3. Traditional vs. Activity-based Overhearing at $1m/s$

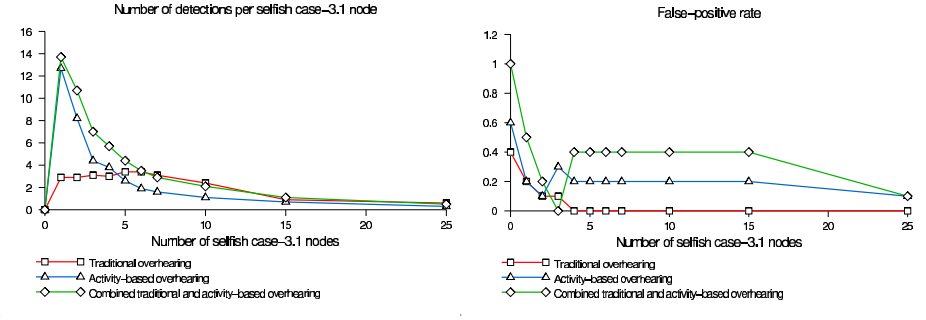


Fig. 4. Traditional vs. Activity-based Overhearing at $20m/s$

But the magnitude of this downrating is determined by the time since the last seen activity by this node. The longer a node has been inactive, the smaller the downrating. As you can see from the simulations, this delivers highly acceptable results.

When the number of selfish nodes becomes large[1], detection rates get really bad. Only one or two nodes will detect a selfish node during the average simulation run. This is partially because we assume that selfish nodes do not act as sensors anymore. So in case of 10 selfish nodes you also have to take into consideration that 20% of the sensors are gone. In order to get good results here, we need to combine the overhearing sensor with other sensors like the probing sensor described later.

The right graph in figure 3 shows the false-positives that the overhearing sensors produce. It is important to note that these values are always low compared to the correct positive identifications of selfish nodes. In MobIDS, a node is excluded from the network only if multiple nodes agree on it being selfish or malicious. So when only one node has a false-positive this has no negative effects on the detected node.

[1] more than 10 nodes or 20% of all nodes!

Simulations at $20m/s$ (figure 4) show that the detection rate of the activity-based and combined overhearing even increases at higher speeds. This is due to the larger number of routing protocol packets that circulate in the network. This enables the activity detector to predict more precisely whether another host is still in communication range.

4.2 Iterative Probing

In [27] the authors describe a mechanism called *probing* to detect selfish or selfish nodes in a MANET route from source S to destination D. They use onion-encryption to embed a probe command for a specific node X into the normal data packets. When X decrypts its onion layer, it will find this command and send back an acknowledge packet to the source. As soon as an acknowledge is missing, S starts a binary search in the path to find out, where packets are being dropped. S simply sends probes to the selected nodes and waits for their replies. Figure 5 shows the binary search after which we call it *binary probing*.

This approach has a number of drawbacks. The onion-encryption is relatively expensive, as the sender has to encrypt each probe packet multiple times depending on the path length. Furthermore each node has to decrypt the packet once and each packet has to be acknowledged explicitly by the recipient D.

But there is one even more severe problem. There is no reliable detection of the node dropping packets. When a selfish node gets a probe packet it will notice that a probing is under way. Now it can choose to cooperate and forward packets for a limited time (until the probe is over) and then continue to drop packets. Even worse depending on how the probing is realized ([27] is not completely clear on this), it may even be able to selectively drop probe packets destined for another host. This host then doesn't acknowledge the probe and is marked as hostile.

In our mechanism, that we call iterative probing, we use a different approach. Like [27] we assume that a source S has established a secret key k_{SX_i} with each node X_i $(i = 1...n-1)$ in it's path to a destination X_n. There is a command field C included in the packet header that *may* contain a node id X_i which is encrypted by k_{SX_i}, so $C = enc_{k_{SX_i}}(X_i, P)$ $(i = 1...n)$. Otherwise the field contains a random number. P is a random padding which makes multiple probe commands to the same node still look different. So no node can tell whether C contains only garbage or a probe command to another node.

Each intermediate node X_j will now try to decrypt C. If the result is its node ID, it will send an (encrypted) probe reply packet back to S, otherwise it will process the packet as usual. So S has to encrypt only a small portion of the packet and it has to do so only once (compared to the onion-encryption approach) Intermediary nodes will only have to decrypt the small command field and not the whole packet.

In normal operation (that is while S receives packets from X_n as a reply to the packets it sent to X_n) there is no need for probing. But when S hasn't received a packet from X_n for a certain amount of time t, it will sent a probe packet to X_n. If there is no reply within a certain timeout, it will send a probe

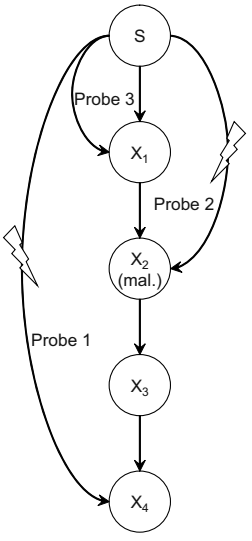


Fig. 5. Binary Probing

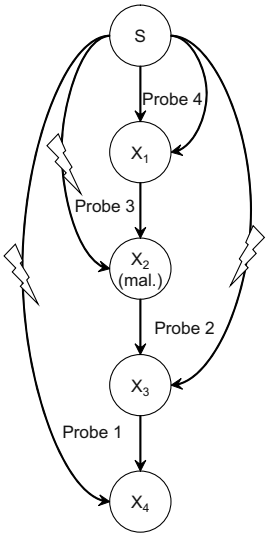


Fig. 6. Iterative Probing

to X_{n-1} and so on until it receives a reply from a node or reaches X_1. This is called *iterative probing* and shown in figure 6.

Iterative Probing has one advantage over binary probing: a selfish node only gains knowledge of an ongoing probing when it is his turn to answer a probe. So he is not able to blame any nodes on an arbitrary position later in the path by selectively filtering out or forwarding probe packets. Instead there are only two possibilities: he can reply to the probe or he can discard it. All later probe packets are sent to nodes earlier in the path and can not be manipulated any more.

But there is still one problem remaining. Let X_j be the first node from which S receives an acknowledge. There are two possibilities now. In the first case X_{j+1} is the selfish node dropping all packets. Then X_{j+1} will also dropping probe packets and X_j is working properly. In the second case X_j is the selfish node dropping packets. But before dropping a packet, X_j checks if it is a probe addressed to himself. In order to be harder to detect, X_j will then reply to the probe. Due to space limitations we cannot show and discuss the result graphs here.

Although the iterative probing sensor is harder to fail than the binary probing, it can not distinguish which of the two nodes is actually the malicious one. We call this problem the *probing dilemma*. In the next section we will present an approach to prevent this. But first we give an analysis of the iterative probing.

Figure 9 (left side) shows the simulation results for the iterative probing sensor facing the standard adversary – a selfish case B node. Even for 10 selfish nodes we still have an average of 4.9 nodes detecting each selfish node. The false-positives are negligibly low. So probing is an efficient way of detecting selfish nodes.

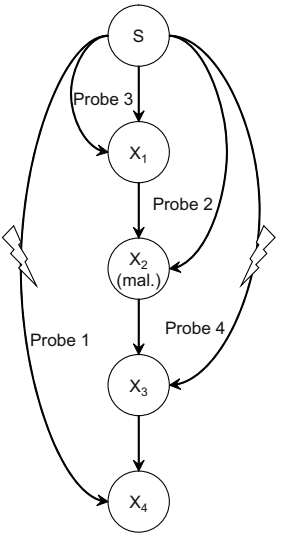


Fig. 7. X_2 answers probes: possible selfish nodes $\{X_2, X_3\}$

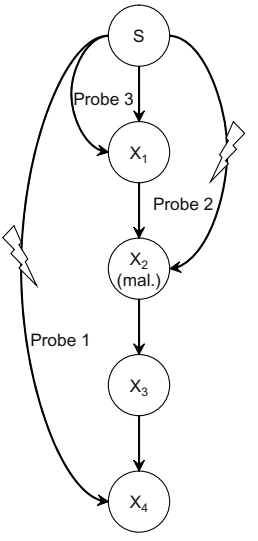


Fig. 8. X_2 answers not: possible selfish nodes $\{X_1, X_2\}$

4.3 Unambiguous-Probing

As indicated above, the probing techniques described so far face a serious problem: probing can not unambiguously detect a selfish node. Even worse, the standard probing described in [27] allows a malicious node to make another arbitrary node look selfish. Our iterative-probing can narrow the potential adversary nodes down to two nodes. In order to clearly identify one of these nodes as being responsible for the dropped data packets, we can combine the iterative probing with overhearing. Let X_j and X_{j+1} be the nodes that are suspicious of dropping packets like described above. Now we can verify if X_j is dropping the packet

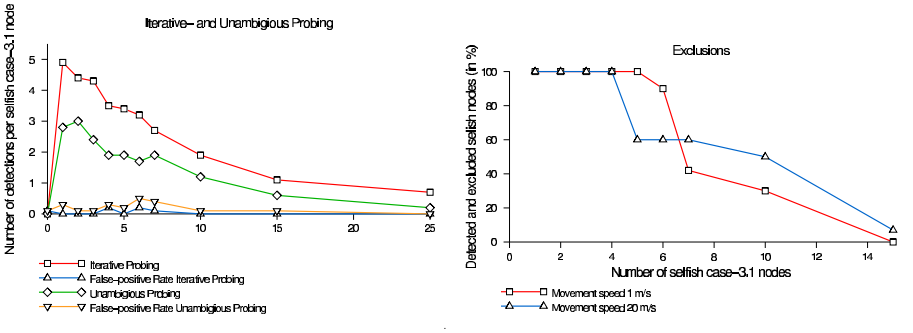


Fig. 9. Iterative Probing and exclusion of selfish nodes

by asking X_{j-1} to check if he can overhear the forwarding of a following probe packet by node X_j. If this probe fails and X_{j-1} can't hear X_j forwarding the packet, then it is very likely that X_j is dropping the packets, otherwise X_{j+1} is the node responsible for the packet drop.

4.4 Overall Detection Rate

MobIDS combines all the presented sensors in order to make a decision on excluding nodes from the network. Our simulation results show that the detection of misbehaving nodes is very accurate and we have practically no false accusations. Figure 9 (right side) shows the percentage of discovered and excluded selfish nodes at different movement speeds. In this scenario, three different nodes were needed to detect another node as selfish in order to exclude it from the network. In the simulations, we used combined-overhearing, unambiguous-probing and route-request scanning sensors in parallel. The route-request scanning sensor is a specialized overhearing sensor that specifically checks whether route requests of the routing protocol are rebroadcasted correctly by neighboring nodes. So it can detect misbehaving nodes that do not forward route requests properly. Due to space limitations, it was not presented here.

5 Conclusion and Future Work

As we have seen the construction of sensors to detect selfish or malicious nodes in ad hoc networks is a complex task. In this paper we have presented a number of different sensors that can detect different kinds of selfish nodes with a good confidence as shown by our simulation results. If multiple sensors are active in parallel and a selfish node is detected by a number of these sensors, then this is a good indication for excluding the node from the network.

One remaining problem with our current simulations is that all the thresholds need to be set manually in order to get good detection results. So in the future we will try to find ways how these values can be set and adjusted automatically during operation. Possible candidates might be some kind of an adjustment algorithm or a self-learning system using neural networks. Furthermore we plan to develop and test additional sensors that will e.g. use topology information from the routing protocol in order to detect selfish nodes.

References

1. David B. Johnson, David A. Maltz, Yih-Chun Hu, and Jorjeta G. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). `http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-09.txt`, April 2003.
2. Charles E. Perkins, editor. *Ad Hoc Networking*. Addison-Wesley, 2001.
3. Bruce Schneier. Modeling security threats. *Dr Dobb's Journal*, December 1999. also available as `http://www.ddj.com/documents/s=896/ddj9912a/9912a.htm`.

4. Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure On-Demand Routing Protocol for Ad hoc Networks. In *Proceedings of MobiCom 2002*, Atlanta, Georgia, USA, September 2002.
5. Panagiotis Papadimitratos and Zygmunt J. Haas. Secure Routing for Mobile Ad hoc Networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, San Antonio, TX, January 2002. also available as `http://wnl.ece.cornell.edu/Publications/cnds02.pdf`.
6. Panagiotis Papadimitratos and Zygmunt J. Haas. Secure Routing for Mobile Ad Hoc Networks. Working Session on Security in Wireless Ad Hoc Networks, EPFL, (published in Mobile Computing and Communications Review, vol.6, no.4), June 2002.
7. Panagiotis Papadimitratos and Zygmunt J. Haas. Securing Mobile Ad Hoc Networks. In M. Ilyas, editor, *Handbook of Ad Hoc Wireless Networks*. CRC Press, 2002.
8. Panagiotis Papadimitratos, Zygmunt J. Haas, and P. Samar. The Secure Routing Protocol (SRP) for Ad Hoc Networks. draft-papadimitratos-secure-routing-protocol-00.txt, December 2002.
9. Panagiotis Papadimitratos and Zygmunt J. Haas. Secure Link State Routing for Mobile Ad Hoc Networks. In *IEEE Workshop on Security and Assurance in Ad hoc Networks, in conjunction with the 2003 International Symposium on Applications and the Internet*, Orlando, FL, January 2003.
10. Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth M. Belding-Royer. A Secure Routing Protocol for Ad Hoc Networks. In *Proceedings of 2002 IEEE International Conference on Network Protocols (ICNP)*, November 2002. also available as `http://signl.cs.umass.edu/pubs/aran.icnp02.ps`.
11. Manel Guerrero Zapata. Secure Ad hoc On-Demand Distance Vector Routing. *ACM Mobile Computing and Communications Review (MC2R)*, 6(3):106–107, July 2002. also available as `http://doi.acm.org/10.1145/581291.581312`.
12. Manel Guerrero Zapata and N. Asokan. Securing Ad hoc Routing Protocols. In *Proceedings of the 2002 ACM Workshop on Wireless Security (WiSe 2002)*, pages 1–10, September 2002. also available as `http://doi.acm.org/10.1145/570681.570682`.
13. Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Mobile Computing and Networking*, pages 85–97, 1998. also available as `http://citeseer.nj.nec.com/broch98performance.html`.
14. Levente Buttyán and Jean-Pierre Hubaux. Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks. Technical Report DSC/2001/001, EPFL-DI-ICA, January 2001.
15. Levente Buttyán and Jean-Pierre Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5), October 2003.
16. R. Anderson and M. Kuhn. Tamper Resistance - a Cautionary Note. In *Proceedings of the Second Usenix Workshop on Electronic Commerce*, pages 1–11, November 1996. also available as `http://citeseer.nj.nec.com/article/anderson96tamper.html`.
17. Ross Anderson and Markus Kuhn. Low cost attacks on tamper resistant devices. In *IWSP: International Workshop on Security Protocols, LNCS*, 1997. also available as `http://citeseer.nj.nec.com/anderson97low.html`.

18. Sheng Zhong, Jiang Chen, and Yang Richard Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *Proceedings of IEEE Infocom '03*, San Francisco, CA, April 2003.
19. Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000. also available as `http://citeseer.nj.nec.com/marti00mitigating.html`.
20. Yongguang Zhang and Wenke Lee. Intrusion detection in wireless ad-hoc networks. In *Mobile Computing and Networking*, pages 275–283, 2000. also available as `http://citeseer.nj.nec.com/zhang00intrusion.html`.
21. Yongguang Zhang, Wenke Lee, and Yi-An Huang. Intrusion Detection Techniques for Mobile Wireless Networks. *to appear in ACM Wireless Networks (WINET)*, 9, 2003. also available as `http://www.wins.hrl.com/people/ygz/papers/winet03.pdf`.
22. Pietro Michiardi and Refik Molva. Prevention of Denial of Service attacks and Selfishness in Mobile Ad Hoc Networks. http://www.eurecom.fr/ michiard/pub/michiardi_adhoc_dos.ps.
23. Pietro Michiardi and Refik Molva. A Collaborative Reputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks. In *Proceedings of the 6th IFIP Communication and Multimedia Security Conference*, Portorosz, Slovenia, September 2002.
24. Sonja Buchegger and Jean-Yves Le Boudec. Nodes Bearing Grudges: Towards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks. In *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403–410, Canary Islands, Spain, January 2002. IEEE Computer Society. http://citeseer.nj.nec.com/article/buchegger02nodes.html.
25. Sonja Buchegger and Jean-Yves Le Boudec. Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes - Fairness in Distributed Ad-hoc Networks. In *Proceedings of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Lausanne, CH, June 2002.
26. Frank Kargl. *Sicherheit in Mobilen Ad hoc Netzwerken.* PhD thesis, University of Ulm, Ulm, Germany, 2003. also available as `http://medien.informatik.uni-ulm.de/~frank/research/dissertation.pdf`.
27. Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. An On-Demand Secure Routing Protocol Resilient to Byzantine Failures. In *ACM Workshop on Wireless Security (WiSe)*, Atlanta, Georgia, September 2002. also available as `http://citeseer.nj.nec.com/article/awerbuch02demand.html`.

A Security Architecture for Mobile Wireless Sensor Networks

Stefan Schmidt[1], Holger Krahn[2], Stefan Fischer[1], and Dietmar Wätjen[3]

[1] TU Braunschweig, Institute of Operating Systems and Computer Networks, Mühlenpfordtstr.23, 38106 Braunschweig, Germany
{schmidt|fischer}@ibr.cs.tu-bs.de
[2] TU Braunschweig, Institute of Software Systems Engineering, Mühlenpfordtstr.23, 38106 Braunschweig, Germany
krahn@sse.cs.tu-bs.de
[3] TU Braunschweig, Institute of Theoretical Computer Science, Mühlenpfordtstr.23, 38106 Braunschweig, Germany
waetjen@iti.cs.tu-bs.de

Abstract. Wireless sensor networks increasingly become viable solutions to many challenging problems and will successively be deployed in many areas in the future. However, deploying new technology without security in mind has often proved to be unreasonably dangerous. We propose a security architecture for self-organizing mobile wireless sensor networks that prevents many attacks these networks are exposed to. Furthermore, it limits the security impact of some attacks that cannot be prevented. We analyse our security architecure and show that it provides the desired security aspects while still being a lightweight solution and thus being applicable for self-organizing mobile wireless sensor networks.

1 Introduction

Wireless sensor networks are increasingly showing viable solutions to many challenging problems that require the monitoring of real-world events and are predicted to affect our future daily lives in important ways [8]. Large numbers of small, severely resource-constraint devices form these networks, by cooperating to achieve a common goal, which would be unattainable for the individual nodes.

These networks are usually deployed in uncontrollable environments that are not trustworthy. In addition to common threats in wireless networks, e.g. information disclosure, message injection, and replay attacks, sensor networks are physically accessible and consequently more vulnerable. An attacker may capture and compromise a node and thus be able to control a valid member of the network. Furthermore, a variety of Denial of Service attacks are possible in sensor networks.

In this paper we present a security architecture for self-organizing mobile wireless sensor networks that addresses most of the problems above. It utilizes lightweight cryptographic algorithms that allow for easy authentication between

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 166–177, 2005.

the mobile sensor nodes and secure the communication inside the network. Furthermore, it minimizes the effects of compromised sensor nodes.

The rest of the paper is structured as follows. In the beginning we set the stage for our work by introducing a sensor network setting and its environment. Furthermore, we provide a detailed threat analysis for this setting. Thereafter, we describe the technical aspects of our security architecture and present some implementation details. Subsequently, we provide an analysis of our security architecture. A discussion of related as well as future work and a summary of our contribution conclude the paper.

2 Sensor Network Architecture and Environment

In this section we want to introduce the sensor network characteristics on which our security architecture is based. Three groups of aspects have a direct impact on the design of our security architecture: the sensor nodes characteristics, the network characteristics, and the environment.

2.1 Sensor Nodes

The sensor nodes are characterised as severely resource-constraint devices in terms of available energy, memory, and computational power. For example, our research node, the Embedded Sensor Board ESB 430/1, developed by the Freie Universität Berlin [19], is powered by three standard AAA batteries, incorporates 60 kbyte flash memory, 2 kbyte RAM, and 8 kbyte EEPROM, and uses the micro controller MSP 430 from Texas Instruments. Furthermore, the sensor nodes are not tamper-proof, due to cost factors and the general difficulty in building such devices [1]. Consequently, it is possible to physically manipulate the devices if captured. For interaction purposes, the nodes are equipped with radio frequency communication capabilities. However, this wireless communication provides only limited bandwidth.

These sensor node-specific factors set several constraints for the security architecture. Since only a fraction of the total memory may be used by the cryptographic algorithms and key material, the security architecture demands very lightweight cryptographic algorithms with relatively short key sizes. Furthermore, cryptographic computations need to be executable in an appropriate amount of time as the execution of cryptographic algorithms is not the main task of the nodes. Due to the limited bandwidth and communication being the most expensive operation in terms of energy, messages should not be extended significantly in length when applying security services.

