TECHNISCHE UNIVERSITEIT EINDHOVEN
Department of Mathematics and Computing Science

MASTER'S THESIS

Securing Epidemic Communication in Wireless Sensor Networks

by

I. van Vessem

Supervisors:

dr. S. Mauw (TU/e)
dr. ir. A.M. Bos (Chess iT)

Eindhoven, January 2006

Preface

This document presents the results of the research performed for the Master's project concluding the Information Security Technology study at the Eindhoven University of Technology. The research was performed from March 2005 until December 2005 at Chess iT located in Best. The project was supervised by dr. Sjouke Mauw of the Eindhoven Computer Science Security Group of the department of Mathematics and Computer Science and by dr. ir. Bert Bos of Chess iT. The assessment committee consisted of dr. Erik de Vink (acting chairman), dr. ir. Bert Bos, dr. ir. Berry Schoenmakers and dr. ir. Michel Reniers.

Acknowledgements

I would like to express my gratitude towards my supervisor Sjouke Mauw for his support in finding a suitable project, his guidance and support during the project, and his confidence. I'm also grateful to Chess iT and my company supervisor Bert Bos in particular, for giving me the opportunity to do my research there and for allowing me the freedom to choose the direction of my research. I also thank Erik de Vink for his assistance in finding a project and for his agreement to act as replacement chairman for the assessment committee. I thank Berry Schoenmakers for his helpful advice during the project and for joining the committee. I thank Michel Reniers for joining the committee on short notice. Thanks are due also to Maarten van Steen for his kind agreement to introduce us to the Gossip communication model.

Ivo van Vessem

Contents

1 Introduction **1**

2 Problem description **2**

3 Literature study **3**

3.1 Surveys . . . 3

3.2 Protocols . . . 4

3.3 Algorithms . . . 8

3.4 Energy efficiency . . . 8

3.5 Miscellanea . . . 9

3.6 Discussion . . . 9

4 Gossip primer **11**

4.1 Introduction . . . 11

4.2 Gossip protocol . . . 11

4.3 Suitability analysis . . . 12

5 Gossip analysis and optimisations **13**

5.1 Introduction . . . 13

5.2 Observations on the protocol description . . . 13

5.3 Optimisations . . . 14

6 Gossip formalisation **17**

6.1 Introduction . . . 17

6.2 Definitions . . . 17

6.3 Gossip protocol . . . 18

6.4 Validation . . . 18

6.4.1 Evaluation . . . 22

7 Solution space analysis **23**

7.1 Authentication . . . 23

7.2 Freshness . . . 23

7.3 Confidentiality . . . 24

7.4 Robustness . . . 24

7.5 Efficiency . . . 24

7.6 Introduction to solutions . . . 25

8 Single shared network key **26**
8.1 Authentication . . . 26
8.2 Confidentiality . . . 26
8.3 Freshness . . . 27
8.4 Robustness . . . 28
8.5 Efficiency . . . 28
8.5.1 Bandwidth . . . 28
8.5.2 Storage . . . 29
8.5.3 Computation . . . 29
8.6 Evaluation . . . 29

9 Unique asymmetric key-pair **30**
9.1 Authentication . . . 30
9.2 Freshness . . . 31
9.3 Confidentiality . . . 32
9.4 Discussion . . . 32
9.5 Robustness . . . 32
9.6 Efficiency . . . 33
9.6.1 Bandwidth . . . 33
9.6.2 Storage . . . 34
9.6.3 Computation . . . 34
9.7 Optimisations . . . 35
9.7.1 Replacing asymmetric encryption in message (3) . . . 35
9.7.2 Modification message (2) . . . 35
9.7.3 Symmetric key derivation . . . 36

10 Unique symmetric key **38**
10.1 Definitions/assumptions . . . 38
10.2 Authentication . . . 39
10.3 Key evolution . . . 39
10.4 Confidentiality . . . 39
10.5 Freshness . . . 39
10.6 Verification . . . 40
10.6.1 Signature forgery . . . 40
10.6.2 Confidentiality violation . . . 41
10.6.3 Forward security . . . 41
10.7 Robustness . . . 41
10.8 Implementation . . . 41
10.8.1 Hash function . . . 42
10.8.2 MAC function . . . 42
10.8.3 Encryption function . . . 42
10.8.4 Choice of security parameter sizes . . . 43
10.9 Efficiency . . . 43
10.9.1 Bandwidth . . . 43
10.9.2 Storage . . . 43
10.9.3 Computation . . . 43