2.2 Sensor Network

The sensor network consists of numerous mobile sensor nodes. It does not incorporate an infrastructure or any kind of hierarchy. In fact, the sensor nodes are equal devices in terms of the role they can play in the network and should self-organize to accomplish their appointed task, without any external guidance

or supervision. Thus, the sensor nodes gather information based on their sensing capabilities and make decisions based upon the gathered data. The communication paradigm is based on a distributed virtual shared information space (dvSIS) combined with content-based forwarding of information as described in [6, 14]. The dvSIS describes the state of both the network and the environment. It is based on the information that is acquired by the sensor nodes. Information exchange is based on one-hop communication, which eliminates the need for routing mechanisms. Basically, all information is flooded inside the sensor network and the nodes decide whether or not to forward the received information based on filtering rules, e.g. the information being new or already known from their local dvSIS. However, no device has complete knowledge (i.e. has all information in its dvSIS) and must therefore rely on partial information. Nevertheless, employing these concepts, any node may become a gateway to an external network or observer, which is preferable in mobile networks since the user might not have control over the position of potential gateway-nodes. Another important point in sensor networks is the limited lifetime of sensor data. Sensor data and accordingly events that are derived from it should be communicated in realtime.

The network characteristics, similar to the node characteristics, determine important aspects of the desired security architecture. Considering the node mobility, authentication and key exchanges must not depend on numerous extra messages, since the topology is subject to frequent change. Additionally, all necessary cryptographic functions and key material must reside and be executable on the nodes. With respect to the realtime property of sensor networks, cryptographic algorithms should also be as fast as possible. Finally, the security architecture needs to be scalable to accommodate high numbers of mobile nodes.

2.3 Environment

The environment of these sensor networks depends on the assigned task and must in most cases be seen as uncontrollable and not trustworthy. Even scenarios that are characterised by a hostile environment can be envisioned.

This strongly emphasizes that a security architecture must be fault tolerant and ensure certain levels of security even in the case of compromised nodes.

3 Threat Analysis

Any sound security architecture requires a thorough threat analysis beforehand to enable an evaluation of its benefits. This section provides a detailed threat analysis for the sensor network that was outlined above.

The commonly known two categories of active and passive attacks can also be applied to sensor networks. However, sensor networks are susceptible to more attacks than ordinary networks, as we will show in the following.

3.1 Passive Attacks

Passive attacks on the sensor network are relatively easy because of wireless communication. The illegitimate disclosure of information, which breaks confiden-

tiality, presents the major threat here. Therefore, information that is exchanged between the nodes should be encrypted to ensure its confidentiality. Traffic analysis presents a minor threat to this kind of sensor networks since the sensor nodes communicate frequently to publish their sensor readings, derived events, or to coordinate their behaviour.

3.2 Active Attacks

Successful active attacks allow the attacker to seriously disrupt the functioning of the network. Several points of attack are imaginable to influence the network in its data capture and decisions. It must be ensured that it is impossible to inject or replay messages into the network. Otherwise an attacker could masquerade as a legitimate member of the network and send its own information or replay old data, which might lead to wrong decision making inside the network. Another means of influencing the network originates from the way sensor networks operate. If an attacker is able to generate physical stimuli he can in parts specify what data the sensor nodes collect. The commonly known threat of message modification, however, does not pose a threat to our sensor network. Due to the local broadcast of the messages, it should be almost impossible to intercept and modify a single message before any other node receives it.

A serious vulnerability arises from the physical accessibility of the sensor nodes. Communicated location information might lead attackers to find out about the positions of single nodes, which further emphasizes the need to keep the communication between the nodes confidential. Furthermore, considering that the sensor nodes are not tamper-proof, it is possible to physically compromise captured nodes. In particular, the attacker is likely to attain a node's cryptographic key material. If an attacker successfully compromises a node, he gains full control about it and might use it to spy on the the rest of the sensor nodes. He is now able to decipher any encrypted communication directed at that particular node and to send legitimate custom messages.

Also several possibilities exist to start a Denial of Service (DoS) attack against a sensor network [22]. For example, an attacker who is able to create physical stimuli might flood the network or at least parts of it, so that real events drown in the artificial noise of stimuli. Furthermore, attackers who possess the location information of the nodes could capture or destroy the nodes one by one. The traditionally known jamming of the wireless medium or sensor specific DoS-attacks such as sleep deprivation torture [21] present threats as well.

4 Security Architecture

Generally, asymmetric as well as symmetric cryptography could be employed to achieve security. However, an implementation [12] of an elliptic curve, one of the fastest available asymmetric methods with an underlying field of about 2^{128} elements, shows, that it seems to be infeasible to implement a fast public key system on the given nodes. The calculations would delay the message transmis-

sion more than a second, which is unacceptable. Consequently we refrain from using asymmetric cryptography.

Our security architecture is based on three different interacting phases: a pairwise key agreement to provide authentication and the initial key exchange, the establishment of sending clusters to extend pairwise communication to broadcast inside the communication range, and encrypted and authenticated communication of sensor data.

4.1 Pairwise Key Agreement

To achieve pairwise key agreement we use the Blundo-et-al.-scheme [5], which is based on a predistribution scheme by Blom [4] and enables two nodes to determine a pairwise secret that is solely shared among these two nodes. The scheme is unconditionally secure and resistant against collusion of a maximum of t users. In terms of a cooperative sensor network, this means that an attacker has to compromise more than t nodes to compromise the whole network.

The scheme requires a certificate authority (CA) which randomly generates a symmetric bivariate polynomial $f(x, y)$ of degree t over an arbitrary finite field.

$$f(x,y) = \sum_{i,j=0}^{t} a_{ij} x^i y^j \quad (a_{ij} = a_{ji})$$

The CA, ensuring that each sensor has an unique ID, evaluates the polynomial in the following way:

$$g_{ID}(x) = f(x, ID).$$

Thus g_{ID} is a polynomial of degree t with a single variable x. The CA transfers the individual key material (the coefficients of the polynomial g_{ID}) to the nodes prior to the deployment of the sensor nodes.

Two nodes are able to determine their pairwise secret by evaluating their private polynomial $g_{ID}(x)$ where x denotes the other node's identity. It can be derived directly from the symmetry of the polynomial $f(x, y)$ that both nodes calculate the same value.

An efficient way to implement the Blundo-et-al.-scheme is presented in appendix A.

4.2 Sending Clusters

To communicate securely, every node establishes a randomly generated key within its neighbourhood. This key is used solely by this node to encrypt and authenticate its messages. If a node receives a message of which the content cannot be decrypted and authenticated (i.e. it does not know the key) it calculates the pairwise secret for the sender and itself using the Blundo-et-al.-scheme. This secret is then used to transfer its own key secretly to the sender, which replies by transferring its key. The node automatically sends its own key, because we assume that if it could not decrypt and authenticate the other message, a change

in the topology must have happened and the other node equally does not know its sending key.

This protocol enables a node to establish its key in a new environment and get to know the other nodes' keys with just two messages (request and reply) per neighbour. It is not restricted to an initialisation phase and can be used at any time and with any other legitimate node of the network. Thus it supports mobile sensor networks as well as the deployment of new nodes at a later point of time.

4.3 Encrypted and Authenticated Communication

The sensor nodes always broadcast messages to their direct neighbourhood. For encryption purposes the counter mode of operation [10] (see also figure 1) is used, which allows an encryption of the message without changing its length. In addition to the message itself the counter s_j is added, which results in ordered and unique messages. This overhead can be avoided if both sender and receiver increment the counter after each transmission. However, due to the lossy nature of the communication in sensor networks this procedure seems inappropriate. A message loss by any of the neighbours would require two additional messages (request of counter value and reply). Thus, the counter value is included in every message. It is not necessary to use the full block size of an encryption algorithm

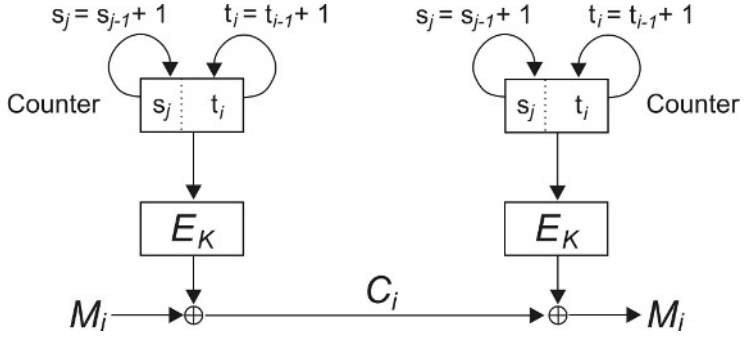


Fig. 1. Counter mode of operation. The counter register is divided into two parts: s which is incremented once per message and t which is reset to zero for a new message and incremented once per block

for the counter value, because any counter length can be padded with zeros. Since the same counter should not be used twice, the size of the counter limits the number of messages which can be encrypted by a single key.

Our security architecture can be implemented using any encryption algorithm. Often the RC5 algorithm [18] is suggested as being a well suitable algorithm for sensor networks. We give two reasons why this choice is not optimal for our architecture. First, the key expansion step of the algorithm cannot be integrated in the encryption process. This results in the need to have round keys

stored for the whole encryption process which leads to a slightly higher memory requirement than desired (at least 112 bytes for RC5-32/12/8 when the original key needs to be kept). Second, the RC5 algorithm extensively uses circular shifts. This operation is often not supported by low cost processors and must be emulated by single step shifts, which leads to bad runtime behaviour.

For our existing implementation we examined the AES-final candidates. The Rijndael algorithm [16] seems to offer the best performance, but requires large lookup-tables. Also this algorithm is not constructed for a key length shorter than 128 bit, which are often used in sensor networks. Simply padding the key might expose new flaws. Therefore, we decided to incorporate Serpent [2] which has good runtime behaviour because it can be implemented using logical operations only. These are much faster than circular shifts on low cost platforms. A reduction of the number of rounds from 32 to 16 yields to linear speed-up while at the same time not allowing any published attacks to be successful.

The encryption algorithm can be reused for the well-know CBC-MAC. The result of the chain of encryption operations can be used to ensure the integrity and the authentication of sensor data. Once again, there is no need to add the complete output of the MAC-function as a checksum to the message, because 16 byte of overhead per message seems to be inappropriate for sensor networks.

5 Implementation

We implemented a prototype of our security architecture using the sensor nodes ESB 430/1 [19]. The algorithms require 17,1 Kb. This results mainly from the Serpent algorithm being a standard implementation, which has only been optimized with regard to volatile memory usage and speed but not code size. During run-time the algorithms need additional 86 bytes for their operations. The average delay caused by this is 30,9 ms per 16 byte of cleartext information.

The security degree t (number of nodes an attacker has to compromise to successfully calculate the CA's polynomial) is completely adjustable but influences the run-time and memory requirement. For our 80-bit implementation (size of the pairwise secrets) $(t+1) \cdot 10$ bytes are required. The coefficients can be freely distributed between volatile memory, code space and EEPROM wherever space is available. For our testing, we chose the security degree to be 20 and stored the coefficients in memory. Additionally, a node has to store the sending key of each neighbour. In our implementation nodes store up to 20 sending keys. After that they start to reuse the space by deleting the oldest one. In summary, we use around a quarter of the available memory for our security architecture.

Also, for our prototype we used a 4 byte MAC and a 2 byte index which merely results in an extension of the messages by 6 bytes.

6 Analysis

By employing our security architecture we are able to prevent most of the depicted threats while considering the constraints that the sensor network de-

mands. It provides confidentiality as well as integrity for the communicated information and ensures the authenticity of the sensor nodes. Furthermore, it minimizes the effects of compromised nodes. All deployed cryptographic algorithms are efficient in terms of run-time and memory usage and do not extend the messages significantly.

Confidentiality is achieved by encrypting the messages. This prevents any illegitimate disclosure of information. Futhermore, the CBC-MAC ensures the integrity of the messages. We use short MACs in our architecture to reduce the message overhead, which, on the one hand, enables the attacker to determine legitimate messages by brute force. On the other hand, he has no control over the message content due to the encryption of the message. Furthermore, the counter mode of operation, which is used in the encryption process, automatically adds an index to each message without adding overhead compared to simple encryption. Hereby, replay attacks are prevented, because a receiver can keep track of already used counter values, while the messages are only slightly extended.

Only legitimate nodes can join the communication, since no illegitimate node is able to derive the sending key of the other nodes. An attacker may conduct the pairwise key agreement protocol using a captured key request message from another node. However, it is only possible for him to attack the encrypted sending key (encrypted with the pairwise secret key), which, if no new ways to attack Serpent become known, can be seen as impossible. Furthermore, the authentication process is implicitly included in the key exchange, which only requires two additional messages.

One threat we cannot prevent is the possible capturing and compromising of nodes. However, our security architecture minimizes its effects. While an attacker, who successfully compromised a node, may be able to authenticate himself to the network and be able to join in the communication, he can only do so using the one compromised node. The communication between non-compromised nodes remains secure. This holds true for up to t compromised nodes, which is based on the security specified by the initial symmetric bivariate polynomial calculations. It is important to note that t is not neccessarily linearly dependent on the actual network size. It denotes the number of nodes that need to be physically compromised, which is by far the most expensive attack in this scenario and thus more unlikely than radio-based attacks.

Due to the fact that cryptographic algorithms cannot prevent DoS attacks, the security architecture does not address these threats any further.

Our architecture is scalable and robust since all operations take place inside a node's communication radius. Thus, the actual size of the network does not influence its local security associations.

7 Related Work

Security in sensor networks has been studied by several other researchers.

Perrig et al. developed the security architecture SPINS, which is based on the two protocols SNEP, a protocol for data confidentiality, two-party data authen-

tication, and data freshness and μTESLA, a broadcast authentication protocol [17]. Their architecture relies on the concept, that every node shares a secret key with a trusted base station, which is at all times able to communicate with every node in the network.

Furthermore, several key management schemes have been put forward for sensor networks: Basagni et al. proposed a solution to periodically update a symmetric key which is shared by all nodes in the network [3]. Their solution is based on the assumption that all nodes are constructed tamper-proof, which is not always the case [1]. Carman et al. studied several key management protocols in sensor networks with respect to performance on different hardware platforms [7]. Zhu et al. proposed the Localized Encryption and Authentication Protocol (LEAP) [24] which utilizes four types of keys for each node. These are used for different purposes and range from the individual key that is shared with the base station, up to a group key that is shared with all nodes in the network. Eschenauer and Gligor presented a pool-based random key predistribution system [11], which Chan et al. extended by presenting three new mechanisms for key establishment [9].

The key setup algorithm that is most related to ours is presented by Liu and Ning [15]. They proposed a combination of the Blundo-et-al.-scheme and the pool-based random key predistribution system by Eschenauer and Gligor [11]. With the same storage requirements, the CA uses shorter polynomials than the non-modified Blundo-et-al.-scheme but deploys more than one private polynomial from a pool to a single node. If two nodes share at least one polynomial from the same scheme, they use this to determine the pairwise secret as described in section 4.1. The combination has a higher resistance against the possibility of compromising nodes than the Blundo-et-al.-scheme if the compromised nodes are picked randomly, with the disadvantage that only certain nodes are able to communicate. Lui and Ning state that this can be amended by using an intermediate node to establish a connection between two nodes which have no scheme in common. We think this enables new attacks in mobile sensor networks. A single compromised node is able to vouch for other nodes the attacker has fabricated and allows them to become part of the sensor network.

Furthermore, the system is only advantageous if an attacker is not able to determine the identity of the polynomials stored on a certain node. Therefore, in [11] a protocol is proposed where an encrypted list, using the pairwise keys for the encryption, is exchanged. Adding to the disadvantage that this requires an additional message exchange of at least 32 byte length (64-bit-block cipher and 4 polynomials per node), the system is still not secure against compromising nodes. If an attacker has compromised at least a single node, he can actively use it to communicate with others to determine whether they share any polynomial. With this method, the attacker is able to actively attack nodes that possess certain polynomials. Due to the longer messages and the described attacks, we decided to use the unmodified Blundo-et-al.-scheme.

Wood and Stankovic identified several DoS attacks in sensor networks [22] and presented a protocol, which allows to map regions that are subject to DoS by radio jamming [23].

8 Summary and Future Work

In this paper, we have proposed a security architecture that provides confidentiality, integrity, and authentication for a mobile wireless sensor network. For this purpose, we have presented algorithms to easily set up pairwise secret keys between the mobile sensor nodes and to establish a sending cluster per node, in which it can communicate its messages securely. Furthermore, our solution minimizes the effects of compromised nodes. Compromising an adjustable number of sensor nodes does not compromise the whole security architecture but restricts the security breach to the immediate neigbourhood of the compromised node. Finally, we have implemented a prototype of our security architecture, which clearly shows that it is a lightweight solution and applicable for self-organizing mobile wireless sensor networks.

Several directions for future research arise from our solution. First, we intend to simulate our approach, using NS-2, in order to determine the maximum grade of node-mobility our security architecture is able to cope with. Second, we would like to integrate the ability to identify compromised nodes and methods to exclude them from the network. Another interesting question is to determine how much further we can optimize the employed algorithms with respect to memory usage and speed.

References

1. Anderson, R., and Kuhn, M.: Tamper Resistance – A Cautionary Note. In: *Proceedings of the* 2^{nd} *Usenix Workshop on Electronic Commerce*, USENIX Association, Oakland 1996, pp. 1 – 11.
2. Anderson, R., Biham, E., and Knudsen, L.: *A Proposal for the Advanced Encryption Standard*. http://ftp.cl.cam.ac.uk/ftp/users/rja14/serpent.pdf. 1998.
3. Basagni, S., Herrin, C., Bruschi, D., and Rosti, E.: Secure Pebblenets. In: *Proceedings of the* 2^{nd} *International Symposium on Mobile Ad Hoc Networking & Computing*. ACM Press, Washington DC, 2001. pp. 156 – 163
4. Blom, R.: An Optimal Class of Symmetric Key Generation Systems. In: *Advances in Cryptology - EUROCRYPT '84* (LNCS 209). Springer-Verlag, Berlin 1985, pp. 335 – 338.
5. Blundo, C., Santis, A.D., Herzberg, A., Kutton, S., Vaccaro, U., and Yung, M.: Perfectly-Secure Key Distribution for Dynamic Conferences. In: *Advances in Cryptology - CRYPTO '92* (LNCS 740). Springer-Verlag, Berlin 1993, pp. 471 – 486.
6. Buschmann, C., Fischer, S., Luttenberger, N., and Reuter, F.: Middleware for Swarm-Like Collections of Devices. *IEEE Pervasive Computing Magazine*, Vol. 2, No. 4, 2003, p. 96.
7. Carman, D.W., Kruus, P.S., and Matt, B.J.: *Constraints an Approaches for Distributed Sensor Network Security*. Technical Report, NAI Labs 2000.
8. Chan, H., and Perrig, A.: Security and Privacy in Sensor Networks. In: *IEEE Computer*, Vol.36, No.10, October 2003, pp. 103 – 105
9. Chan, H., Perrig, A., and Song, D.: Predistribution Schemes for Sensor Networks. In: *Proceedings of the IEEE Security and Privacy Symposium*. IEEE Computer Society Press, Los Alamos 2003. pp. 197 – 213

10. Dworkin, M.: *NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation.* http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf. 2001.
11. Eschenauer, L., and Gligor, V.D.: A Key-Management Scheme for Distributed Sensor Networks. In: *Proceedings of the Conference on Computer and Communications Security '02.* Washington DC 2002. pp. 41 – 47.
12. Guajardo, J., Bluemel, R., Krieger, U., and Paar, C.: Efficient Implementation of Elliptic Curve Cryptosystems on the TI MSP430x33x Family of Microcontrollers. In: *Proceedings of the International Workshop on Practice and Theory in Public Key Cryptography '01* (LNCS 1992), Springer-Verlag, Berlin 2001. pp. 365 – 382.
13. Knuth, D.E.: *The Art of Programming: Vol.2 Seminumerical Algorithms.* Addison-Wesley, Reading, Massachusetts 1969.
14. Koberstein, J., Reuter, F., and Luttenberger, N.: The XCast Approach for Content-based Flooding Control in Distributed Virtual Shared Information Spaces – Design and Evaluation. In: *Proceedings of the* 1^{st} *European Workshop on Wireless Sensor Networks* (LNCS 2920), Springer-Verlag, Berlin 2004. pp. 188 – 203.
15. Liu, D., and Ning, P.: Establishing Pairwise Keys in Distributed Sensor Networks. In: *Proceedings of the Conference on Computer and Communications Security '03.* ACM Press, Washington DC 2003. pp. 52 – 61.
16. National Institute of Standards and Technology (NIST): *FIPS 197: Announcing the Advanced Encryption Standard.* http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. 2001.
17. Perrig, A., Szewczyk, R., Wen, V., Culler, D., and Tygar, J.D.: SPINS: Security Protocols for Sensor Networks. In: *Proceedings of the* 7^{th} *International Conference on mobile Computing and Networks.* ACM Press, Washington DC 2001. pp. 189 – 199.
18. Rivest, R.: The RC5 Encryption Algorithm. In: *Proceedings of the* 2^{nd} *Workshop on Fast Software Encryption* (LNCS 1008). Springer-Verlag, Berlin 1995. pp. 86 – 96.
19. Scatterweb Projekt: *Embedded Sensor Board ESB 430/1.* http://www.inf.fu-berlin.de/inst/ag-tech/esb/english/index.htm. 2004.
20. Solinas, J.A.: *Generalized Mersenne Numbers.* Technical Report CORR-9939, Dept of C&O, University of Waterloo, Canada, 1999.
21. Stajano, F., and Anderson, R.: The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In: *Proceedings of the* 7^{th} *International Workshop on Security Protocols* (LNCS 1796). Springer-Verlag, Berlin 2000. pp. 172 – 194.
22. Wood, A.D., and Stankovic, J.A.: Denial of Service in Sensor Networks. In: *IEEE Computer*, Vol. 35, No. 10, October 2002, pp. 54 – 62.
23. Wood, A.D., Stankovic, J.A, and Son, S.H.: JAM: A Jammed-Area Mapping Service for Sensor Networks. In: *Proceedings of the* 24^{th} *Real-Time Systems Symposium.* IEEE Computer Society Press, Los Alamos 2003. pp. 286 – 297.
24. Zhu, S., Setia, S., and Jajodia, S.: LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. In: *Proceedings of the Conference on Computer and Communications Security '03.* ACM Press, Washington DC, 2003. pp. 62 – 72.

A Implementation Details

This part illustrates a way to efficiently implement the Blundo-et-al.-scheme. Two concepts are combined: first, the Horner Rule minimizes the multiplication

effort and second, the choice of the field determines the speed of the modular reduction.

For an efficient implementation the *ID*'s can be reduced to a certain range, e.g. $0 < ID < 2^{16}$. If the evaluation is done using the Horner Rule (e.g. [13], see also figure 2) only multiplications between a short *ID* and a longer field element have to be done, which allows for a much faster modular arithmetic than the general case.

$$a_n ID^n + a_{n-1} ID^{n-1} + \ldots + a_0 =$$
$$(\ldots (a_n ID + a_{n-1}) ID + a_{n-2}) ID + \ldots) + a_0$$

Fig. 2. Horner Rule

The choice of the field F determines the range and the speed of the evaluation of the common secret. The scheme can be implemented using any field. Consequently, a field, which suits the existing hardware platform well, should be chosen. The arithmetic of $GF(p)$ is supported by the existing hardware multiplier of the MSP430 processor. Therefore, this choice is faster than an implementation using $GF(2^m)$.

The following example assumes that the identities are limited to $0 < ID < 2^{16}$ and illustrates the usage of a 80-bit Generalized Mersenne Prime, which allows for an efficient modular arithmetic [20], $p = 2^{80} - 2^{64} - 2^{32} - 1$. First a normal multiplication of a 16-bit identity and a 80-bit coefficient is calculated. Afterwards one can rearrange the 96-bit result $r = \sum_{i=0}^{5} r_i 2^{16i}$ to $s = \sum_{i=0}^{4} r_i 2^{16i}$ and $t = 2^{64} r_5 + 2^{32} r_5 + r_5$. From the special form of p follows that $s + t = r \bmod p$. The advantage is that the reduction modulo p can be calculated by a single 80-bit addition and at most 2 80-bit subtractions. This is substantially faster than any standard method that assumes two equally long operands.

Securely Propagating Authentication in an Ensemble of Personal Devices Using Single Sign-on

Prakash Reddy, Eamonn O'Brien-Strain, and Jim Rowson

HP Labs
{prakash.reddy, eob, jim.rowson}@hp.com

Abstract. More and more, people will continuously be using ubiquitously available networked computational devices as they go about their lives: small personal devices that they carry, appliances that they find in their surroundings, and servers in remote data centers. Some of the data exchanged by these devices will be private and should be protected. Normally to protect data, users would need to authenticate themselves with a device by *signing on* to it. However it will be physically impossible to sign onto devices that have limited or no user interface and even if they all had a sufficient user interface it will be an intolerable burden to have to sign on to each of many devices, particularly as the membership of the ensemble of devices continuously changes with the user's movements. Making authentication in this environment more difficult is the fact that these devices are usually connected in a personal area network that is neither secure nor reliable and uses a broadcast medium for communication. In this paper, we present a simple easy-to-use scheme that allows users to sign on to a single device and enable the rest of the devices connected in the personal area network automatically without requiring a central server or synchronized clocks. As well as being simple for the user, our solution is designed not only to prevent commonly used attacks like replay and man-in-the-middle but also to protect the user's data even if the devices are lost or stolen.

Keywords: Personal Area Network, Security, Single sign on, device ensemble.

1. Problem Statement

1.1 Objectives

We define objectives that are useful to measure the effectiveness of the solution under consideration. One of the key objectives is to support single sign-on (the ability of users to use a single authentication action to authenticate themselves with multiple computers) in an environment where there are no central servers or synchronized clocks. Another objective is to support a system where the device ensemble is constantly changing, that is new devices join, and older devices leave. The goal is to discover devices belonging to an ensemble by their presence in a personal area network (PAN) such as Bluetooth rather than through some pre-established addressing scheme. It should be possible for devices to join and leave a PAN at any time.

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 178 – 189, 2004.

As we are dealing with resource-limited devices, the goal is to minimize the number of the messages, the size of the messages and the amount of encryption and decryption required. An implementation can choose particular cryptographic algorithms that give a desired trade-off of security strength and computation time.

1.2 Requirements

Bird et al [2] have identified some of the common security weakness and described some requirements. For example the protocol should a) avoid allowing an attacker to launch a known text attack, b) prevent the execution of the protocol in parallel sessions, which prevents parallel session attacks, and c) prevent the attacker from using responses of one flow in another flow. In addition to the requirements mentioned above, we also require that replay attacks be prevented, that is if the same messages are replayed it should not result in successful execution of the protocol. As devices can enter and leave the PAN at any time, the protocol must run automatically to enable devices as they enter the PAN. Since devices join at any time, the authentication expiry times should be adjusted to be only the remaining time. Devices enabled automatically must be assigned a shorter authentication time and be required to re-authenticate frequently.

1.3 Assumptions

The devices are in an ensemble over which the user has physical control. They are usually connected via a PAN that is not secure, is not reliable, and has broadcast capability. There are no central online servers, and no device in the ensemble is special. The clocks on the devices are not synchronized. No member of the ensemble knows the identities of all the other ensemble members. Some devices have limited or no user interface.

The particular attacks we address in our security threat model are the *man-in-the-middle* attack, in which an attacker communicates with two devices pretending to each that it is the other, the *replay* attack, in which the attacker records the sequence of communication between two devices and replays it back later pretending to be one of the devices, and an attack in which an attacker *steals* a device from the ensemble (or finds a lost device) and takes it to a remote location to read its data.

Our proposal fits into the general goal to make consumer devices very easy to use while working well together. It makes the authentication of an ensemble of devices be as simple as the authentication of one device, while providing a high level of security.

2. Protocol

2.1 Authentication Propagation

Each device uses two keys, an *ensemble key*, which is used to prove ensemble membership, and an *enabling key*, which gives access to encrypted data stored on

the device. Before the user has logged in, the ensemble key is stored and usable on every device but the enabling key is in encrypted form, protected by a password or PIN.

A device can gain access to the enabling key in two ways. Either the user's manual sign-on gives the key to the device or our new single sign-on protocol automatically propagates the key to the device. Once the user logs on to one device it in turn enables other devices in the ensemble, which can themselves, continue to enable yet more devices in the ensemble.

2.2 Architecture

Our overall solution comes in four parts:

Key distribution uses a trusted key-coining device and a secure connection to distribute an ensemble key and an encrypted enabling key to the ensemble of devices. The enabling key is shared by all members of the ensemble, but is encrypted uniquely and for each device.

Second factor authentication allows users to decrypt the enabling key on one of their devices using a password or equivalent. (The first factor is physical possession of the device.)

A *data sharing mechanism* uses the enabling key to transfer data securely between the devices.

Finally, single sign-on *authentication propagation* gets the enabling key to the rest of the devices in the ensemble without user intervention.

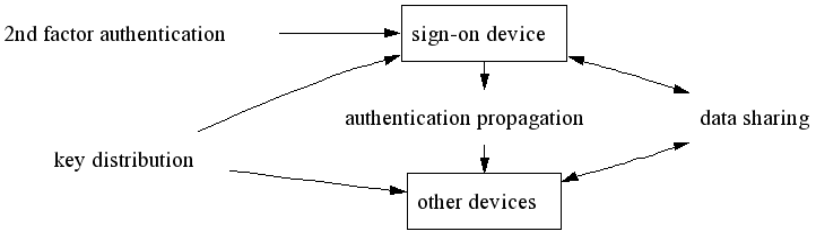


Fig. 1. Overall Solution

2.3 Advantages of Single Sign-on

The single sign-on simplifies the process of securing an ensemble of devices and the data stored on them, propagates authentication to all member devices of an ensemble connected in a PAN with a single user sign-on. It limits the damage incurred in case a device is lost or stolen, allows devices with no user interface to be protected, and automatically extends the enabled status of all devices as long as the original sign-on device is nearby.

2.4 Overview of Protocol

The user initially enable their ensemble by choosing one of their devices that presents a user interface for entering a password or some other second factor of authentication.

This decrypts the enabling key stored on the device, which automatically triggers activation of the single sign-on protocol that propagates the authentication to all devices in the ensemble that are reachable on the PAN.

Each enabled device periodically runs the protocol round specified below, taking the role of Alice, with all other reachable devices in the ensemble taking on the role of Bob.

At the end of a protocol round an enabling key has been securely transferred to the Bob device, thus rendering it enabled.

2.5 Protocol Messages

The single sign-on protocol consists of three messages exchanged between an enabled device and a non-enabled device. We dub these Alice and Bob respectively. Both machines have a copy of the ensemble key (the shared secret from the key distribution mechanism) and Alice has a copy of the enabling key (either from second factor authentication or by playing the role of Bob in a previous round of this protocol).

1. Alice creates a random nonce (use-once value to act as a challenge to the receiver) and sends it in a message broadcast to the network along with Alice's network ID. The message is encrypted and authenticated with the ensemble key.
2. Bob sends back to Alice a reply that contains the received nonce (a reply to Alice's challenge), a new nonce that it randomly generates (a challenge to Alice), and its current local time. The message is encrypted and authenticated with the ensemble key. The authentication also includes the authentication code of the first message.
3. Alice verifies that it got back its original nonce and replies to Bob with a message that contains Bob's nonce and the enabling key. It also contains a disabling key and the sign-on expiry time expressed relative to Bob's local time. The message is encrypted and authenticated with a single-use key that both Alice and Bob can independently generate by using the ensemble key to hash the combination of the authentication codes of the two previous messages. The authentication of this message also includes the authentication of the previous messages.

Bob, after verifying that it got back its nonce, ends up with the enabling key. It then switches to the Alice role and starts periodically running the protocol to enable devices not reachable from the original device, or which enter the network later.

2.6 Protocol Details

In the protocol description above the operator || means concatenation. Symbol K_G is the ensemble (group) key, K_E is the enabling key, K_D is the disabling key, and k_{AB} is a

Table 1. Protocol

Alice		**Bob**
Known: (I_A, K_G, t_E, $\mathbf{K_E}$) Possibly Known: (K_D) If K_D not known $K_D \leftarrow$ random() $n_A \leftarrow$ random() $m_1, a_1 \leftarrow$ pack(K_G, n_A‖I_A)		Known: (I_B, K_G)
The device Alice is enabled. Device Bob is listening for an enabling message. Both devices have a shared secret ensemble key, and each has a unique ID. Alice generates a random nonce and if necessary generates a disabling key.		
	"enable", m_1, a_1 $\longrightarrow$ *broadcast*	
Alice uses the ensemble key to broadcast securely an enabling message containing its ID and nonce		
		n_A‖$I_A \leftarrow$ unpack(K_G, m_1) $n_B \leftarrow$ random() $t_B \leftarrow$ localTime() $m_2, a_2 \leftarrow$ pack (K_G, n_A‖n_B‖t_B, a_1)
Bob receives and authenticates the message. It generates its own random nonce and determines its current time		
	I_B, m_2, a_2 $\longleftarrow$ *to I_A*	
Bob uses the ensemble key to send a reply containing both nonces and Bob's current time. The authentication accumulates.		
n_A'‖n_B‖$t_B \leftarrow$ unpack(K_G, m_2, a_1) assert($n_A' = n_A$) $k_{AB} \leftarrow$ H(K_G, a_1‖a_2) $t_E' \leftarrow t_B + t_E$ – localTime() $m_3, a_3 \leftarrow$ pack(k_{AB}, n_B‖t_E'‖K_D‖$\mathbf{K_E}$, a_2)		$k_{AB} \leftarrow$ H(K_G, a_1‖a_2)
Alice receives and authenticates the message. Both devices use the ensemble key and the message authentication codes to generate independently the same session key. Alice calculates expiry time as an offset from Bob's current time.		
	I_A, m_3, a_3 $\longrightarrow$ *to I_B*	
Alice replies by using the session key to send back the enabling key, the disabling key, and the expiry time. Authentication accumulates.		
		n_B'‖t_E‖K_D‖$\mathbf{K_E}$ $\leftarrow$ unpack(k_{AB}, m_3, a_2) assert($n_B' = n_B$)
Bob receives and authenticates the message. It now has the enabling key and so is an enabled device.		

single-use session key. All the keys are for a symmetric cipher. The symbols n_i, are nonces, m_i are encrypted messages, a_i are message authentication codes (MACs), I_d are unique device IDs used for network addressing, t_B is current local time on Bob, t_E is the *near-expiry* time which is described in a later section of this paper. The operation E(*k*, *x*) is encryption of *x* using key *k*, D(*k*, *m*) is decryption of *y* using key *k*, and H(*k*, *x*) is a MAC function of *x* using key *k*. Finally we define some functions that combine encryption and authentication of messages transferred between devices:

pack(k,x)	::=	return (E(k,x), H(k,x))
pack(k,x,c)	::=	return (E(k,x), H(k,x‖c))
unpack(m,a)	::=	x ← D(k,m); assert(a=H(k,x)); return x
unpack(m,a,c)	::=	x ← D(k,m); assert(a=H(k,x‖c)); return x
random()	::=	A pseudo random number generator
localTime	::=	A method that returns the current time on the device
assert()	::=	Ensure that the conditional expression is true

2.7 Analysis

The choice of message flow, the number of messages, the content of the messages have been chose carefully to satisfy all of the requirements. The protocol is always initiated by an enabled device. One key assumption we make is that the devices are tamper proof and that the protocol running on the device is trusted.

This prevents attackers from launching known text attacks. If an attacker broadcasts a message pretending to be an enabled member of the ensemble, the protocol would be aborted by all recipients (message will not be a valid one because the attacker does not possess the ensemble key).

Replay attacks are also not possible because the key used to encrypt/decrypt message 3 is session dependent and is a single use key. Even if the message is played back to the same host, the expiry time included in message 3 would prevent the device from being enabled.

Parallel session attacks are not possible as we carefully designed the protocol such that there is at most one instance of the protocol running per device.

Since we are dealing with resource constrained devices, it was important to limit the number of messages required to carry out the authentication. Only three messages are used to accomplish mutual authentication and key exchange.

The protocol is designed so that the expiry time is computed in terms of the destination device local time without requiring a synchronized or global clock.

2.8 Time Out

In our single sign-on scheme, the expiration of authentication is a critical feature and is helpful to minimize how long data is vulnerable in case a device is stolen or lost.

The aim of the single sign-on scheme is to provide security and at the same time make it convenient for people to use an ensemble of devices. The user explicitly authenticates with a sign-on device, which can be any device with a user interface that allows the user to authenticate. Explicit authentication in this context involves the user manually entering the second authentication factor associated with the device. The sign-on device gets a *far-expiry* time, which typically ranges from an hour to a

day, and which the user can customize. All automatically authenticated devices will have a *near-expiry* time in the order of minutes to hours. Each automatically authenticated device would be required to re-authenticate much more frequently than the sign-on device. Devices re-authenticate themselves automatically after the near-expiry by participating in the single sign-on protocol.

2.8.1 Far-Expiry

When the user initially signs on to a device, the device sets the far-expiry, and it initiates the single sign-on protocol. This device is responsible for generating periodic near-expiry times. All times are device specific and do not rely on a global synchronous clock.

2.8.2 Near-Expiry

Since devices may join and leave the PAN, we have introduced the notion of near-expiry time, written as t_E in the protocol description in a previous section, which allows the devices in the network to be authenticated for a shorter time than the sign-on device. The sign-on device is responsible for assigning the near-expiry. The remaining authentication time is used to computing the new near-expiry. For example if at 1:00 PM, in the Alice device's local time, the near-expiry is 1:30 PM and Bob device joins the network later at 1:10 PM, the Alice device sets Bob's near-expiry to be 20 minutes later in Bob's local time. Recall Bob sends its local time as part of message two of the protocol. Each automatically authenticated device then has its own version of the near-expiry, computed based on device authentication time.

The expiry time received by Bob may be slightly inaccurate because of network delays and other latencies in the protocol implementation. However, any inaccuracy will never result in Bob's expiry time being later than it should be.

The scheme allows computation of near-expiry in terms of Bob's clock. It gives a notion of a global clock without requiring a true global synchronous clock.

Figure 2 shows the different states a device may go through. In the top state, the device is acting in the Bob role of the sign-on propagation protocol while in the right and left states it is acting in the Alice role. When first turned on the device is in the top Bob state.

If it is a sign-on capable device and the user signs on to it then it moves to the right-hand Alice state setting the far-expiry (T_E) to a fixed time (ΔT) in the future and the near-expiry (t_E) to a shorter fixed time (Δt) in the future. As part of its Alice role, it then starts attempting to propagate the near-expiry time to other devices. Every time the near-expiry time arrives, it sets it again to the same short fixed time in the future. When the far-expiry time arrives, the device moves back to the top Bob state waiting to be authenticated by the user or an Alice device.

Alternatively, when it is in the top Bob state it can get the protocol messages from another device in the Alice role from which it receives the near-expiry time. It moves to the left-hand Alice state until the near-expiry at which time it transitions back to the top Bob state.

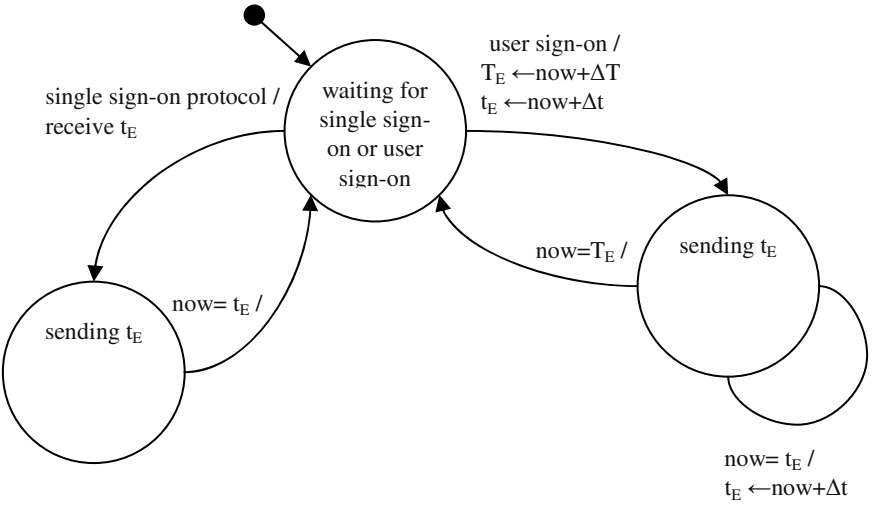


Fig. 2. State Transition Diagram of a Participating Device

Figure 3 shows a timing diagram example for three devices. Time increases from left to right. The heavy black lines indicate the periods when the devices are authenticated and acting in the Alice role. The sign-on device is authenticated from the time the use signs in until the far-expiry time. Authentication propagates to P across the single sign-on protocol and lasts until the initial near-expiry time. After the P's authentication expires, it participates in the single sign-on protocol again and gets a new near-expiry time. This repeats (five times in the above example) until the far-expiry time after which no further authentication arrives from the sign-on device. Meanwhile Q can "see" P on the network but cannot see the sign-on device, so it gets its authentication indirectly via P. In this example, Q moves out of network range for a while and so misses some of the protocol messages, resulting in a gap in its authentication, thus protecting its data from someone who might have stolen it and removed it from the network.