11 Evaluation and comparison **44**

12 Conclusions **46**
12.1 Open problems and future research 46

A Inexpensive authenticated round numbers **49**
A.1 Construction 49
A.2 Distribution 50
A.2.1 Distribution through Gossip 50
A.2.2 Distribution through broadcasts 51
A.3 Evaluation 51

Bibliography **53**

1 Introduction

Wireless Sensor Networks are envisaged to consist of thousands of autonomous sensor nodes. Sensors communicate wirelessly with neighbouring sensors thus forming a sensor network. Sensors rely on these neighbours to forward their messages such that these messages eventually reach a base station where they can be processed. In this scenario the budget per sensor is severely limited, which translates to a limited amount of chip area and a low capacity energy supply. This in turn puts tight restrictions on the amount of available storage, the affordable computational complexity, the amount of data that can be transmitted, and the transmission range.

Yet, there is an obvious need for authenticating sensor readings and some scenarios call for confidentiality as well. Consider, for example, the case of smoke detection sensors where an attacker could trigger false fire alarms if there were no means to establish the origin of a message. Sensors used for military purposes are also a prime target for adversarial manipulation. In a scenario where sensors monitor the presence of persons or perhaps their health condition, a confidential information exchange is required for privacy protection.

This thesis strives to strike a balance between the resource constraints and the security requirements. Among others, we propose a scheme that achieves node to base station authentication with weak freshness and message confidentiality, yet has a modest computational complexity, minimal storage requirements, and quite acceptable message expansion.

We use a particular epidemic communication model, Gossip, as our starting point. Other than the robustness achieved by a highly redundant message exchange, the Gossip protocol specification has no security provisions whatsoever. The aim of this thesis is to define appropriate extensions to the Gossip model realising certain security objectives while minimising the overhead in terms of resource usage. Wherever possible we have defined the security extensions independent from the specifics of the Gossip model such that the solutions have relevance beyond the limited context of epidemic communication. This holds in particular for the solution presented in Chapter 10 that can be applied generally.

The remainder of this thesis is organised as follows. Chapter 2 provides a description of the problem setting. Chapter 3 summarises and discusses related work. Chapter 4 introduces the Gossip epidemic communication model. Chapter 5 continues with an analysis of the Gossip specification, identifies some defects, and suggests some optimisations. Chapter 6 formalises the Gossip protocol and shows it meets the identified requirements. Chapter 7 lists relevant evaluation criteria for the solutions to be proposed both with respect to security properties and resource requirements. Chapter 8 presents the first, simple solution that uses a single globally shared symmetric key. Chapter 9 uses asymmetric cryptography to secure communication. Chapter 10 introduces a novel forward secure solution that uses a unique symmetric key per node. Chapter 11 compares the three solutions with respect to resource requirements. Chapter 12 provides conclusions and various suggestions for further research. Appendix A finishes with a mechanism for inexpensive authenticated round numbers.

2 Problem description

Our problem setting is characterised as follows. An area has to be monitored with respect to certain environment variables. For this purpose, a collection of sensors is distributed over the area. The readings of these sensors must be transmitted to a base station for further processing. We assume wireless communication and only a limited number of sensors in the range of the base station. Sensors have no location awareness and the network topology is unknown to both sensors and base station.

We do not place any unreasonable restrictions on the attacker. In particular, since sensors are deployed in a potentially hostile environment and the limited budget does not allow for tamper proof sensor nodes, we have to assume that the attacker is able to capture one or more sensors and extract key material. In such a scenario the best we can accomplish at the cryptographic level is a forward secure scheme such that a node capture does not compromise the security of prior messages. At another level the situation in which not all sensor readings can be trusted can perhaps be remedied by placing redundant sensors and assuming the number of captured sensors has a sufficiently low upper bound. We do assume an invulnerable and reasonably powerful base station[1].

We assume the possibility of secure node initialisation, that is the base station is able to securely distribute an initial key or other relevant cryptographic information such that this information cannot be observed or influenced by a third party.

[1]Note that we refer to the central trusted authority as *the* base station but of course it could equally well be distributed over multiple physical base stations provided they each satisfy the given requirements (i.e. reasonable resources, invulnerability).

3 Literature study

At the beginning of the project an elaborate literature study has been performed. We include here a summary of the most relevant publications encountered.