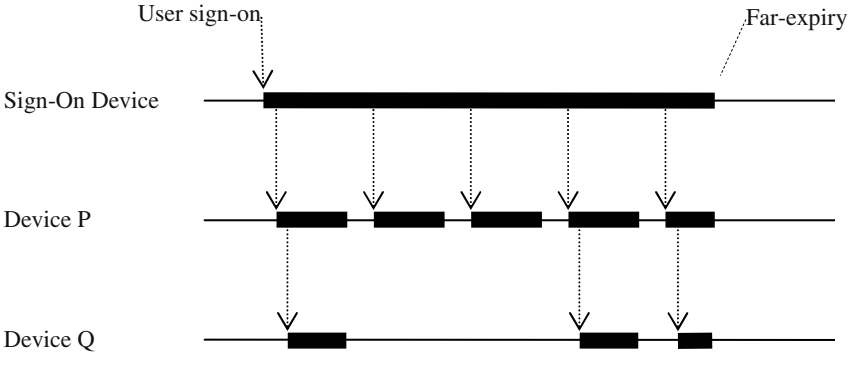


Fig. 3. Example Timing Diagram

3 Related Work

The single sign-on protocol we proposed in this paper is novel, because it addresses several issues as part of a single solution. There is considerable body of work which addresses specific issues we have highlighted, however none of them address all of the issues at once. We look at proposal which offer solutions to particular aspects of our requirements and identify the similarities and the differences. The three main areas of work that relates to our proposal include a) Single sign-on where in a user signs-on to a single device and the authentication propagated to all other related services. b) Key exchange over insecure networks and c) Minimizing the user interaction in authenticating with a network of devices. Most of these assume a model where there is either a single central server or the network configuration is known or there exists a synchronized monotonic clock.

3.1 Single Sign-on

Kerberos [7] is an example of a system where users provide a password and receive a ticket in exchange. The ticket can be used to authenticate users to different network services. Kerberos single sign-on is possible because all of the services are under the same administrative control. There is a centralized database containing keys that are shared with each service, and tickets can be issued, encrypted under the keys of the target services. Kerberos suffers from several limitations especially when it is be implemented in an un-trusted, insecure public networks. The limitations of the Kerberos authentication protocol are discussed in detail in [8]. In addition to the limitations, the Kerberos protocol is not suitable for our purposes because it requires a synchronized clock among all the participating devices and assumes that there is a central server (authentication server) which is responsible for generating the session key that is to be used by the client and verifier. In our system, all of the participating devices are peers and they are discovered dynamically and there is no central server or a need for a synchronized clock.

Passport from Microsoft is yet another single sign-on service which allows users to gain authenticated access to multiple, independent web services that are under completely different administrative control.

A client's interaction with the Passport service begins when they visit a merchant site, the merchant website re-directs the request to a well-known Passport server. The user is connected to the Passport server over an SSL link and logs into the Passport server. Once the user has been authenticated by the Passport server, the user is re-directed back to the merchant site. An encrypted version of the authentication information is stored on the users computer, so when the user visits other merchant sites, the authentication of the user is performed without requiring the user to enter the login information. A detailed discussion and limitations of the passport service are in [9]. The key limitations of this approach in relation to our system are a) the system depends on a centralized Passport server for client authentication. This is not suitable for a peer oriented ad-hoc network. b) Key management is another issue, the Passport service and the merchants must share a key. These shared keys are distributed out of band, which has its own limitations. In our proposal the keys are distributed out of band, but are done explicitly by the user under their control c) Passport service uses persistent cookies to avoid having users to reenter the login information. In our

system, if the device used for the explicit authentication leaves the communication range the rest of the devices will be disabled as soon as the time out expires, if at a later stage the primary device comes back into the range the devices get re-activated without user intervention assuming the primary device is still enabled.

3.2 Key Exchange over Insecure Networks

Encrypted Key Exchange (EKE) is a combination of asymmetric (public-key) and symmetric cryptography that allow two parties sharing a common password to exchange confidential and authenticated information over insecure network. The EKE protocol is secure against active attacks and is protected against off-line "dictionary" attacks. It is designed for a client-server interaction in which the client and the server share a password. It addresses a common problem of two-party key exchange protocols: an attacker can obtain the long-lasting secrets using an off-line brute-force attack. A sufficiently long random string will be resistant to such an attack but many protocols have secrets that are user-chosen easily cracked passwords [3, 4]. In our system, we use a password to prevent access to the device, but the password is never used in the protocol. Brute-force offline attacks are not possible in our system as the weaker user chosen password never leaves the device. All members of a device ensemble in our system share a secret that is made up of long random strings which makes brute-force offline attacks ineffective. Unlike EKE our solution does not require the more expensive asymmetric encryption during the authentication phase. As discussed in the description section above, our protocol is safe against active attacks. Given that our solution is intended for resource limited devices, the algorithms used, the number of messages and the size of the messages have a significant impact on the adoptability of the proposal. By using only symmetric encryption and only three messages as opposed to six for EKE, we have satisfied this constraint without compromising the security of the overall system.

3.3 Minimal User Interaction

Given the premise that users carry multiple devices, the user interaction required to enable an ensemble of device should be kept to a minimum, other wise the tendency of the user would be to disable the authentication system in its entirety. The proposal made by the authors of "Zero-Interaction Authentication" (ZIA) [5] is very interesting, in that their stated goal is to prevent exposing sensitive data stored on laptops in case of theft. The ZIA proposes to protect the data by storing all of the data on the device in encrypted form, and when ever the rightful owner needs access to the data it is decrypted. In order for this to work properly they propose that the owner wear a small authentication token. This token gives decryption authority to the laptop, communicating over a short-range, wireless link. The decryption is possible only when the user is within the short communication range, effectively locking the data if an attacker steals the laptop. Tying the wearable authentication token to a device for enabling and disabling it is attractive. However if the token is misplaced or forgotten, the authorized user has no way to access the data. Also the proposal is geared at protecting a single device. In our system we have the need to authenticate multiple devices and the set of devices involved is dynamic. However it is possible to incorporate ZIA into our system as a way to enable the primary device and avoid the manual login process. This would make the entire system free of manual human interaction.

4 Status and Next Steps

We have implemented the basic protocol and tested it on a configuration of three devices. The implementation was based on Rendezvous running on an IP subnet. We can plug in different cryptographic algorithms: in the initial version we used an SHA1-based pseudo-random number generator, DES for symmetric encryption, and HMAC-SHA-1 for message authentication.

Our plan is to finish the implementation of the timeout mechanism, integrate the key distribution module, and benchmark the latency involved in enabling the devices. We will integrate all of this into an existing prototype infrastructure for running distributed applications on an ensemble of intermittently connected devices. We will do usability testing to verify that we are putting minimal burden on the user and to provide feedback on the best values for various parameters of the algorithm such as the expiry times and the types of second-factor authentication. We will also submit our detailed specification of the protocol for review by security experts to give more confidence in our security property claims.

5 Conclusion

Our protocol has security properties that address our target threat models. An attacker cannot replay the messages to enable a stolen device because each participant generates a random nonce on each round that is only valid for that round. The protocol prevents man-in-middle attacks, by using symmetric (secret-key) encryption to encode messages, maintaining freshness and by limiting the execution of the protocol to a single instance on the device to be enabled. The timeout mechanism that bounds the amount of time a particular device is enabled reduces the vulnerability of the device in case of theft or loss. Offline brute force attacks are prevented primarily by using long random sequences and encrypting multiple objects within a single message. The shared key in our case is a symmetric key that is long enough that it is effectively impossible to guess. A device can be compromised if the device is stolen and the attacker gains access to the second factor used to protect it. However, the damage in this case is limited to the data stored on that stolen device.

Our usability requirements are also met. The protocol allows a single sign-on to authorize an ensemble of devices and helps secure devices that do not have any user interface to implement a login scheme. Users could use a single device to login to the ensemble of devices and they will be enabled as long as the user is nearby, however when the user moves away from a subset of the devices, they will be disabled as soon as they expire.

References

[1] M. Bellare and P. Rogaway. "Entity Authentication and Key Distribution". *Advances in Cryptology - Crypto '93, Lecture Notes in Computer Science 773*, pp. 232- 249

[2] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Lolva and M. Yung, "Systematic design of two-party authentication protocols". *Advances in Cryptology. Proceedings of Crypto 91*, Springer-Verlag, 1991

[3] S. M. Bellovin, M. Merritt. “Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks”. *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, May 1992.

[4] S. M. Bellovin, M. Merritt. “Augmented Encrypted Key Exchange: a Password-Based Protocols Secure Against Dictionary Attacks and Password File Compromise”. *Proceedings of the 1st ACM Conference on Computer and Communications* Security, 1993.

[5] M. Corner and B. Noble. “Zero-interaction authentication”. *Proceedings of the Eighth International conference on Mobile Computing and Networking (MOBICOM)*, pages 1--11. ACM, 2002

[6] T. Woo and S. Lam, “A semantic model for authentication protocols,” *Proceedings 1993 IEEE Symposium on Research in Security and Privacy*, May 1993.

[7] J.G. Steiner and B.C. Neuman and J.I. Schiller. "Kerberos: An Authentication Service for Open Network Systems". *Usenix Conference Proceedings*, 1988.

[8] S. M. Bellovin and M. Merritt. “Limitations of the kerberos authenication system.” *Computer Communication Review*, 20(5):119-132, October 1990.

[9] David P. Kormann and Aviel D. Rubin. “Risks of the Passport Single Signon Protocol.” *ComputerNetworks*, Elsevier Science Press, volume 33, 2000.

Key Management in Wireless Sensor Networks

Yann-Hang Lee[1], Vikram Phadke[2], Amit Deshmukh[3], and Jin Wook Lee[1]

[1] Arizona State University,
Tempe, AZ 85281, USA
{yhlee, Jinwook.Lee}@asu.edu
[2] Qualcomm USA,
San Diego, CA 92121, USA
vikram@qualcomm.com
[3] Siemens USA,
San Diego, CA 92126, USA
Amit.Deshmukh@icm.siemens.com

Abstract. Wireless sensor networks hold the promise of facilitating large-scale, real-time data processing in complex environments. As sensor networks edge closer towards widespread deployment, security issues become a central concern. Key management in wireless sensor networks is a challenging problem. Computationally complex asymmetric crypto techniques are unsuitable for use in resource-constrained sensor nodes and use of symmetric cryptography makes the entire network vulnerable in the event of node compromise. This paper presents a key management scheme that satisfies both operational and security requirements of distributed wireless sensor networks. The proposed scheme accommodates other techniques like data aggregation for energy efficiency and watchdog mechanism for intrusion monitoring. Also, the proposed scheme is generic enough so that it can be applied to variety of sensor network protocols.

1 Introduction

Wireless sensor networks can be used for wide range of applications including health, military and security. Realization of these and other sensor network applications re-quire wireless ad hoc networking techniques, although many protocols and algorithms proposed for ad hoc wireless network are not well suited for sensor networks because of sensor node and network constraints [1].

As sensor nodes might be deployed in an unattended and unsecured terrain, it is possible for an adversary to obtain data that sensor nodes transmit to the base station. In unsecured environment, sensor networks are so vulnerable that adversary can attack routing messages during network initialization phase and disrupt the whole network before it even starts functioning.

Providing confidentiality and authentication is critical to prevent adversary from compromising the security of a distributed sensor network. However, providing key management for achieving confidentiality and authentication is difficult due

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 190–204, 2005.

to the ad hoc nature, dynamic topology and resource limitations of the sensor network environment.

Apart from basic functionality, some sensor networks apply techniques like data aggregation for improving energy efficiency or watchdog mechanism for enhancing security. As these techniques are gaining importance, design of key management algorithm should facilitate these approaches. Also, sensor networks tend to be application specific, with different routing and data forwarding protocols. Therefore, key management algorithm should be generic so that it can be applied to wide range of sensor networks.

Other than providing basic security features such as confidentiality and authentication, characteristics of sensor networks place additional requirements on key management scheme. Key management scheme should be flexible enough to accommodate ad hoc nature of sensor networks and it should be scalable to handle large number of nodes typically associated with sensor networks. It should also be able to provide good support for post-routing establishment and should have provisions for re-keying to guarantee key freshness.

Our research work proposes key management scheme that tries to address the issues mentioned above. We propose a combined key management approach that is more secure than the current implementation of key management scheme.

2 Background

2.1 Wireless Sensor Node

For the purpose of this research, we used Berkeley's motes as the required hardware and software for these nodes have been in fairly developed stage. University of California, Berkeley provides developmental support [2], while Crossbow Technologies, Inc. [3] makes these nodes commercially available.

2.2 Assumptions About Wireless Sensor Network

Sensor nodes are randomly deployed in the environment and all nodes report to single destination, called the base station. Though sensor nodes are resource-constrained in terms of computational power and energy, the base station is a computer system that is assumed to have unlimited capability. Sensor nodes are very much prone to get at-tacked and compromised, but we assume that the base station is trustworthy.

We assume that they use distributed flooding routing protocol to form the network topology. The base station initiates the routing discovery by broadcasting a beacon message. Each node on receiving beacon message updates the neighbor information and broadcasts the beacon message further.

Routing discovery phase, also called as beaconing phase, is followed by data for-warding phase, in which all sensor nodes forward sensor data back to the base station. As communication range of sensor node is limited, data is transmitted from the source node to the ultimate destination, the base station through number of intermediate nodes. Neighbor information gathered in beaconing phase

is used to support such multi-hop routing. Channel reliability is not considered for beaconing as well as for data forwarding phase. We assume communication medium to be reliable.

We assume that sensor nodes are not mobile. Though they are deployed randomly; once placed at a particular location, they do not tend to move from that location. But it is dynamic in nature as new sensor nodes may be added after network formation or some of the nodes may die down due to energy exhaustion or malfunction. This causes change in neighbor information and overall network topology. To gather this changing topology, beaconing phase is carried out periodically.

Data Aggregation and Packet Combining. Sensor networks are distributed event-based systems that have certain characteristics that differ from traditional communication networks. Some of the characteristics from data flow point of view are redundant data flow and many-to-one (nodes to the base station) communication paradigm. There are two simple rules that allow two packets with the same destination to be combined to form a single packet, should they meet at a node. For example, the data fields carried by the two packets may be physically appended together, or two application-related packets may be aggregated. Packet Combining is a technique of physically combining several data messages into a single data message. Also, data aggregation is a technique of combining the data coming from different sources enroute, thus eliminating redundancy and saving energy by minimizing the number of transmissions. Instead of sending data in various packets, sensor nodes might aggregate them and send the combined data as a single packet. In this paper, we use the term 'data aggregation' for both techniques. As data communication involves heavy energy consumption, reducing the number of data transmissions helps in conserving significant energy for already energy-constrained sensor nodes. Thus, we assume that when data is forwarded to the base station, some of the intermediate nodes will participate in data aggregation. Though criterion for the selection of aggregating node is outside the scope of this research work, we assume that parent node determines child-level node's participation in data aggregation in beaconing phase and such aggregating node aggregates the received data before forwarding it to the parent node.

Watchdog Mechanism. A node may behave maliciously by receiving the data packets; but dropping them or modifying them before forwarding. Watchdog mechanism is one of the solutions to identify such malicious nodes. When a node forwards a packet to the parent node, the node switches its state to watchdog mode and verifies that the parent node in the path also forwards the packet correctly to the grand-parent node. As the sensor node communication is inherently broadcast in nature, the watchdog node can listen promiscuously to the next node's transmissions. If the parent node does not forward the packet correctly, then it can be said to be misbehaving. Such misbehavior is considered as intentional malicious activity. As a response mechanism, a node can change the parent node, as required information is available with the node during beaconing

phase. In presence of cryptographic techniques, additional complexity arises due to the fact that parent node may forward the data using different encryption key. In this case, watchdog node must know the key that parent node uses for encryption.

3 Key Management

This section describes the proposed key management algorithm for sensor network. The aim of the algorithm is to make sensor network more secure by using combined key propagation techniques so that it is difficult for any adversary to perform malicious activities like reading or modifying in-transit data. For this, we make certain assumptions about the sensor network as specified in earlier section.

3.1 Algorithm Initialization

Information at Sensor Node. Each sensor node must be equipped with algorithm related cryptographic data. This data can be embedded into ROM of each sensor node during manufacturing phase. Thus, this node-related data is not changed during the operation. This information includes:

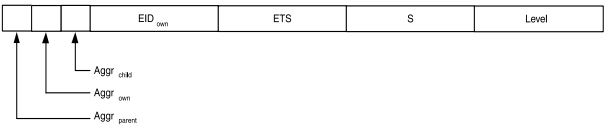
- K_{pb}: Pubic key of the base station
- ID: Identity of each node
- EID: Encrypted identity of each node. This is calculated by using private key of the base station. As only the base station has this private key, no one else can compute EID from given ID.
- CONST: A constant number
- RAND: A random number for each node. (This number may be duplicated in some other random node)
- F_{seed}: One-way function that takes old seed S as input along with CONST or RAND; and calculates new seed S_{new}. As this is one-way function, computing old seed from new seed is difficult task.
- F_{key}: One-way function that takes seed S as input parameter along with CONST or ID to calculate set of keys for each node.

Beacon Message. Most of the routing protocols for sensor networks require that each node maintain neighbor information. This distributed neighbor information at each node forms the network topology as a whole. This is done using beacon messages. Initially, the base station creates and broadcasts a beacon message. Each node, on receiving a beacon message, notes the ID of the sender of the message. Each node broadcasts its only beacon message after receiving the first beacon message; thus eliminating duplicate beacons.

Generally, the base station initiates the beaconing phase and it is done periodically. Because of its periodic nature, newly added nodes become part of the network in the next beaconing phase, while dead nodes (possibly because of energy exhaustion or any other malfunction) are removed from the network. The

frequency of beaconing phase depends on the dynamic nature of the network. If network topology changes quite rapidly, beaconing phase should be carried out more frequently so as to gather new changed network topology. As beaconing phase involves broadcasting messages, it has high communication overhead in terms of node-energy. Therefore, if the net-work topology is less likely to be changed, then beaconing frequency should be low enough.

As these beacon messages are used for collecting neighbor information, only ID of sender node needs to be included. Routing protocols can pass other necessary routing information through these beacon messages. Our approach also makes use of beacon messages to pass algorithm-related data to all nodes. We use beacon messages to generate session keys between pair of nodes. For this purpose, we add several fields into beacon message format. Beacon message is modified by adding these fields as follows:



The first three Boolean fields are for notifying data aggregation.

- $Aggr_{parent}$: This Boolean field indicates whether sender node's parent is going to aggregate incoming data. True indicates that parent node is going to participate in data aggregation, while false indicates it is not.
- $Aggr_{own}$: This Boolean field indicates whether sender node is going to participate in data aggregation or not.
- $Aggr_{child}$: This Boolean field indicates whether child node of sender node should participate in data aggregation or not.
- EID_{own}: Encrypted ID of sender node. This is used for node ID authentication.
- ETS: Encrypted Time Stamp. This is used for beacon authentication.
- S: Seed Value. The base station initially generates a random seed. Each node then computes the new seed using F_{seed} function.
- Level: Level of the node. This value indicates the number of hops between the node and the base station. For any node, this is useful in determining parent, sibling and child level nodes. For node at level n:
  - All neighboring nodes at level n-1 act as parent-level nodes.
  - All neighboring nodes at level n act as sibling-level nodes.
  - All neighboring nodes at level n+1 act as child-level nodes.

We assume no node can decide of its own whether it wants to participate in data aggregation or not. For each node, it is decided by its parent node. This decision may be based on various criteria like network limit on number of data aggregating nodes, number of non-aggregating nodes between two aggregating nodes, frequency of in-coming data, data load on network path, etc. This decision criterion is out of scope of this research work. For the purpose of this

research, parent node randomly decides about child node's participation in data aggregation.

EID is stored in each node, which is calculated using private key of the base station. As only the base station has this key, no one can calculate EID using known ID. We also assume that each ID is formed based on certain rule (for example, each ID starts with letter 'S'). Therefore, only valid EIDs would be decrypted to valid IDs using public key of the base station. Any other random EID would be decrypted to some garbage ID.

At each beaconing phase, the base station encrypts current timestamp to ETS using its private key. All other nodes can check ETS by decrypting it using the base station's public key. As only the base station can generate valid ETS, any attempt by adversary to initiate beaconing phase would be detected, as nodes cannot decrypt malicious beacon's ETS to valid timestamp.

Apart from establishing network topology, we use beaconing phase to establish neighboring relationship and to authenticate participating nodes. Thus, no other foreign node or unauthenticated node can become part of the network topology after beaconing phase, as no other node in the network has any information about such node.

3.2 Key Management

Beaconing phase is also used for distributing appropriate key generation information in the network. Seed value is used in the computation of set of keys to be used for data confidentiality between pair of nodes in data forwarding stage. This appropriate seed value is passed to nodes through beacon messages and then, required keys are computed.