3.1 Surveys

Security in Wireless Sensor Networks [33]

Discusses main security aspects of wireless sensor networks. Existing methods for key establishment are not applicable due to the limited computational capabilities. The simplest approach is a single key shared throughout the network, but it has the disadvantage that one compromised sensor results in the compromise of the security of the entire network. The variant where the shared key is destroyed after derivation of a session key has the disadvantage that dynamic insertion of sensornodes is prevented. Asymmetric cryptography requires too much computation but has the advantage that nodes can communicate with arbitrary other nodes. A shared key between every pair of nodes in the network is an approach that does not scale well with the size of the network. With a shared key between sensornode and base station, connections to other nodes can be set up via the base station. Nodes can also be equipped with an arbitrarily chosen subset of a large set of keys, where for well chosen parameters there is a large probability that a connected network will result. However, if a sufficient number of nodes is compromised, the complete set of keys can be determined.

Regarding confidentiality and authenticity, encryption from source to destination offers a high degree of security but it requires a shared key with all destinations and it does not allow for local broadcasts. Encryption from hop to hop with a key shared throughout the network enables local broadcasts but intermediary nodes can eavesdrop on the communication.

Tinysec [21] observes that costs of encryption are mainly caused by larger data packets and hence specialised hardware to decrease the computation cost has limited benefit. Privacy threats of sensor networks can occur if sensors are used by employers to monitor employees, by stores to monitor customers or by the government to monitor citizens.

Denial-of-service attacks constitute a serious threat to wireless sensor networks. An attacker can simply transmit a strong signal disrupting all communication. Also inserting forged routing messages or replaying previously sent routing messages can result in denial of service.

One of the largest challenges is mitigating the threat posed by attacks in which an attacker captures one or more sensor nodes. The attacker is then able to obtain secret keys, modify the sensor node's program code or replace sensors by sensors under her control. Constructing *tamper resistant* sensor nodes is a possible, yet expensive, solution. Therefore, algorithmic solutions are to be preferred. Sensor information could be sent along multiple routes such that at the destination manipulations can be detected. Sensors can also be deployed redundantly such that multiple sensors have to report a particular observation before it is considered valid.

Security considerations in ad hoc sensor networks [15]

Emphasises the difference between wireless sensor networks and ad hoc networks: sensors have a very low power consumption, severely limited computation and storage capacity, a typical sensor network contains an order of magnitude more nodes than a typical ad hoc network and the risk of failure of a sensor node is significantly higher than that of a node in an ad hoc network.

There is a conflicting interest between minimal resources for sensors and maximal information security. A sensor that regularly enters a sleep mode might not be able to receive messages essential to the security. Since tamper protection is too expensive, the possibility of one or more compromised nodes in the network has to be taken into account.

The following properties complicate the design of a security architecture. Nodes placed in an uncontrolled environment do not allow for regular physical inspection to detect tampering or to replace batteries. In order to make the deployment of a sensor network easy and flexible, pre-configuration strategies where for instance cryptographic keys are placed on nodes can only be used to a limited extent. The available bandwidth and data packet size limits the possibilities for overhead of data expansion caused by encryption. Since the wireless connection is unreliable, messages can easily be lost during transmission.

Based on an analysis of the threats at different OSI protocol stack layers (from physical to transport) the following problem areas are identified as most important: key management, secure routing and the prevention of denial of service attacks.

The following solutions are to some extent suitable for key management in wireless sensor networks. Due to the large differences in topology, density and size, no single protocol is optimal in every situation. A combination of symmetric and asymmetric encryption, including group keys, pairwise keys, and identity based keys will result in an energy efficient solution. Threshold cryptography where the CA is distributed over multiple specialised nodes is resistant to the compromise of a limited number of these nodes, but it requires asymmetric encryption capabilities at each node.

Lacking authentication of routing information enables an attacker to perform a DoS attack by manipulating the mechanism that establishes the topology of the network. SPINS [30], Pebblenets and INSENS are mentioned as one of few solutions to provide secure routing in wireless sensor networks.

Typical DoS attacks are transmitting disrupting radio signals, physically removing sensors, causing collisions between data packets, not forwarding incoming packets, sending large amounts of forged data packets that congest the network and consume energy, and disturbing mutual synchronisation. Existing solutions use reputation based systems, a virtual currency used to pay for data packets, or routing along multiple paths as a DoS prevention. However, these solution are mainly suitable for general ad hoc networks and less suitable for wireless sensor networks.

3.2 Protocols

Towards a new paradigm for securing wireless sensor networks [19]

Requires the installation of a set of keys on each sensor. Direct communication is only possible if peers share at least one key. The communication model used is from node to node, as opposed to node to base station.

A key-management scheme for distributed sensor networks [9]

References [6] for measures of the energy consumption of RSA versus AES encryption and the transmission of messages over various distances. The approach is comparable with [19], via nodes with a common key, nodes without a common key can be reached. Assumes tamperproof sensors, but damage is limited in case of a compromised sensor node.

Establishing pairwise keys in distributed sensor networks [25]

Provides a general model of which both polynomial based key distribution where each node is equipped with part of a polynomial in two variables and probabilistic key distribution are instances.

SPINS: security protocols for sensor networks [30]

Rather elaborate framework based on symmetric cryptography offering confidentiality, authenticity and freshness ("SNEP") and additionally, authenticated broadcasts from base station to nodes ("μTESLA"). It is assumed that base station and node share a secret $\mathcal{K}$. A message M is sent in SNEP as:
$\{M\}_{\mathcal{K}_{encr},C}, MAC(\mathcal{K}_{mac}, C|\{M\}_{\mathcal{K}_{encr},C})$ where C is a counter maintained by both node and base station and the keys are derived from $\mathcal{K}$. The counter ensures that multiple occurrences of the same plaintext do not lead to identical ciphertexts. The advantage over a random value to achieve that goal is that the counter value does not need to be included in the message, such that the size of the message does not increase as a result of encrypting it. The counter also provides weak freshness of the message in the sense that the receiver is able to conclude that the message was sent after the previous message it received with a lower counter value. The MAC in the message guarantees its authenticity and integrity. For strong freshness the following construction is suggested:
$A \rightarrow B : N_A, R_A$
$B \rightarrow A : \{R_B\}_{\mathcal{K}_{encr},C}, MAC(\mathcal{K}_{mac}, N_A|C|\{R_B\}_{\mathcal{K}_{encr},C})$
where N_A is the nonce of A, R_A is the request from A, and R_B is the reply from B. Since the nonce is not directly included in the reply message but is only indirectly used in the construction of the MAC, the length of the message is not increased. The nonce provides a unique match to the message in the previous request from A to B.

A block cipher is chosen to implement encryption, MAC computation and nonce generation security primitives. The block cipher is used in counter mode which effectively creates a stream cipher and which offers the advantage that encryption and decryption use exactly the same program code. With *Cipher Block Chaining* that same code can also be used to compute a MAC. Random numbers can be generated as follows: $MAC(\mathcal{K}_{rand}, \mathcal{C})$ where $\mathcal{C}$ is a counter. Derivation of keys also utilises the MAC: $\mathcal{K}' = MAC(\mathcal{K}, x)$ where x differs for each of the keys to be derived $\mathcal{K}_{encr}$, $\mathcal{K}_{mac}$ and $\mathcal{K}_{rand}$.

Minimalist block ciphers such as TEA and TREYFER are avoided due to lack of sufficient cryptanalysis. Opts for RC5 as it does not require multiplication or large tables. An implementation in TinyOS [13] requires 1594 byte, key initialisation 8000 cycles, encryption of 8 byte 120 cycles. The increase in energy consumption is predominantly caused by the increase in packet size due to the appended MAC. Sending the 8 byte MAC constitutes 20% of the consumed energy whereas its computation requires just 2%.

μTESLA requires synchronisation between node and base station. A hash chain is used of which preimages are released that authenticate previous messages. The initial commitment to the hash chain uses SNEP, after which nodes are able to verify that subsequently revealed keys belong to the hash chain. The disadvantages are the required buffer space at the node to store messages for which the authentication key has not yet been revealed, the requirement for a clock at the sensor nodes and the additional messages to be sent by the base station to reveal the keys.

A formal validation of the protocols is lacking as is forward security.

TinySec: a link layer security architecture for wireless sensor networks [21]

Refers to [13] for the costs of transmitting a single bit, these are equivalent with executing 800 to 1000 instructions. Observes that when aggregation of data or removal of duplicates in the network is desirable (in order to avoid large streams of data towards the base station), security from node to base station is unlikely to suffice. Moreover, the network is more vulnerable to DoS attacks when the integrity of messages can only be verified at the base station as valuable resources have been committed to deliver the message to the base station. The foregoing provides the motivation to study security at the level of the data link layer.