Seed Calculation. The value of seed computed at each node and the seed value forwarded at each node is dependent on the participation of the node and its parent node in data aggregation. If the node is not participating in data aggregation phase, then it should not be involved in data encryption-decryption and should just forward the encrypted data received from child node to its parent node. As seed is used for calculating the key for data confidentiality, non-aggregating node should just forward the received seed so that aggregating child (or descendant) node and aggregating parent (or ancestor) node can synchronize their key-values for data encryption-decryption.

The following table describes how seed value is computed depending on the node's and its parent's participation in data aggregation. For any node, these values are avail-able as $Aggr_{own}$ (indicates parent node's data aggregation participation), $Aggr_{child}$ (indicates node's data aggregation participation) and S1 (indicates received seed) in first received beacon message.

Use of one-way function ensures that it is difficult to get previous values of seed from the new value. But one can compute forward values of seed from the given value. When two consecutive nodes in the hierarchy (for example, a node and its parent-level nodes or child-level nodes) participate in data aggregation, new seed is computed as a function of received seed and random number. This

$Aggr_{own}$	$Aggr_{child}$	Seed Received	Seed Forwarded	Seed Calculation
0	0	S1	S1	
0	1	S1	S2	$S2=F_{seed}(CONST, S1)$
1	0	S1	S1	
1	1	S1	S2	$S2=F_{seed}(RAND, S1)$

Fig. 1. Calculating Seed

introduces randomness in the seed value calculation, which makes calculation of forward seed values difficult.

Key Calculation. We introduce three key sets for each node:

- Node uses K_{child} to decrypt the data it receives from particular child node. Thus, number of these keys is equal to number of child-level nodes of this node.
- Node uses K_{own} to encrypt the data that it wants to forward to parent node. This is the only key the node uses to encrypt the data.
- Node uses K_{parent} to decrypt the data the parent node forwards. (for watchdogging) Thus, number of these keys is equal to number of parent-level nodes of this node.

a) Own Key Whenever a sensor node receives first authenticated beacon message, it calculates its own key K_{own} based on contents of the beacon message. For any node, required values are available as $Aggr_{own}$ (indicates parent node's data aggregation participation), $Aggr_{child}$ (indicates node's data aggregation participation) and S1 (indicates received seed) in first received beacon message.

$Aggr_{own}$	$Aggr_{child}$	Seed Calculation	K_{own} of the node
0	0	$S2=F_{seed}(CONST, S1)$	$F_{key}(CONST, S2)$
0	1	$S2=F_{seed}(CONST, S1)$	$F_{key}(CONST, S2)$
1	0	$S2=F_{seed}(CONST, S1)$	$F_{key}(CONST, S2)$
1	1	$S2=F_{seed}(RAND, S1)$	$F_{key}(ID_{own}, S2)$

Fig. 2. Calculating Own Key

When two consecutive nodes in the hierarchy (for example, a node and its parent-level nodes or child-level nodes) participate in data aggregation, new seed is computed as a function of received seed & random number and K_{own} is calculated as a function of computed seed & ID of the node. Whenever a node generates any data to send or it is necessary to encrypt the data received from a child node to forward, it uses K_{own} to encrypt the data. This is the only key the node uses for encryption.

b) Parent Key On receiving first authenticated beacon message, node marks sender of the message as its parent node and calculates the parent key K_{parent}

based on contents of the beacon message. For any node, required values are available as $Aggr_{parent}$ (indicates parent of parent node's data aggregation participation), $Aggr_{own}$ (indicates parent node's data aggregation participation) and S1 (indicates received seed) in first received beacon message.

Aggrp $_{arent}$	Aggr $_{own}$	Seed Calculation	K$_{parent}$ of the node
0	0	S2=F$_{seed}$ (CONST, S1)	F$_{key}$ (CONST, S2)
0	1		F$_{key}$ (CONST, S1)
1	0	S2=F$_{seed}$ (CONST, S1)	F$_{key}$ (CONST, S2)
1	1		F$_{key}$ (ID$_{parent}$, S1)

Fig. 3. Calculating Parent Key

If a node has multiple parent-level nodes, number of K_{parent} keys is equal to the number of parent-level nodes. But the node always computes one parent key for a particular parent node. Initially the node marks sender of first authenticated beacon message as its parent node and computes K_{parent} for that parent node. When the node decides to change the parent node and selects one of parent-level nodes as its new parent node, it computes new K_{parent} for the new parent node.

c) Child Key After receiving first beacon message, node uses other received beacon messages to gather neighbor information. Whenever the node receives authenticated beacon message, it stores aggregation field values, plaintext and encrypted ID, level and seed received. Depending on the level, node (at level n) classifies neighbors into parent-level nodes (with level n - 1), sibling-level nodes (with level n) and child-level nodes (with level n + 1). On receiving beacon message from child-level nodes, the node calculates child key K_{child} based on contents of the beacon message. For any node, required values are available as $Aggr_{parent}$ (indicates parent of child node's data aggregation participation), $Aggr_{own}$ (indicates child node's data aggregation participation) and S1 (indicates received seed) in the received beacon.

Aggrp $_{arent}$	Aggr $_{own}$	Seed Calculation	K$_{child}$ of the node
0	0	S2=F$_{seed}$ (CONST, S1)	F$_{key}$ (CONST, S2)
0	1		F$_{key}$ (CONST, S1)
1	0	S2=F$_{seed}$ (CONST, S1)	F$_{key}$ (CONST, S2)
1	1		F$_{key}$ (ID$_{child}$, S1)

Fig. 4. Calculating Child Key

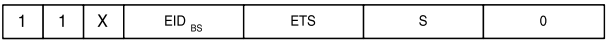
If a node has multiple child-level nodes, number of K_{child} keys is equal to the number of child-level nodes. But the node computes child key for a particular child only after receiving encrypted data from it. As a performance improvement, the node may store child keys of few child-level nodes in its memory.

3.3 Beaconing Phase

Beaconing Steps. This section describes in details various steps in beaconing phase. This phase enables each node to gather neighbor information and to generate network topology. Our algorithm uses this phase for distribution and creation of various session keys; thus achieving key management. The operating sequence in beaconing phase is as follows:

a) Beacon generation steps by the base station:

- The base station encrypts its own ID and current timestamp to EID & ETS respectively using its private key.
- It generates random seed S, and assumes to be at level 0.
- The base station broadcasts following beacon message.



where X is the data aggregation participation flag for all child-level nodes (nodes at level 1) of the base station, determined by certain rule. (Analysis of this rule is out of scope of this research work. For the purpose of this research, this value is calculated as random value between 0 and 1)

b) Beacon generation steps by individual node: Whenever any node receives first beacon message as,

it creates the new beacon message as

- Data aggregation participation flags B and C are copied to appropriate fields in the new beacon message and flag X for child-level nodes is determined by certain aggregation rule.
- Node fills EID field with its own encrypted ID and copies encrypted timestamp from the received beacon's ETS field.
- Node generates new seed value S2 from S1 depending on the data aggregation flags, as described in the previous section.
- Node increments its level by 1 to n + 1.
- Node broadcasts this newly created beacon message.

c) Network formation steps:

- The base station initiates the beaconing phase by creating & broadcasting the new beacon message.
- When a node receives first beacon message, it decrypts ETS using public key (K_{pb}) of the base station. If it is decrypted to the valid timestamp, then the node can assume the beacon message to be authentic. As only the base station has private key to encrypt the valid timestamp, no one else can generate valid ETS.

- The node also decrypts received EID using public key (K_{pb}) of the base station to identify and authenticate source of the beacon message. As only the base station has private key to encrypt the valid ID, no one else can generate valid EID. This ensures that no node can fake its ID; thus achieving node authentication.
- The node marks sender of the first authenticated beacon message as its parent node.
- As described in earlier sections, the node calculates its own key (K_{own}) and parent key (K_{parent}). The node uses K_{own} to encrypt all messages that it sends or forwards and uses K_{parent} to watchdog parent node in data forwarding phase.
- The node creates and broadcasts its own beacon message.
- On receiving subsequent authenticated beacon messages, the node stores data aggregation flags, received seed, level and ID of the sender node. This helps in gathering neighbor information and in calculating K_{child} later.

d) Network reformation steps: Beaconing phase is periodic in nature. The frequency of beaconing phase is deter-mined by dynamic nature of the network. As beaconing phase helps forming network topology, frequency of beaconing phase should be high enough in dynamic networks where individual nodes are mobile and topology keeps changing frequently.
As beaconing phase involves broadcasting beacon messages, it has higher overhead in terms of communication energy consumption. Therefore, choosing right frequency for beaconing phase is important factor in energy-constrained sensor networks. As we make assumption about the sensor nodes not being mobile in nature, we can say that the sensor network as a whole is also static. But new sensor nodes may be added to the network at later stages or sensor nodes may die down (because of energy exhaustion or node malfunction). This causes updating of changed neighbor information in some of the nodes. Periodic beaconing phase facilitates updating of neighbor information in such nodes. Every time the base station initiates beaconing phase with following steps:

- The base station encrypts its own ID and current timestamp to EID and ETS respectively using its private key.
- The base station broadcasts a beacon message with newly generated random seed (S).
- All nodes follow the same steps as in network formation phase to gather new up-dated neighbor information along with new key information.
- Thus, each node gathers information about newly added nodes and deletes information about nonfunctional nodes. This accommodates changing network topology.

Whenever a node receives beacon message from old neighbor (one which was this node's neighbor node in earlier beaconing phase), the node authenticates neighbor ID by comparing neighbor EID with the stored information. This helps in considerable saving in time and energy consumption of the node, as public key encryption is computationally intensive and requires considerable energy. Thus,

each node makes use of public key encryption only once per neighbor node for node authentication. This is one of the advantages of the algorithm as it uses public key encryption sparingly.

4 Analysis of the Algorithm

Earlier sections described various steps in the proposed key management algorithm. This section analyzes the algorithm as a whole to explain its features that make this algorithm feasible to implement and better alternative option, compared to other similar key management algorithms.

4.1 Advantages of Proposed Key Management

a) Combined key management approach We know that single key management strategy is not sufficient and feasible for the needs of resource-constrained sensor network. They require key management technique with high key granularity and high tolerance to key or node compromise. Conventional public key management technique is not communication efficient, use considerable battery energy and are time-consuming. Conventional secret key management technique is simple and energy efficient, but single node compromise may make the entire network vulnerable. The proposed algorithm makes use of both key management techniques to exploit their advantageous properties while minimizing their drawbacks. The proposed algorithm combines several cryptographic techniques as follows:

- Public key cryptography: This is used for beacon message authentication (by checking ETS), node authentication (by checking EID) and communicating node-identity in secure manner for key setup. Its property of being more secure than any other scheme is advantageous in key setup operation. Though, public key technique has higher energy consumption and higher latency, it would only be used periodically in beaconing phase. Any node uses this technique for each of its neighbor nodes only once in its lifetime.
- Identity based cryptography: This scheme uses identity of a node to create secret key, which is then used for pair-wise node communication. In beaconing phase, node ID is communicated securely. Identity-based cryptography makes use of this ID to form secret key. Therefore, a sensor node need not broadcast its key explicitly, thus reducing communication overhead, as lower number of bits needs to be broadcast.
- Secret key cryptography: This scheme is more energy efficient and less computationally intensive. The secret key is computed using information distributed in beaconing phase. This secret key is then used for communication between pair of sensor nodes in data forwarding phase.

b) No Extra Messages This scheme establishes key management using existing messages. It does not require any extra message for key setup. Algorithm carries out key establishment by distributing cryptographic information with beacon message. This causes an increase in the size of beacon message; but the

communication overhead caused by increase in length of original message is lower than the communication overhead for the setup of new message.

c) Adaptivity Key management algorithm also takes into consideration sensor network that employs data aggregation. The algorithm distributes the keys in such a way that node not participating in data aggregation can forward encrypted message without any cryptographic operation, while aggregating nodes can decrypt the data encrypted by its first aggregating descendent. Thus the scheme is tunable to data aggregation, as only aggregating node needs to use cryptographic operation, while non-aggregating node can just forward the encrypted data.

The algorithm also makes sure that a node can use watchdog mechanism even for encrypted messages. Every node can calculate the key that its parent node uses for encrypting outgoing messages. The node can use this key for decrypting messages forwarded by parent node to carry out watchdogging.

In case of failure on part of parent node, child-level node can forward encrypted packet to its alternate parent and do watchdog on alternate parent. Thus the scheme considers the dynamic behavior of the network. Also, this scheme is self-organizing as it makes use of beaconing phase for gathering changing topology information.

5 Simulation Results

For the performance evaluation of proposed key management algorithm for sensor networks, we used network simulator NS-2 (Network Simulator version 2.26).

5.1 Energy Consumption

For any algorithm related to sensor networks, issue of energy consumption is the most important because of energy-constrained nature of sensor nodes. As energy consumption is much larger in communication overhead than in computational overhead, the simulation focuses on the communication cost. The proposed algorithm makes use of beacon message to distribute key-related information. In data forwarding phase, size of data message is same regardless of use of cryptographic technique, as the size of plain data and its corresponding encrypted data is same. Therefore, increase in the size of beacon message is the cause for increased communication overhead in the proposed algorithm. We compared average energy consumption per node for different beacon message types. In Fig. 5, we compared communication overhead for three beacon messages.

- Original beacon message that carries no data
- Beacon message with 16 Bytes of additional data to original message for beacon authentication only
- Beacon message with 24 Bytes of additional data to original message for pro-posed algorithm (beacon authentication and data encryption)

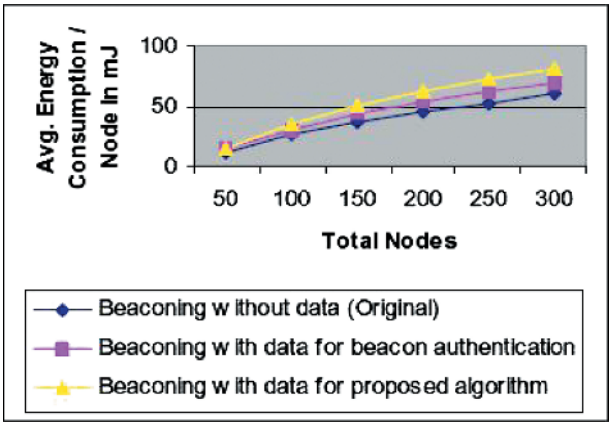


Fig. 5. Energy Consumption Comparison

5.2 Robustness to Node Compromise

We consider that sensor nodes are deployed randomly and may not be attended later. This makes sensor nodes prone to get attacked and compromised. Adversary can get all information from the compromised node, including sensitive cryptographic data. Apart from public key (K_{pb}) of the base station, encrypted ID (EID) of the node, constant number (CONST) and one-way function, adversary can also gather neighbor information stored in the node; thus making neighboring nodes vulnerable. Therefore, compromising one node affects its surrounding nodes also as adversary can decrypt messages encrypted by these surrounding nodes. But as possibly different key is used by neighbor's neighboring node to communicate; effect of compromised node is limited and localized. The

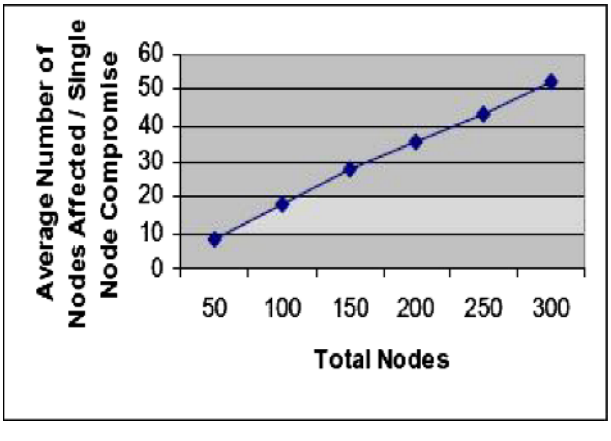


Fig. 6. Effect of Single Node Compromise

following graph shows average number of nodes affected be-cause of single node compromise when proposed key management scheme is used.

As only some part of the network is affected, proposed scheme makes the network more robust. The other implemented key management scheme, TinySec [4], which relies completely on symmetric cryptography, has single key for the entire network. Therefore, compromising one node makes the entire network vulnerable as adversary can decrypt any message between any two nodes. Following graph is another version of the above graph, which compares the percentage network affected by single-node compromise when proposed algorithm (average 17%) and TinySec (100%) are used.

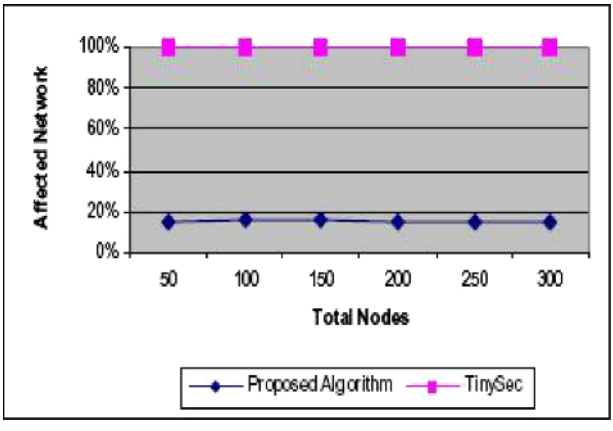


Fig. 7. Comparison between Proposed Scheme and TinySec for Affected Network by Single Node Compromise

6 Conclusion

As with the other networks, it has been acknowledged that cryptographic techniques need to be considered to address issues related to data authentication and confidentiality in sensor networks. But both forms of cryptographic techniques - symmetric and asymmetric - are not adequate when considered alone. Our work combines both cryptographic techniques to get benefits of both schemes while minimizing their draw-backs. The proposed key management scheme uses public key cryptography judiciously for authentication purpose, thus reducing overhead associated with it. The scheme uses efficient secret key cryptography for data confidentiality by setting up (possibly different) session keys between each pair of nodes. Having different session keys reduces the impact of single-node compromise on the network.

This combined key management scheme is generic in nature so that it can be ap-plied with different routing algorithms for sensor networks. This scheme also accommodates sensor networks that utilize other techniques like energy-efficient data aggregation and secure watchdog mechanism.

As a future work, we can make this key management scheme more dynamic so as to vary security level as per the need. After initial beaconing, the base station can decide whether nodes should participate in only beacon authentication or in only data encryption or in both. It can also decide to switch off entire encryption mechanism temporarily. This helps in conserving energy when there is low security requirement.

References

1. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci: A survey on Sensor Networks, IEEE Communication Magazine, August 2002, pp. 102-114.
2. TinyOS: http://webs.cs.berkeley.edu/tos/index.html
3. Crossbow Technology, Inc.: http://www.xbow.com
4. C. Karlof, N. Sastry, U. Shankar, D. Wagner: TinySec: TinyOS Link Layer Security Proposal, July 2002.

SDD: Secure Directed Diffusion Protocol for Sensor Networks*

Xiaoyun Wang[1,2], Lizhen Yang[1], and Kefei Chen[1]

[1] Department of Computer Science and Engineering,
Shanghai Jiaotong University,
1954 Huashan Road, Shanghai, People's Republic of China
[2] Department of Information and Communication,
Tongji University, Shanghai,
People's Republic of China
{ellen_wang, yang-lz, chen-kf}@cs.sjtu.edu.cn

Abstract. A sensor network is a collection of tiny sensor nodes, which consists of sensing, data processing and communicating components. Directed Diffusion is an important data-centric routing protocol of sensor networks. In this paper we present the design of a new secure Directed Diffusion protocol (SDD), which provides a secure extension for the Directed Diffusion protocol. We mainly focus on secure routing and give a simple scheme to securely diffuse data. We have not considered with the in-network aggregation as a goal. In order to support the use of SDD for sensor nodes with extremely limited CPU processing capability, we use an efficient one-way chain and do not use asymmetric cryptographic operations in this protocol. Our security analyses show that SDD is robust against any active attackers or compromised nodes in the network.

1 Introduction

Recent advances in wireless communications and electronics have enabled the development of low-cost, low-power, multifunctional sensor nodes that are small in size and communicate in short distances. These tiny sensor nodes, which consist of sensing, data processing and communicating components, leverage the idea of sensor networks.

Routing in sensor networks is very pivotal and could be classified as data-centric, hierarchical or location-based although there are few distinct ones based on network flow or quality of service (QoS) awareness. Many new algorithms have been proposed for the problem of routing data in sensor networks. But these protocols have not been designed with security as a goal.

In this paper, we focus on securing data-centric routing of sensor networks and in particular Directed Diffusion protocol. Directed Diffusion protocol [1, 2] is an important milestone in the data-centric routing research of sensor networks.

* This work is supported by China NSFC under grant 90104005 and 60273049.

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 205–214, 2005.

A number of proposed data-centric sensor networks routing protocols are based on adapting the basic Directed Diffusion protocol, including Rumor routing [3], Gradient-based routing [4] and CADR [5].