Offers authenticity and confidentiality but deligates replay protection to higher layers as the required maintenance of state for many communication peers is infeasible for memory limited sensor nodes.

Encryption uses a block cipher in CBC mode where *ciphertext stealing* is used to prevent data expansion. Authentication reuses the existing block cipher to construct a 4 byte CBC-MAC. Mentions the following possibilities for key distribution schemes: a single shared network key (with the obvious disadvantage that one captured node suffices to compromise the security of the entire network), a unique key for each node with which communication is required, or a shared key between a group of neighbouring nodes.

Extensive analysis of the implementation of TinySec on TinyOS shows that the increase in energy, bandwidth, and latency is limited to at most 10% as a result of encryption and authentication. The increase is mainly caused by larger data packets, computation contributes only marginally to the increase. Hence a software implementation will not perform much worse than a hardware implementation.

Secure pebblenets [3]

Nodes have a secret group key. Tamper resistance of the sensor nodes is assumed as is the absence of malicious nodes in the network. To limit the exposure of the shared group key to cryptanalytic attacks its use is restricted to establishment of a so called *Traffic Encryption Key (TEK)*. In a distributed key update algorithm one node is selected to determine a new TEK. No assumptions are made regarding the underlying communications model.

A security design for a general purpose, self-organizing, multihop ad hoc wireless network [28]

High level design of a security architecture based on IEEE 802.15.4 [1]. Offers security at MAC and NWK layer. In the simplest case a “secret” key is established by simultaneous button presses at two devices that trigger a key exchange algorithm. Supports shared secret key, public keys without certification and certified public keys coupled to the unique IEEE address. Offers methods for key update, orphan management (nodes that have lost connection with the network, requires unique 128 bit *orphan key*). AES is used for both encryption, hashing, and random number generation. Asymmetric encryption uses ECC.

Security considerations for IEEE 802.15.4 networks [32]

Describes vulnerabilities of IEEE 802.15.4 [1] as well as suggestions for solutions. Since the packet sequence number (introduced for replay protection) is not authenticated, a DoS attack is already successful with a single message in which the sequence number is set to the maximal value.

Information assurance in sensor networks [8]

Based on an urgency class resources of the network are committed to deliver a message to the *sink node* (no spread through the entire network as in the case of Gossip). A node knows its hop distance to the sink node and the error probability of the channel. A TDMA MAC layer is assumed. Utilises the broadcast property of the wireless channel.

SEKEN (Secure and Efficient Key Exchange for sensor Networks) [17]

Uses a model with communication restricted to communication from sensor to base station. The sensor shares a unique secret with the base station of which the sensor also has the public key. The shared secret key between sensor and base station is the MAC under the unique identifier of the sensor on a timestamp to be chosen by the sensor. Initially the shared key is sent to the base station encrypted with its public key. Communication between two sensors requires nonces and the intervention of the base station that supplies a shared secret for the two sensors. The base station maintains the topology of the network which requires methods for the addition and removal of nodes. Simulation suggests that the energy consumption is comparable to that of Kerberos.

Robust key establishment in sensor networks [38]

Information is sent along multiple non overlapping paths. If it can be assumed that an attacker controls only a limited number of nodes, a secret key can be established.

Assessing Security-Critical Energy-Efficient Sensor Networks [23]

Offers a fairly extensive overview of existing literature in the area of network protocols and key establishments. Distinguishes the following “critical system parameters”: data confidentiality, tamper resistance, public key cryptography capability, and availability of rich uncles (i.e. sensors with additional computational capabilities). Categorises existing solutions based on these criteria. Discusses code size and energy consumption of RC5 and XTEA.

Towards provable security for ad hoc routing protocols [5]

Has a restricted attacker model (attacker controls just a single node) but shows attacks against SRP and Ariadne. Not relevant to a Gossip context.

Epidemic-style Monitoring in Large-Scale Sensor Networks [12]

Node to base station is adopted as a communication model. Nodes have a unique identifier and a cache for storage of news items. It is assumed that at any point in time just a small fraction of nodes (termed “source nodes”) has a new item to be reported to the base station, the remaining nodes merely act as routers to forward items originating from other nodes. Nodes lack knowledge of the topology of the network and of the (relative) position of the base station. In each period a node selects from the nodes in its communication range a peer, chooses from its cache a set of items, sends these to the selected peer, receives from that peer a set of items and determines which of the items from its cache and the received items to save.