We present the design and security analyses of a new Secure Directed Diffusion protocol (SDD) which provides a secure extension over the Directed Diffusion protocol. In this protocol we focus mainly on routing security and give a simple scheme to diffuse data securely. In our scheme in-network aggregation is forbidden, though which can make the data propagation more efficient. Our security analyses show that our scheme is robust against multiple attackers in spite of any active attackers or compromised nodes in the network. And in order to support use of SDD for sensor nodes with extremely limited CPU processing capability, we use efficient one-way chain and do not use asymmetric cryptographic operations in this protocol.

In section2 of this paper, we summarize the basic operation of Directed Diffusion. Section3 presents our assumptions about the network and nodes involved in the sensor network. In section 4, we propose the design of SDD, first we give an overview of basic design, and then we divide SDD into four phases and describe them in detail. Section 5 presents the analyses of security. In section 6 we discuss related work and finally, in section 7, we present conclusions and the future work.

2 Directed Diffusion Protocol

Directed Diffusion protocol is an important data-centric routing protocol that suggests the use of attribute-value pairs for the data and queries the sensors in an on demand basis by using those pairs. This protocol could be divided into four phases.

The first phase is interest propagation phase. Interest is a task descriptions defined by a list of attribute-value pairs such as name of objects, interval, duration, geographical area, etc. For each active task, the sink periodically broadcasts an interest message to each of its neighbors. When a node receives an interest, it checks to see if the interest exists in the cache. If no matching entry exists, the node creates an interest entry and save the parameter instantiated from the received interest. One parameter of interest entry is gradient that specifies both a data rate and a direction in which to send events.

Then the next phase is low-rate data propagation and routing setup phase. A sensor node that detects a target searches its interest cache for a matching interest entry. When it finds one, the node sends low-rate data to each for whom it has a gradient. A node that receives a data message from its neighbors attempts to find a matching interest entry in its cache. If a match exists, the node adds the received message to the data cache and resends it to the node's neighbors.

The third phase is reinforcement phase. Now the low-rate data reaches the sink along multiple paths. The sink then selects and reinforces one particular path in order to draw down higher quality events. To reinforce the path, the sink resends the original interest message but with a smaller interval. In this

situation the reinforcement is triggered by a sink. In order to enable local repair of failed or degraded paths, the intermediate nodes could trigger reinforcement also.

The last phase is data propagation phase. The source node computes the highest requested event rate among all its outgoing gradients and sends them to its neighbors. The message receiver examines the matching interest entry's gradient list. If there is a lower data rate than the received data rate, it may downconvert the data to the appropriate gradient. And it also does some in-network data aggregation before resending the message in order to make the data aggregation more efficient.

3 Assumptions

First we assume that network physical and MAC (Media Access Control) layer attacks such as physically jamming and physically DOS attack are beyond the scope of this paper.

Then we assume that our sensor network has one base station (sink) and thousands of sensor nodes. There's a pre-deployed shared key between sink and each sensor node.

We also assume that the nodes in the sensor network may be strictly resource constrained. Thus in Secure Directed Diffusion routing protocol (SDD), we use efficient one-way hash chains rather than relying on expensive asymmetric cryptographic operations.

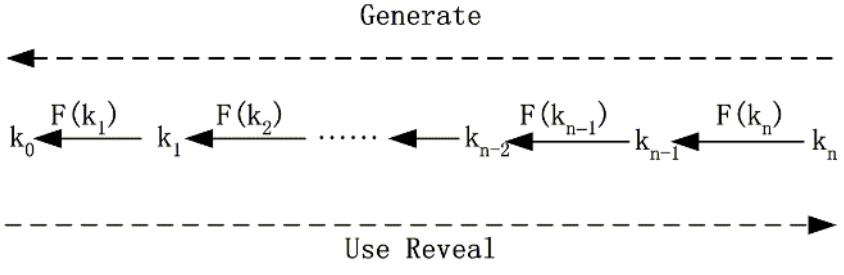


Fig. 1. One-way chain example. Redraw from [6]

Figure1 shows the one-way chain construction. To generate a chain of length n we randomly pick the last element of the chain k_n. We generate the chain by repeatedly applying a one-way function F. We can verify a latter element of the chain through a previous one, e.g. to verify that $k_j$is part of the chain if we know that k_i is the ith element of the chain ($i < j$), we check that $F^{j-i}(k_j) = k_i$.

Finally we assume that there's a pre-deployed one-way chain in the sink and the first element k_0 of this one-way chain is pre-deployed to all sensor nodes. After the elements of the pre-deployed one-way chain use up, we generate another one-way chain and announce k_0 of the new one-way chain by the protection of the last element of the previous one-way chain.

4 Secure Directed Diffusion

4.1 Basic Design of SDD

The Secure Directed Diffusion is also divided into four phases. In each phase cryptographic mechanisms are adopted to realize security.

In the first phase of interest propagation since the interest should be cached by all the sensor nodes but should not be modified during diffusion, we use a modified immediate authentication TESLA protocol [7] to ensure the interest is from the sink, could be read by all the sensor nodes, and has not been modified by intermediate nodes.

The next phase is the low-rate data propagation and secure routing setup phase. In this phase low-rate data is collected along with setting up multiple routes. Though in this phase data are always low-rate, intermediate nodes still need to read data to decide which neighbor to forward. So we use a modified immediate authentication TESLA protocol to ensure that data are from the source node, could be read by intermediate nodes and have not been modified. Each intermediate node also adds its node identity to a node list in forwarded packet and adds an "signature" to the packet by encrypting a nonce using its shared key.

In the third phase of path selection and reinforcement propagation, the sink selects a path probabilistically and reinforces the path. A reinforcement is protected just in the same way as an interest is protected.

In the last phase of data propagation, though we suppose that the in-network data aggregation and data downconverting are forbidden in our scheme, intermediate nodes still need to read data to know its data rate and decide which neighbor to forward. So we protect data the same way as data is protected in the low-rate data propagation and secure routing setup phase.

The notation we will use is listed as following:

$F()$A hash function used to calculate one-way chain.
$H()$A hash function used to calculate hash value
$k_0, k_1, \ldots\ldots,\quad k_n$The sink's one-way chain used as keys
$MAC(k_i, Message)$MAC(Message Authentication Code) of Message, using key k_i.
$[message]_x$Represents that the message is encrypted by the key x
N_nNode identity of sensor node N
S_nA shared key between sensor node N and the sink
$|$Stands for message concatenation.
$k_F^0, k_F^1, \ldots\ldots, k_F^n$Node F's one-way chain used as keys.

4.2 Interest Propagation Phase

Since the first element k_0 of sink's one-way chain is pre-deployed to all sensor nodes, the sink could use a modified immediate authentication TESLA protocol to broadcast message to all sensors.

When the sink wants to send an interest, it floods a packet in the form of $\{H(\text{INTEREST}_1)|MAC(k_1, H(\text{INTEREST}_1))\}$.

Suppose after time d, all the sensor nodes have received the above packet. Then the sink floods another packet in the form of $\{\text{INTEREST}_1|k_1\}$.

The second packet can be verified at once when a sensor node receives it. So it resists cache overflow attack in TESLA.

First we verify if it is from sink by check if $F(k_1) = k_0$. Then since the node knows k_1, the node can verify $H(\text{INTEREST}_1)$ by $\text{MAC}(\text{k}_1,\text{H}(\text{INTEREST}_1))$ to see if it has been modified during the diffusing. Finally the node can authenticate the INTEREST_1 by $H(\text{INTEREST}_1)$.

Thus we diffuse an authenticated INTEREST_1 to the whole sensor network. Then we research how to secure the routing.

4.3 Low-Rate Data Propagation and Secure Routing Setup Phase

Suppose the INTEREST_1 has been flooded to all the sensor nodes and the gradients have been set up as the figure2 shows.

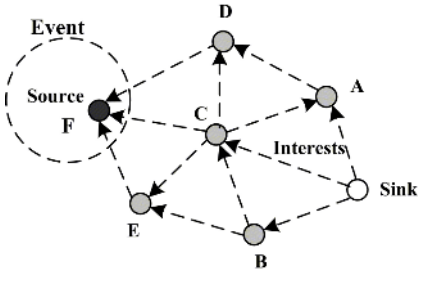


Fig. 2. (a)Interest propagation (b) Initial gradients set up

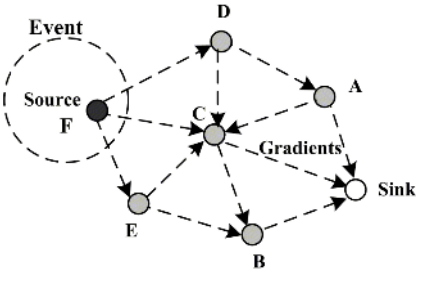


Fig. 3. A simplified scheme for directed diffusion (Redraw from [1])

In routing setup phase all the gradients are low-rate. The low-rate data propagation and secure routing setup are shown as following.

First the source node F floods the packet to other sensor nodes and the sink in the form of $\{H(\text{DATA}_1)|MAC(k_F^1, H(\text{DATA}_1))|[k_F^0]_{S_F}|F\}$ where DATA_1 is the data which F wants to send.

The sink receives the packet and decrypt $[k_F^0]_{S_F}$ by the shared key S_F and gets the first key of F's one-way chain k_F^0. Then it sends the k_F^0 to all sensor nodes by flooding the packet $\{H(k_F^0|F)|MAC(k_2, H(k_F^0|F))\}$ and after disclosure time $d$flooding the packet $\{k_F^0|F|k_2\}$. Then all the sensor nodes know k_F^0 the first key of F's one-way chain.

Then Node F begins to send data. It sends $\{\text{DATA}_1|k_F^1|nonce_1|\{F\}|[nonce_1]_{S_F}\}$ to Node D, C and E. where {F} is the node list of the path. $nonce_1$ is used to prove its freshness and to protect against replay attacks.

Node D receives the packet from Node F, and it sends $\{\text{DATA}_1|k_F^1|nonce_1|\{F, D\}|[[nonce_1]_{S_F}]_{S_D}\}$ to Node A and C.

Node C receives the packet from Node F, and it sends $\{\text{DATA}_1|k_F^1|nonce_1|\{F,C\}|[[nonce_1]_{S_F}]_{S_C}\}$ to Node B and sink.

Node E receives the packet from Node F, and it sends $\{\text{DATA}_1|k_F^1|nonce_1|\{F,E\}|[[nonce_1]_{S_F}]_{S_E}\}$ to Node B and C.

Node A receives the packet from Node D, and then it sends $\{\text{DATA}_1|k_F^1|nonce_1|\{F,D,A\}|[[[nonce_1]_{S_F}]_{S_D}]_{S_A}\}$ to Node C and sink.

We suppose that Node B firstly receives the packet from Node E, and then it sends$\{\text{DATA}_1|k_F^1|nonce_1|\{F,E,B\}|[[[nonce_1]_{S_F}]_{S_E}]_{S_B}\}$ to sink.

The sink receives three packets $\{\text{DATA}_1|\text{k}_F^1|nonce_1|\{F,C\}|[[nonce_1]_{S_F}]_{S_C}\}$, $\{\text{DATA}_1|k_F^1|nonce_1|\{F,D,A\}|[[[nonce_1]_{S_F}]_{S_D}]_{S_A}\}$ and $\{\text{DATA}_1|k_F^1|nonce_1|\{F,E,B\}|[[[nonce_1]_{S_F}]_{S_E}]_{S_B}\}$.

The sink checks if the DATA_1 are from F and have not been modified and checks if $nonce_1$ could be decrypted by using the intermediate nodes' shared key in turn. If either check fails the sink silently discards the packet. Otherwise the sink accepts the packet and the path. If all of the three packets in the above pass the verification, the sink now has three paths {F, C}, {F, D, A} and {F, E, B}

4.4 Path Selection and Reinforcement Propagation Phase

Multiple paths appear more robust against the malicious attacks. And how to select a path to reinforce is still a good method against the malicious attacks such as wormhole attack.

Since wormhole attacker or a laptop-class attacker can supply a higher quality route than normal nodes, the sink should not choose a path always base on lower delay or shorter hop. The sink should choose a path probabilistically. And the detailed probability distribution depends on the different designer.

We suppose that the sink chooses path {F, C} to reinforce. In order to propagate a reinforcement the sink firstly sends a packet in the form of $\{H(\text{REINFORCEMENT}_1|\{F,C\})|MAC(k_3,H(\text{REINFORCEMENT}_1|\{F,C\}))\}$.

After disclosure time d the sink sends a packet in the form of $\{\text{REINFORCEMENT}_1|$ {F,C}$|k_3\}$.

The node, which receives the two packets, checks if the two packets are from the sink and if REINFORCEMENT_1 has been modified in the diffusion. If either check fails, the node discards the two packets. Otherwise the node checks if it is an intermediate node on the path. If the node is on the path that is being reinforced, it records the reinforcement information in its cache. Otherwise it does nothing.

Here we just consider the case that the reinforcement is triggered by a sink. The case that reinforcement is triggered by the intermediate node, in order to enable local repair of failed or degraded paths, is forbidden for the sake of security.

4.5 Data Propagation Phase

Here we suppose that the in-network data aggregation and data downconverting are forbidden and the intermediate node should not modify the data. The source

node sends different rate data to its neighbor. And the neighbor forwards the data to its appropriate neighbor according to the data rate.

F floods $\{H(\text{DATA}_2)|MAC(k_F^2, H(\text{DATA}_2))\}$ to all the sensor nodes. After disclosure time d it sends $\{\text{DATA}_2|k_F^2\}$ to each neighbor for whom it has an appropriate gradient.

A node that receives the above packet from its neighbor firstly checks if the DATA_2 are from node F and have not been modified in the diffusion. Then the node attempts to find a matching interest entry and corresponding appropriate gradients in its cache. At last the node resends the packet to the appropriate neighbors.

5 Security Analyses

Scenario1: consider the case that an attacker modifies or forges and replays interest packets.

Case1:If it forges the interest packets in the form of $\{H(\text{INTEREST}')|$ $\text{MAC}(\text{k}_1',\text{H}(\text{INTEREST}'))\}$ and $\{\text{INTEREST}'|k_1'\}$. The sensor nodes can check that $F(k_1') \neq k_0$ and know they are not from the sink. Then the sensor nodes discard the interest packets.

Case2:If the packet is modified in the form of $\{H(\text{INTEREST}')|$ $\text{MAC}(\text{k}_1,\text{H}(\text{INTEREST}))\}$ and $\{\text{INTEREST}'\|mboxk_1\}$. Then the sensor nodes check that $H(\text{INTEREST}') \neq H(\text{INTEREST})$ and then discard the interest packets also.

Case3: If the packet replays two previous packets $\{H(\text{INTEREST}_1)|$ $\text{MAC}(\text{k}_n,\text{H}(\text{INTEREST}_1))\}$ and $\{\text{INTEREST}_1|k_n\}$ the sensor nodes can check that the $\text{k}_n$is not a fresh one and then discard the interest packets.

Scenario2: consider the case that in routing setup phase, a wormhole attacker can provide a less hop path and a lap-top class attacker can provide a lower delay path than normal nodes. They tempt the sink to choose and reinforce a path, which they are on, and gain a full control over the data flow. They can modify and selectively forward packets of their choosing. Thanks for the probabilistic path selection method, which the sink adopts, thus the attackers are not so easy to realize the above attack.

Scenario3: consider the case that an attacker may modify the path it is on and let the sink accept the path. For example we assume Node B in figure 1 is a malicious node.

Case 1:Node B deletes node E in the node list and want to provide a less hop path. But it couldn't remove the encryption by Node E. So it sends to the sink the packet $\{\text{DATA}_1|k_F^1|nonce_1|\{F,B\}|[[[nonce_1]_{S_F}]_{S_E}]_{S_B}\}$. The sink will decrypt the $[[[nonce_1]_{S_F}]_{S_E}]_{S_B}$ using key $S_B$and $S_F$in turn. But the sink fails in decrypting. So the sink discards the packet and rejects the path.

Case2: Node B modifies Node E as Node C or adds Node C into the node list. Since it does not know the shared key between node C and the sink. It cannot forge a true encryption using Node C's shared key. So the sink will discard the packet and reject the path also.

Case3: For the broadcast property of the wireless channel, node B may hear other packet, which is sent to other nodes, such as $\{\text{DATA}_1|k_F^1|nonce_1|\{F,D\}|[[nonce_1]_{S_F}]_{S_D}\}$. Node B forges a packet $\{\text{DATA}_1|k_F^1|nonce_1|\{F,D,B\}|[[[nonce_1]_{S_F}]_{S_D}]_{S_B}\}$ and sends to the sink. The sink verifies that it is ok and accepts it. But we notice that in [1](section4.4), it is said that in diffusion, data traffic is transmitted using MAC (Medium Access Control) unicast. So this problem could be avoided.

Scenario4: consider the case that an attacker forges, modifies or replays reinforcement packets in order to let the path, which the attacker is on, be reinforced. And then the attacker could gain full control of the data flow. Since reinforcement packets are authenticated by keys from one-way chain, each sensor node could check if reinforcement packets are from the sink, if they are modified by other nodes and if they are fresh ones. The detailed authentication could reference scenario1.

Scenario5: consider the case that an attacker forges, modifies and replays data packets. Since data packets are authenticated by keys from one-way chain. Every sensor node could check if data packets are from the claimed source node, if they are modified by other nodes and if they are fresh ones. The detailed authentication could reference scenario1.

6 Related Works

Secure routing in sensor networks is a really new area. There is only one analyses paper [8] we could find in this area. In [8] karlof and wagner propose security goals for routing in sensor networks, gives six classes attacks against sensor networks and analyze the security of all the major sensor network routing protocol, at last suggest countermeasures and design considerations. But they do not propose any integrated scheme for a particular routing protocol.

Since Directed Diffusion is a query-driven on demand protocol we do reference many secure on demand protocols in ad hoc networks.

Sanzgiri and Dahil [9] propose ARAN secure routing protocol for an on-demand routing protocol. ARAN requires the use of a trusted certificate server (T) and requires each node to request a certificate signed by T before entering the ad hoc network. In ARAN, every node that forwards a route discovery or a route reply message must also sign it. So it consumes computing power and causes the size of the routing messages to increase at each hop.

Zapata [10] gives an extension of the AODV[11] routing protocol (SAODV). The basic idea is to use a RSA signature and the hash value to protect the route discovery mechanism providing security features. The use of public-key cryptography imposes a high processing overhead on the intermediate nodes.

As above when the nodes in an ad hoc network are generally unable to verify asymmetric signatures quickly enough, or network bandwidth is insufficient, these protocols may not be suitable.

Papadimitratos and haas[12] propose a protocol(SRP) that can be applied to several existing routing protocols. The sole requirement of the proposed scheme

is the existence of a security association between the node initiating the query and the sought destination.

Hu, Perrig and Johnson [13] present the design of a new secure on-demand ad hoc network routing protocol, called Ariadne. Afiadne prevents attackers from tampering with uncompromised routes consisting of uncompromised nodes, and also prevents a large number of types of Denial-of-Service attacks. In addition, Ariadne is efficient, using only highly efficient symmetric cryptographic primitives. However it requires clock synchronization, which is a difficult requirement for ad hoc networks.

7 Conclusion and Future Work

In this paper, we present the design and give security analyses of SDD, a Secure Directed Diffusion protocol. We divide SDD into four phases, the interest propagation phase, secure routing setup phase, path selection and reinforcement propagation phase and data propagation phase. We also apply one-way chain to protect the security of each phase. We consider mainly about the routing security in Directed Diffusion protocol and give a simple scheme to diffuse data securely. We forbid the in-network data aggregation in our scheme, though, which can make the data propagation more efficient. Our security analyses show that our scheme is robust against multiple attackers in spite of any active attackers or compromised nodes in the network.

In future work, we plan to do some further research on SDD such as realizing secure in-network data aggregation in SDD. And we plan to do some simulation based on NS2. We will research the network performance of SDD and give an evaluation with the normal Directed Diffusion.

References

1. C. Intanagonwiwat, R.Govindan, D.Estrin, “Directed diffusion: a scalable and robust communication paradigm for sensor networks”, Proceedings of the 6th Annual ACM/IEEE International Congerence on Mobile computing and Networking(MobiCom’00), Boston, MA, August 2000.
2. D.Estrin,et al., “Next century challenges:scalable coordination in sensor networks” Proceedings of the 5th Annual ACM/IEEE International Congerence on Mobile computing and Networking(MobiCom’99), Seattle, WA, August 1999.
3. D.Braginsky, D.Estrin, “Rumor routing algorithim for sensor networks”, Proceedings of the First Workshop on Sensor Networks and Applications(WSNA), Atlanta, GA, Octoboer 2002.
4. C. Schurgers, M.B. Srivastava, “Energy efficient routing in wireless sensor networks”,The MILCOM Proceednge on Communications for Network-Centric Operations: Creating the Information Force, McLean, VA, 2001.
5. M.Chu, H.Haussecker, F.Zhao, “Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks ”, the International Journal of High Performance Computing applications 16(3)(2002) 293-313.

6. Adrian Perrig, Ran Canetti, J.D. Tygar, and Dawn Song, "The Tesla Broadcast Authentication Protocol", RSA CryptoBytes, vol.5, 2002
7. A. Perrig, R. Canetti, D. Song, and J. D. Tygar. "Efficient and secure source authentication for multicast". In Proceedings of the IEEE Symposium on Network and Distributed System Security(NDSS 2001), pages 35-46, 2001
8. Chris Karlof, David Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures", Ad hoc Networks 1(2003) 293-315
9. Bridget Dahill, Brian Neil Levine, Elizabeth Royer, and Clay Shields. "A Secure Routing Protocol for Ad Hoc Networks". Technical Report UM-CS-2001-037, Electrical Engineering and Computer Science, University of Michigan, August 2001.
10. Manel Guerrero Zapata. "Secure Ad hoc On-Demand Distance Vector (SAODV) Routing". Internet draft draft-guerrero-manet-saodv-00.txt. October 2001
11. C.E.Perkins, E.M.Royer, S.R.Das "Ad hoc On-Demand Distance vector Routing" draft-ietf-manet-aodv-08.txt. IETF MANET working Group, June 1^{st}, 2001
12. P. Papadimitratos and Z. J. Haas. "Secure routing for mobile ad hoc networks". SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), Jan 2002
13. Yih-Chun Hu, Adrian Perrig, David B. Johnson Ariadne: "A secure on-demand routing protocol for ad hoc networks" , Proceedings of the 8th annual international conference on Mobile computing and networking (September 2002)

Secure AES Hardware Module for Resource Constrained Devices

Elena Trichina[1] and Tymur Korkishko[2]

[1] Department of Computer Science, University of Kuopio,
P.O.B. 1627, FIN-70211, Kuopio, Finland
etrichin@cs.uku.fi
[2] Information security TG, i-Networking Lab, Information Security Group,
Samsung Advanced Institute of Technology, Korea
k.tymur@samsung.com

Abstract. Low power consumption, low gate count, and high throughput are standard design criteria for cryptographic coprocessors designated for resource constrained devices such as smart cards. With the advent of side channel attacks, devices' resistance to such attacks became another major requirement. This paper describes a cryptographic hardware module for an AES algorithm that provides complete protection against first order differential power analysis by embedding a data masking countermeasure at a hardware level. We concentrate on inversion in $GF(2^8)$ since this is the only non-linear operation that requires complex transformations on masked data and on bits of the masks. The simulation and synthesis results confirm that the proposed solution is suitable for applications in GSM and ad-hoc networks in terms of performance, gate count and power consumption. To our knowledge, this is the first implementation of a side channel-resistant AES hardware module suitable for smart- and SIM-cards.

1 Introduction

In applications such as smart cards and related embedded devices, hardware complexity and tamper resistance are very important issues that directly affect the cost and consumer acceptance of such devices. When invasive attacks on smart cards had been reported [2], an industry reacted by incorporating in a chip features such as glue logic, metal shields, and sensors of abnormal behavior [15]. But it was not long before a new class of *side channel* attacks emerged as a powerful thread. These attacks enable breaking cryptographic algorithms by measuring timing characteristics [13], power consumption [12, 20] or electromagnetic radiation [9, 25] of a smart card microprocessor when it runs cryptographic applications.

Until recently, most of these attacks exploited some specific features of software implementations of cryptographic algorithms. Fittingly, most countermeasures were also designed at software level. For many applications, however, it is necessary that cryptographic algorithms should be realized in hardware. Hence,

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 215–229, 2005.

research into vulnerability of cryptographic hardware is just as important. Although not many results had been published yet, it is prudent to suggest that cryptographic hardware also leaks side channel information, and that alongside with general tamper-resistant features, cryptographic coprocessors should include countermeasures specifically targeted to protect them against side channel attacks.

One of the most powerful techniques to counteract side channel attacks is to *mask* all input and intermediate data with some random values in order to de-correlate any information leaked through the side channels from actual secret data being processed [5]. The idea is simple: the message and the key are masked with some random masks at the beginning of computations, and thereafter everything is almost as usual. Of course, the value of the mask at the end of some fixed step must be known in order to re-establish the expected value at the end of the execution; we call this *mask correction*.

In this paper we propose an architecture for an AES [7] coprocessor that is immune to simple and first order differential power analysis attacks. The main idea is to implement directly in hardware a novel method of computing all field operations directly on masked data. A price to pay for such built-in security is increased gate counts and power consumption. The synthesis results for an architecture with data masking countermeasure implementing a 16 clock/round version of the AES with a flexible key size, show that with 0.18 μm technology the performance of 4Mbps can be achieved with the circuit comprising 20,506 gates clocked at 5MHz and requiring 1.7 μA. This is better than countermeasures such as dynamic and differential CMOS logic [29] or asynchronous circuits with dual rail logic [23] can offer. Also, our solution has an advantage of using standard technologies, standard gate libraries and well-established design tools.

The general design flow for a secure AES module is depicted in Fig. 1. First, a Verilog model had been created and tested, after which Cadence Design System Verilog-XL simulator was used to generate timing diagrams and to verify the correctness of the design. When RTL code had been verified, the digital circuit was synthesized with Synopsys Design Analyzer tool using Samsung 0.18 μ libraries. Power compiler simulation data at 5MHz were obtained with the in-house simulation tool CubicWare.

The rest of the paper is organized as follows. After a brief description of the AES algorithm and basic hardware architecture, we outline in Chapter 3 how to reduce inversion in the field $GF(2^8)$ to inversion in the composite field $GF((2^4)^2)$, and how the latter can be realized in combinational logic only. Chapter 4 discusses a general countermeasure against side channel attacks comprising masking all manipulated data with random values, and points out the difficulties of implementing inversion on masked data. We suggest our solution to this problem by showing how computations on masked data and corresponding mask corrections can be carried out at a gate level. The resulting secure AES hardware architecture is described in details in Chapter 5, where we also estimate the cost in terms of the gate count and power consumption. The paper is concluded with simulation results and an outline of a future work.

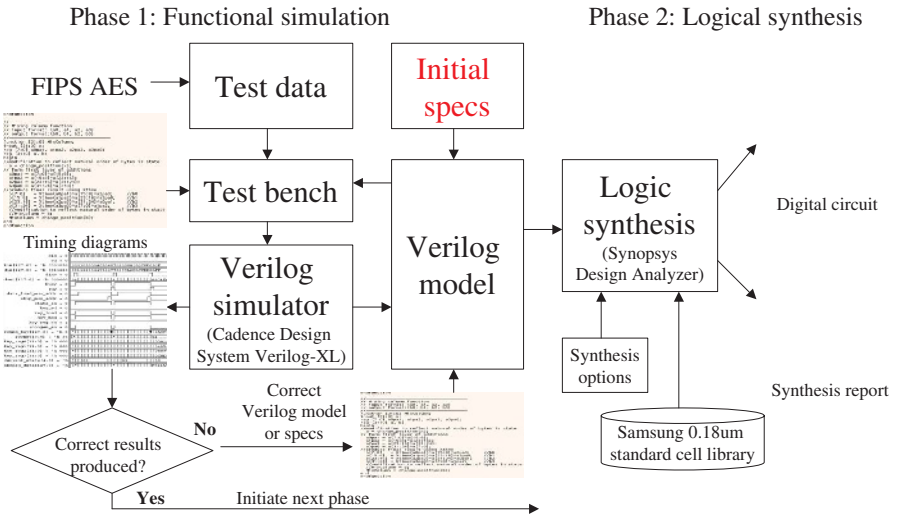


Fig. 1. AES hardware design flow

2 AES Reminder

The Advanced Encryption Standard [7] is a round-based symmetric block cipher. The round consists of four different operations namely, *SubBytes*, *ShiftRows*, *MixColumn*, and *AddRoundKey* that are performed repeatedly in a certain sequence; each operation maps a 128-bit input state into a 128-bit output state. The state is represented as 4×4 matrix of bytes. The number of rounds, depending on the key size, is 10, 12 or 14, with the MixColumn operation being omitted in the last round. Prior to the main loop, AddRoundKey is executed for initialization. The standard key size is 128 bits; but for some applications 192 and 256-bit keys must be supported as well. In the decryption process, the inverse operations of each basic function are executed in a slightly different order.

ShiftRows is a cyclic shift operation on each of four row in a 4×4-byte state using $0 \sim 3$ offsets. *MixColumn* treats 4-byte data blocks in each column of a state as coefficients of a 4-term polynomial, and multiplies them modulo $x^4 + 1$ with the fixed polynomial $c(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$. *AddRoundKey* is a bit-wise XOR operation on 128-bit round keys and the data. These three operations are *linear*.

SubBytes is the main building block of the AES algorithm. It replaces each byte in a state by its substitute in an *Sbox* that comprises a composition of two transformations:

– First, each byte in a state is replaced with its reciprocal in the finite field $GF(2^8)$, except that 0, which has no reciprocal, is replaced by itself. This is the only *non-linear* function in the AES algorithm.

– Then an affine transformation f is applied. It consists of a bitwise matrix multiply with a fixed 8×8 binary matrix followed by XOR with the hexadecimal number {63}.

The *round key* is computed in parallel to the round operation. It is derived from the cipher key by means of key expansion and round key selection operations, which are similar to those of the round operation and use *Sbox*-es.

Fig. 2 shows an implementation of a fully "unfolded" 128 key bits AES algorithm that executes one round of encryption/decryption per cycle.

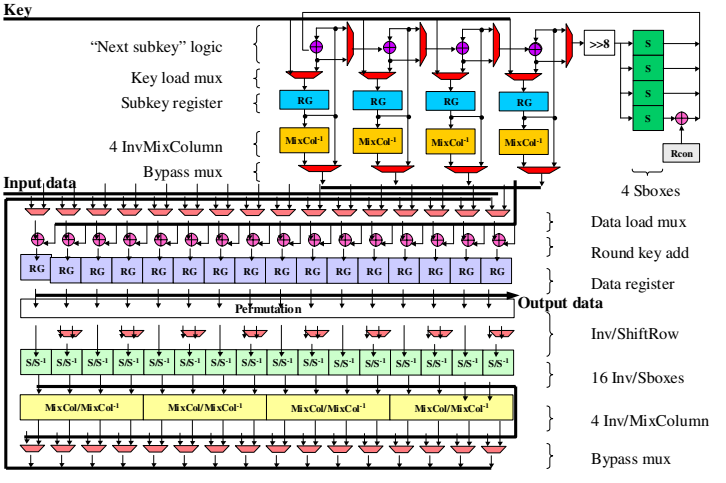


Fig. 2. Structure of the datapath and the key scheduler for a fully unfolded AES architecture

A sequence of round operations is implemented as a combinational circuit and its input and output are connected to 128-bit data registers. The multiplexors are used to skip some operations in the first and the last rounds and to choose between encryption or decryption modes. While *AddRoundKey* and *ShiftRows* have rather trivial hardware realization, there are many possible choices how to optimize silicon area for the other two operations. Important design parameters are suitability for both, encryption and decryption, and the number of *Sbox* and *MixColumn* blocks used in the design. An efficient implementation of these two blocks plays a major role in reducing area [16, 18] and power consumption [22].

3 Sbox Architecture

There are many design trade-offs to be considered when implementing an *Sbox* in ASIC since the size, the speed, and the power consumption of an AES coprocessor depend largely on the number and the style of implementation of

Sbox-es [22]. It also turned out that this operation is the most difficult to protect against side channel attacks.

A number of flexible ASIC solutions that use similarities between encryption and decryption to share silicon were proposed [16, 22], where the *SubBytes* operation was implemented in two steps, as a combination of inversion in the field and an affine transformation. While the affine transformations used for encryption and decryption are slightly different, the silicon implementing inversion in $GF(2^8)$ can be used for both, as shown in Fig. 3. Therefore, an area- and power-efficient and secure implementation of inversion may have a big impact on an overall design.

The most obvious solution is to use a look-up table for this operation [16]. It is fast and inexpensive in terms of power consumption [22]. There is a major drawback, however. Namely, the size of silicon, which is about 1,700 gate equivalents per table in 0.18μm technology. Considering that up to 20 such tables (including 4 tables for key scheduling) may be required, this solution is hardly feasible for co-processors intended for smart cards and other embedded systems.

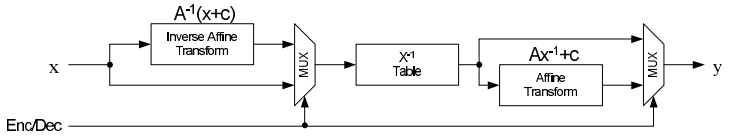


Fig. 3. S-box implementation suitable for encryption and decryption

Among various alternative approaches, the one based on inversion in the composite field produces the most compact AES implementations [26, 27, 32] that can also be optimised with respect to power consumption [22]. As a basis for our design, we use fully combinational logic implementation of inversion in composite fields described in [32].

3.1 Composite Fields

Usually, the field $GF(2^8)$ is seen as an extension of $GF(2)$ and therefore its elements can be represented as bytes. However, $GF(2^8)$ can also be seen as a quadratic extension of $GF(2^4)$; in this case an element $a \in GF(2^8)$ is represented as a linear polynomial $a_H x + a_L$ with coefficients in $GF(2^4)$. We denote it $[a_H, a_L]$. This isomorphic representation had been found to be far better suited for hardware implementation [24, 26, 22, 32].

The bijection from $a = (a_0, ..., a_7)$ to a two-term polynomial $[a_H = (a_{h0}, ..., a_{h3}), a_L = (a_{l0}, ..., a_{l3})]$ is given by a linear function *map* computed by means of the XOR operation on bits of a (see [32]):

$a_L = (a_{l0} = a_C \oplus a_0 \oplus a_5,\ a_{l1} = a_1 \oplus a_2,\ a_{l2} = a_A,\ a_{l3} = a_2 \oplus a_4)$, and
$a_H = (a_{h0} = a_C \oplus a_5,\ a_{h1} = a_A \oplus a_C,\ a_{h2} = a_B \oplus a_2 \oplus a_3,\ a_{h3} = a_B)$,

where $a_A = a_1 \oplus a_7, a_B = a_5 \oplus a_7, a_C = a_4 \oplus a_6$. The inverse transformation map^{-1} converts a two-term polynomial back to element $a \in GF(2^8)$, and is defined in a similar way. For more details see [32].

All arithmetic operations applied to elements in $GF(2^8)$ can also be computed in the new representation. Two-term polynomials are added by addition of corresponding coefficients: $(a_H x + a_L) \oplus (b_H x + b_L) = (a_H \oplus b_H)x + (a_L \oplus b_L)$.

Multiplication and inversion of two term-polynomials require modular reduction to ensure that the result is a two-term polynomial as well. For this purpose, one can use irreducible polynomial $n(x) = x^2 + \{1\}x + \{e\}$ whose coefficients have been chosen to optimize finite field computations.

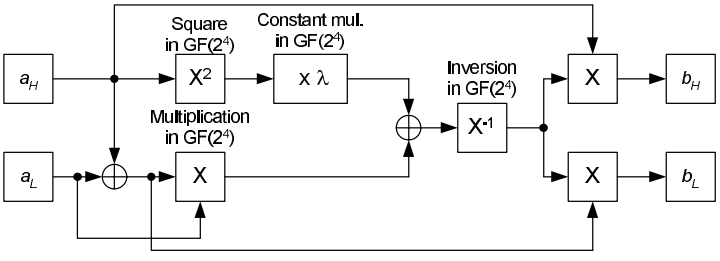


Fig. 4. Inversion in $GF((2^4)^2)$

Inversion of a two-term polynomial is defined as $(a_H x + a_L) \otimes (a_H x + a_L)^{-1} = \{0\}c + \{1\}$. From this definition the formulae for inversion can be derived:

$$(a_H x + a_L)^{-1} = (a_H \otimes d)x + (a_H \oplus a_L) \otimes d, \quad where$$
$$d = ((a_H^2 \otimes \{e\}) \oplus (a_H \otimes a_L) \oplus a_L^2)^{-1}.$$

Fig. 4 depicts a block diagram of inversion in the field $GF((2^4)^2)$. As one can see, one addition, one squaring, one constant multiplication, three general multiplications and one inversion in $GF(2^4)$ are necessary for its implementation. These operations can be realized in combinational logic as described in [32], and reminded below.

3.2 Arithmetic Operations in $GF(2^4)$

Addition in $GF(2^n)$ is a simple bitwise XOR. Multiplication and inversion in $GF(2^4)$ require an irreducible polynomial of degree four, e.g., $m(x) = x^4 + x + 1$.

Multiplication $q(x) = a(x) \otimes b(x) = a(x) \cdot b(x) \mod m(x)$ can be realized in combinational logic as follows:

$$\begin{aligned} q_0 &= a_0 b_0 \oplus a_3 b_1 \oplus (a_2 \oplus a_3) b_2 \oplus (a_1 \oplus a_2) b_3 \\ q_1 &= a_1 b_0 \oplus (a_0 \oplus a_3) b_1 \oplus a_2 b_2 \oplus a_1 b_3 \\ q_2 &= a_2 b_0 \oplus a_1 b_1 \oplus (a_0 \oplus a_3) b_2 \oplus (a_2 \oplus a_3) b_3 \\ q_3 &= a_3 b_0 \oplus a_2 b_1 \oplus a_1 b_2 \oplus (a_0 \oplus a_3) b_3. \end{aligned} \tag{1}$$

Here concatenation of two bits $a_i b_j$ represents binary multiplication, i.e., logical AND.

Squaring $q(x) = a(x)^2 \mod m(x)$ is a special case of multiplication, and can be computed with XOR operations only: $q_0 = a_0 \oplus a_2$, $q_1 = a_2$, $q_2 = a_1 \oplus a_3$ and $q_3 = a_3$.

An inverse $q(x) = a(x)^{-1} \mod m(x)$ of an element $a \in GF(2^4)$ is derived by solving the equation $a \otimes a^{-1} \mod m(x) = 1$, and is implemented in combinational logic as described below:

$$\begin{aligned} q_0 &= a_A \oplus a_0 \oplus a_0a_2 \oplus a_1a_2 \oplus a_0a_1a_2 \\ q_1 &= a_1(a_0 \oplus a_2 \oplus a_3 \oplus a_0a_3) \oplus a_0a_2 \oplus a_3 \\ q_2 &= a_0(a_1 \oplus a_2 \oplus a_3 \oplus a_2a_3) \oplus a_2 \oplus a_3 \oplus a_0 \\ q_3 &= a_A \oplus a_0a_3 \oplus a_1a_3 \oplus a_2a_3, \end{aligned} \tag{2}$$

where $a_A = a_1 \oplus a_2 \oplus a_3 \oplus a_1a_2a_3$.

As one can see, all arithmetic operations in the extension field are eventually reduced to the bit-wise logical XOR and AND functions.

4 Side Channel Attacks and Computations on Masked Data

Basically, side-channel attacks work because there is a correlation between the physical measurements taken during computations, such as power consumption [12, 20], EMF radiation [9, 25], time of computations [13], and the internal state of the processing device, which itself is related to a secret key.

Among attacks, the Differential Power Analysis (DPA) is the most dangerous [20]. It uses statistical analysis to extract information from a collection of power consumption curves obtained by running an algorithm many times with different inputs. Then an analysis of a probability distribution of certain points on the curves is carried on. It uses correlations between power consumption and specific key-dependent bits which appear at known steps of cryptographic computations.

For example, a selected bit b at the output of one *Sbox* of the first round of the AES algorithm will depend on the known input message and 8 unknown bits of the key. The correlation between power consumption and b can be computed for all 256 values of 8 unknown bits of the key. The correlation is likely to be maximal for the correct guess of the 8 bits of the key. Then an attack can be repeated for the remaining S-boxes.

There are many strategies to combat side-channel attacks. On the hardware level, the countermeasures usually include clock randomization [28, 14], power consumption randomization or compensation [8], randomisation of instruction set execution and/or register usage [17]. However, the effect of these countermeasures can be reduced by various signal processing techniques [6]. Software-based countermeasures include introducing dummy instructions, randomization of the instruction execution sequence, balancing Hamming weights of the internal data, and bit splitting. When each data bit is split into two shares, we get what is called *data masking*.

Data masking is one of the most powerful software countermeasures against side channel attacks [5, 12]. The idea is simple: the message and the key are

masked with some random masks at the beginning of computations, and thereafter everything is almost as usual. Of course, the value of the mask at the end of some fixed step (e.g., at the end of the round or at the end of a linear part of computations) must be known in order to re-establish the expected data value at the end of the execution; we call this *mask correction*.

A traditional XOR operation is used for data masking; however, a mask is arithmetic in $GF(2^8)$. The operation is compatible with the AES structure except for inversion in *SubBytes*, which is the only non-linear transformation. In other words, to compute mask corrections for all linear transformations in a round, we simply have to apply these transformations separately to masked bytes and to masks. Numerous efforts to find an efficient secure implementation of the *Sbox* on masked data had only a limited success.

The first attempt to transform masked data between Boolean and arithmetic operations in a secure way [19] was shown to be insufficient against DPA attacks; a more sound method for mask switching [11] involves too much computational overhead.

To overcome this difficulty, Akkar and Giraud [1] proposed so-called transformed masking, where first an additive mask is replaced by a multiplicative mask in a series of multiply and add operations in the field, after which usual inversion takes place, and finally, the transformation of multiplicative mask into additive mask is carried out again as a series of multiply and add operations. However, it was pointed out in [10, 30] that a multiplicative mask does not blind zero, which becomes an Achilles hill of the implementation.

In what follows we show how to overcome this problem by carrying on computations on masked data at the gate level.

4.1 XOR and AND on Masked Data

As one have seen, all arithmetic operations in the extension field are eventually reduced to bit-wide XOR and AND operations. Since XOR is a linear function, one can compute it directly on masked data.

Hence, the problem of "masked inversion" can be effectively reduced to computing binary AND on masked bits and on bits of the mask without ever revealing actual data bits in the process.

Our solution is based on algebraic properties of logic operations. We derive the solution by manipulating algebraic formulae. Denote masked bits via $\tilde{a} = (a \oplus r_a)$ and $\tilde{b} = (b \oplus r_b)$. Then

$$\tilde{a} \cdot \tilde{b} = (a \oplus r_a) \cdot (b \oplus r_b) = (a \cdot b) \oplus (a \cdot r_b) \oplus (r_a \cdot b) \oplus (r_a \cdot r_b). \tag{3}$$

In order to compute the mask correction $(a \cdot r_b) \oplus (r_a \cdot b) \oplus (r_a \cdot r_b)$ in a secure way, we want to express the terms $(a \oplus r_b)$ and $(b \oplus r_a)$ using only masked data $\tilde{a}$, $\tilde{b}$, and the bits of the mask r_a and r_b. From the equation $r_a \cdot \tilde{b} = (r_a \cdot b) \oplus (r_a \cdot r_b)$ we derive the method to compute $r_a \cdot b$ securely:

$$(r_a \cdot b) = r_a \cdot \tilde{b} \oplus (r_a \cdot r_b). \tag{4}$$

Similarly, from equation $r_b \cdot \tilde{a} = (r_b \cdot a) \oplus (r_b \cdot r_a)$ we compute:

$$(a \cdot r_b) = r_b \cdot \tilde{a} \oplus (r_a \cdot r_b). \tag{5}$$

Substituting corresponding terms in equation 3 for their values obtained in 4 and 5, and simplifying the result, we have:

$$(a \cdot b) = \tilde{a} \cdot \tilde{b} \oplus (r_a \cdot \tilde{b}) \oplus (r_b \cdot \tilde{a}) \oplus (r_a \cdot r_b. \tag{6}$$

Hence, the computations of the "mask correction" can be carried out without compromising the bits of actual data.

The question is if recycling previously used mask bits is good enough to make the whole construction robust. In other words, we have to prove that the joint random variable $(\tilde{a}, \tilde{b})$ is balanced and independent of the joint random variable (a, b) and all intermediate terms in the equation 6 are balanced and independent from a, b, $a \cdot b$, and $a \oplus b$.

However, this analysis is made redundant by the fact that all modern secure devices such as smart cards must have random number generator on board. Such generators are implemented in hardware and have a high output rate. Thus, generating random bits on-line is possible.

To achieve balanced and independent intermediate results, we use a freshly generated random bit z as a new mask, computing masked AND as follows:

$$a \cdot b \oplus z = ((((\tilde{a} \cdot \tilde{b}) \oplus (r_a \cdot \tilde{b})) \oplus ((r_b \cdot \tilde{a}) \oplus z)) \oplus (r_a \cdot r_b)).$$

An actual implementation of the masked AND operation is realized as a cascade of layers of logic gates. The first layer generates the necessary elementary AND-terms. All other layers produce variables in the form $(\alpha \oplus z)$, where z is independent on all the bits that have been used so far. Alternative constructions can be used as well stemming from the basic algebraic formulae relating the operations AND, OR and XOR in $GF(2)$.

4.2 Masked Inversion in $GF(2^4)$.

The final architecture for a masked inverter in $GF((2^4)^2)$ is depicted in Fig. 5. Here SMul and X^{-1} boxes represent blocks of combinational logic implementing a multiplier 1 and an inverter 2, where each "$\oplus$" is replaced with masked XOR, and each "$\cdot$" is replaced with masked AND, as discussed previously. M is a random mask, *(P xor M)* represents a masked input, and Z, W, F are new masks used to "refresh" the masked data at the end of each operation in $GF(2^4)$ and at the end of inversion in $GF((2^4)^2)$.

Notice, that the inverse field isomorphism map^{-1} and the affine transformation f are both linear operations, and they can be merged to optimize the gate count in the *Sbox*; the same holds for *map* and f^{-1} in decryption [16, 22]. This is how we implemented it. The final structure of the masked *Sbox* is depicted in Fig. 6. The duplicate datapath is used to compute the mask correction. Altogether, 1.2K gate equivalents is required to implement one *Sbox*, which is 25% better than the table lookup implementation.

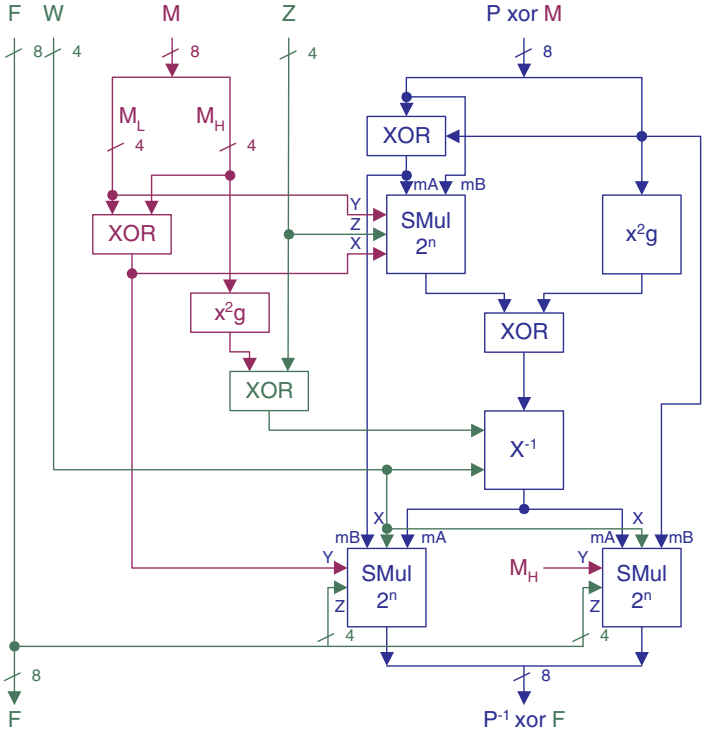


Fig. 5. Masked inversion in $GF(2^4)$

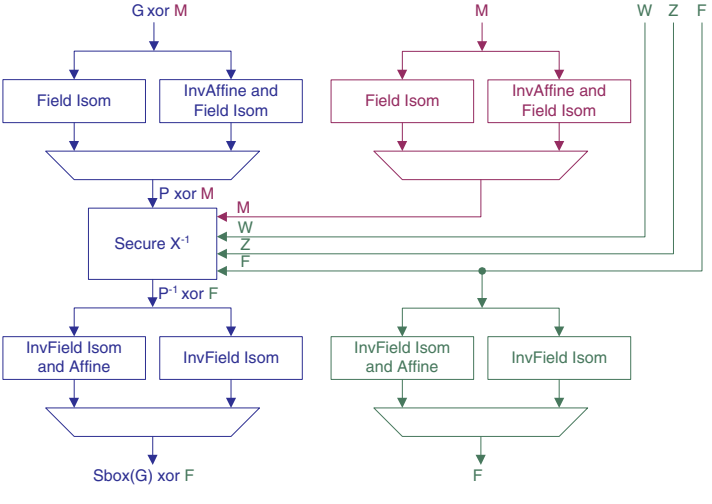


Fig. 6. Masked S-box

5 Secure AES Coprocessor

When manipulating masked data, all stages in a round, apart from inversion, require simple mask corrections in a form of analogous computations on masks that are carried out in parallel with the main computation flow. This can be achieved by duplicating hardware. The final architecture for a secure AES coprocessor is depicted in Fig. 7.

Another solution is to pipeline computations on masked data and on masks, which halves the throughput, and for which we need additional 128-bit registers. We had chosen the first approach because it is simpler conceptually, and it does not use more silicon than a pipelined architecture.

The hardware module based on the described scheme has been fully implemented in 0.18μm technology. The table in Fig. 8 gives the detailed gate count for a "minimalist" architecture where only one *Sbox* had been used for the main datapath. As one can see, a masked *Sbox*, where computations on bits of the masks are interleaved with computations on masked data, consumes relatively small amount of gates in comparison with the total amount of gates required for a whole datapath duplication. Thus, as a balance between the security, the throughput, and the gate count, four *Sbox*-es had been used for the main datapath in a real-life implementation,

The summary of the synthesis results for an original, i.e., unmasked, AES coprocessor and its masked counterparts is given in the table below. The last

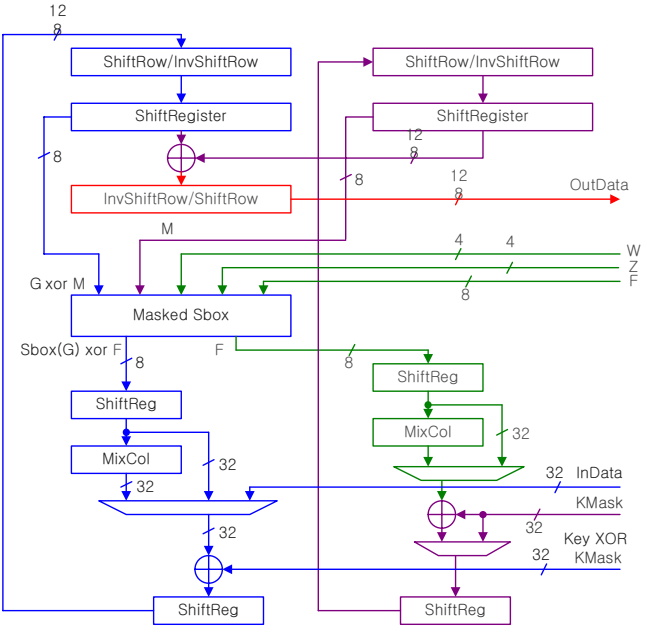


Fig. 7. Datapath for secure AES

Components	Subcomponents	Scalable key size	128 bit key size
Masked Sbox	Inverter in GF($(2^4)^2$)	1206	1206
	Input-output processing	350 (180)	350 (180)
Subtotal for masked Sbox (for key scheduler)		**1556 (1386)**	**1556 (1386)**
Masked datapath	Masked data	4900	4900
	Mask	2732	2732
Subtotal for masked datapath		**7632**	**7632**
Masked key scheduler	Masked key	6523	4835
	Mask	4913	3225
Subtotal for masked key scheduler		**11436**	**8060**
Control	Data control	300	300
	Key control	1140	400
TOTAL for masked AES		**20506**	**16390**

Fig. 8. Gate count for secure AES module

column gives the comparison with the minimalist architecture with only one *Sbox* in a datapath.

Parameter	Unmasked 4 Sboxes	Masked 4 Sboxes	Minimalist Masked 1 Sbox
Clk/Round	4	4	16
Gate count (128 bit key)	11.8K	21.7K	16.39K
Performance at 5MHz	16 Mbps	16 Mbps	4Mbps
Maximal performance	80 Mbps	74 Mbps	–
Power consumption	1.1 mA	2.1 mA	1.6 mA
Critical path	40 ns	43.18 ns	60 ns

The total increase in the gate count for a secure AES module with a fixed key size is 183%. However, it is a power consumption that is a real bottleneck in applications such as mobile communication. For a modern SIM-card with an AES coprocessor on board to be compliant with the GSM standard, the power consumption of the coprocessor itself must be around 1 mA. The power consumption even for a minimalist masked architecture is 1.6 mA in 0.18 μm technolog. However, with 0.13 μm technology the power consumption can be reduced to 1.07 mA, which allows us to use this module in applications such as GSM and ad-hoc networks.

6 Conclusion

We propose a new hardware implementation of the AES module secure against first order DPA attacks. Namely, we designed a combinational logic block to compute inversion directly on masked values without ever revealing actualdata in a process. Moreover, our approach solves the troubling problem of a zero attack.

The described approach can be used in other cryptographic coprocessors that use non-linear operations. It provides comparable protection as dynamic and

differential logic [29] and asynchronous dual rail circuits [23] at the similar (or slightly less) price in terms of the gate count and power consumption. Taking into account that the latter techniques require new logic libraries, careful "balancing" of place and routing, new development tools and longer time-to-market, our design offers a competitive alternative to hardware protection.

There are many interesting research and practical issues that remain unanswered. First of all, it would be good to understand exactly the amount of randomness that is actually needed to provide total protection against various DPA attacks. The first step in this direction was made in [3].

Secondly, while the data masking technique protects well against first-order DPA attacks, where an attacker observes correlations on only one intermediate value, it still may be vulnerable to higher-order attacks where up to N points on the power consumption curves (or, in other words, N intermediate values) can be observed at the same time [19]. In theory, a relevant countermeasure in this case should split each data bit into N shares, and carry on independent computations on each of the shares. Practically, it would amount to N-replication of hardware if we apply a similar technique.

Of course, in designing efficient and secure coprocessors for smart cards, other considerations, notably, production costs, have to be considered as well. Often it is not the best feature that counts, but overall design trade-offs. Investigation of such trade-offs is a challenging experimental problem. It is already known, for example, that clock randomization alone does not offer a sufficient protection even against first-order DPA because its effect can be relatively easy eliminated by clever processing of power curves using signal processing techniques [6]. However, a combination of clock randomization and data masking may make it very difficult to mount a higher-order DPA because an attacker would have to realighn simultaneously N points on power curves.

As a future work, we want to compare modules that work on masked data with other ASIC implementations of the AES algorithm for smart cards, where resistance to attacks is ensured by general-purpose countermeasures, such as variable clocks [28, 14], random current generators, balancing/duplicate logic [8, 4], random wait states, and their combinations [6].

References

1. Akkar, M., Giraud, C.: An implementation of DES and AES, secure against some attacks. Proc. Cryptographic Hardware and Embedded Systems: CHES 2001. Lecture Notes in Computer Science **2162** (2001) 309-318
2. Anderson, R., Kuhn, M.: Low cost attacks on tamper resistant devices. Proc.Security Protocols: IWSP 1997. Lecture Notes in Computer Science **1361** (1997) 125-136
3. Blmmer, J., Merchan J. G., Krummel, V.: Provably secure masking of AES. IACR Cryptology ePrint Archive Report 2004/101 (2004)
4. M. Bucci, L.Germani, M. Guglielmo, R. Luzzi, A. Trifiletti: A simulation methodology for DPA resistance testing of cryptographic processors (manuscript) 2003

5. Chari, S., Jutla, C., Rao, J., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. Proc. Advances in Cryptology – Crypto'99. Lecture Notes in Computer Science **1666** (1999) 398-412,
6. Clavier, C., Coron, J-S., Dabbous, N.: Differential power analysis in the presence of hardware countermeasures. Proc. Cryptographic Hardware and Embedded Systems: CHES 2000. Lecture Notes in Computer Science **1965** (2000) 252-263
7. Daemen, J., Rijmen, V.: *The design of Rijndael: AES - The Advanced Encryption Standard*. Springer-Verlag Berlin Heidelberg (2002)
8. Fruhauf, S., Sourge, L.: *Safety device against the unauthorized detection of protected data*. U.S. patent 5,404,402 (1995)
9. Gandolfi, K., Mourtel, C., Oliver, F.: Electromagnetic analysis: concrete results. Proc. Cryptographic Hardware and Embedded Systems: CHES 2001. Lecture Notes in Computer Science **2162** (2001) 251-261
10. Goliç, J., Tymen,Ch.: Multiplicative masking and power analysis of AES. Proc. Cryptographic Hardware and Embedded Systems: CHES 2002. Lecture Notes in Computer Science **2523** 198-212
11. Goubin, L.: A sound method for switching between boolean and arithmetic masking. Proc. Cryptographic Hardware and Embedded Systems: CHES'01. Lecture Notes in Computer Science **2162** (2001) 3-15
12. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. Proc. Advances in Cryptology – CRYPTO'99. K Lecture Notes in Computer Science **1666** (1999) 388-397
13. Kocher, P.: Timing attacks on implementations of Diffie-Hellmann, RSA, DSS, and other systems. Proc. Advances in Cryptology – Crypto'96. Lecture Notes in Computer Science **1109** (1996) 104-113
14. Kocher, P., Jaffe J., Jun, B.: *Using unpredictable information to minimize leakage from smartcards and other cryptosystems*, USA patent, International Publication number WO 99/63696 (1999)
15. Kommerling, O., Kuhn, M.: Design principles for tamper-resistant smartcard processors. Proc. USENIX Workshop on Smartcard Technology (Smartcard 99) (1998) 9-20
16. Lu, C. C., Tseng, S-Y.: Integrated design of AES (Advanced Encryption Srandard) encryptor and decryptor. Proc. IEEE conf. on Application-Specific Systems, Architectures, and Processors (ASAP'02) (2002) 277-285
17. May, D., Muller, H. L.,Smart, N. P.: Random register renaming to foil DPA. Proc. Cryptographic Hardware and Embedded Systems – CHES'01. Lecture Notes in Computer Science **2162** (2001)
18. Mangard, S., Aigner, M., Dominikus, S.: A highly regular and scalable AES hardware architecture. IEEE Transactions on Computers **52** no. 4 (2003) 483-491
19. Messerges, T.: Securing the AES finalists against power analysis attacks. Proc. Fast Software Encryption Workshop 2000. Lecture Notes in Computer Science **1978** (2000) 150-165
20. Messerges, T. S., Dabbish, E. A., Sloan, R. H.: Examining smart-card security under the thread of power analysis. IEEE Trans. Computers. **51** no. 5 (2002) 541-522
21. Messerges, T. S.: Using second-order power analysis to attack DPA resistant software. Proc. Cryptographic Hardware and Embedded Systems – CHES 2000. Lecture Notes in Computer Science **1965** (2000) 238-251
22. Morioka, S., Satoh, A.: An optimized S-Box circuit architecture for low power AES design. Proc. Cryptographic Hardware and Embedded Systems: CHES 2002. Lecture Notes in Computer Science **2523** (2003) 272-186

23. Moore, S., Anderson, R., Cunningham, P., Mullins, R., Taylor, G.,: Improving smart card security using self-timed circuits. Proc. Proceeding 8th IEEE International Symposium on Asynchronous Circuits and Systems – ASYNC'02. IEEE (2002) 23-58
24. Paar, C.: *Efficient VLSI architectures for bit parallel computations in Galois fields*. PhD Thesis, University of Essen, Germany (1994)
25. Quisquater, J. J., Samide, D.: Electromagnetic analysis (ema): measures and counter-measures for smart cards. Proc. Smartcard Programming and Security. Lecture Notes in Computer Science **2140** (2001) 200-210
26. Rudra, A., Dubey, P., Julta, C., Kumar, V., Rao, J., Rohatgi, P.: Efficient Rijndael implementation with composite field arithmetic. Proc. Cryptographic Hardware and Embedded Systems – CHES'01. Lecture Notes in Computer Science **2162** (2001) 175-188
27. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A compact Rijndael hardware architecture with S-Box optimization. Proc. Advances in Cryptology – ASIACRYPT 2001. Lecture Notes in Computer Science **2248** (2001) 239-254
28. E. Sprunk, *Clock frequency modulation for secure microprocessors*, USA patent number WO 99/63696 (1999)
29. Tiri, K., Akmal, M., Verbauwhede, I.: A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. Proc. IEEE 28th Europen Solid-State Circuit Conf. – ESSCIRC'02 (2002)
30. E. Trichina, E., De Seta, D., Germani, L.: Simplified Adaptive Multiplicative Masking for AES and its secure implementation. Proc. Cryptographic Hardware and Embedded Systems: CHES 2002. 2523 of Lecture Notes in Computer Science **2523** (2002) 277-285
31. Wolkerstorfer, J.,: An ASIC implementation of the AES MixColumn operation, In Proceedings Austrochip 2001 (2001)
32. Wolkerstorfer, J., Oswald, E., Lamberger, M.: An ASIC implementation of the AES S-Boxes. Proc. Topic in Cryptography – CT-RSA 2002. 2271 of Lecture Notes in Computer Science **2271** (2002) 67-78

Author Index