Different strategies can be used to determine the peer with which to engage in an exchange of cache items. If knowledge of the relative position of the base station is available, the neighbour closest to the base station can be selected with high probability in order to expedite delivery to the base station. To maintain sufficient robustness, with low probability a random node from the neighbours not closest to the base station is selected. Several simulation results are presented for various network topologies, cache sizes and strategies.

Problem areas for the use of Gossip in sensor networks are: the large amount of required storage (which is up to twice the size of the cache as a node can only determine which items to keep after having received the items from its peer and hence must have sufficient space to (temporarily) store those received items) and the large amount of data to be sent as a result of the inherent high redundancy. Another problem is that messages will linger in the network indefinitely unless a mechanism to remove items is introduced. It is suggested to maintain for each item the *lifetime* to be used to determine which items to delete.

3.3 Algorithms

Public Key Cryptography in Sensor Networks - Revisited [11]

The Rabin variant of RSA is used that requires just a single modular squaring for the public operation. With a 512 bit modulus an encryption using specialised hardware requires about 400 nJ. The gate count for this hardware amounts to 16726 gate equivalents and the computation uses 1440 clock cycles.

Reinventing the Travois: Encryption/MAC in 30 ROM Bytes [39]

Aimed at applications where even minimalist algorithms such as TEA and SAFER are to expensive. A block of available (non-cryptographic) machine code is used as an S-box.

3.4 Energy efficiency

Design considerations for distributed microsensor systems [7]

Lists various methods to harvest energy from environmental sources such as solar energy, vibrations, and magnetic fields and provides for each of these the expected yield.

Encryption overhead in embedded systems and sensor network nodes: modeling and analysis [34]

Provides measurements for RC4, IDEA, RC5, and SHA1 at five architectures and presents a model to estimate the performance of other algorithms at other architectures.

Supporting privacy in RFID systems [14]

Provides some measures for the chip area and energy consumption of XTEA, 3DES and AES obtained through simulation with GEZEL.

Analyzing the energy consumption of security protocols [31]

Cites [6] for the following measures: transmitting 1024 bit at a speed of 10 kbps consumes 21.5 mJ and receiving it requires 14.3 mJ. The RSA encryption of that same amount of data takes 42 mJ. Provides an extended overview of the energy consumption of hash functions (MD2, MD4, MD5, SHA, SHA1, and HMAC), symmetric algorithms (DES, 3DES, IDEA, CAST, AES, RC2, RC4, RC5, and BLOWFISH), asymmetric algorithms split up into key generation, encryption, and decryption (RSA, DSA, and ECDSA), and key exchange algorithms (DH and ECDH).

3.5 Miscellanea

Visualization of wormholes in sensor networks [36]

An attacker can create a so called *wormhole* by eavesdropping messages, forwarding them via an own fast channel and reinserting the messages at a different location in the network. A wormhole poses a threat to algorithms that establish the network's topology and routing protocols.

Reputation-based framework for high integrity sensor networks [10]

Non-cryptographic reputation system to assess the reliability of sensor information. Here nodes monitor the reliability of other nodes. Argues that due to the fact that a cryptographic solution will never be able to provide protection against malicious nodes intentionally spreading wrong information or simply benign malfunctioning nodes, in addition a reliability assessment should be implemented. Uses a Bayesian model.

Training a wireless sensor network [35]

Security is obtained by parameterised jumping through different frequencies.

Cryptographic tamper evidence [16]

When the secret key is retrieved by an attacker and the key subsequently continues to be used by the legitimate owner and the signature contains randomness (not pseudo randomness as the seed may have been revealed as well), then possibly it can be detected that the key has been revealed to a third party.

3.6 Discussion

Closest to our work, particularly the solution of Chapter 10 is SNEP, which is the part of [30] based around a single block cipher that is used for authentication, confidentiality, and random number generation. The key difference to note is that SNEP does not provide forward security. Although some empirical results [34] suggest that there is a notable performance penalty for using SHA-1 as our basic building block as opposed to RC5, the latter has the disadvantage that it is covered by a patent. Data from [31] indicates that the widely adopted AES block cipher, which would be a natural replacement for RC5, consumes at least 60% more energy per byte than SHA-1.

TinySec [21] is targeted at the security of node to node communication. The ability to detect in an early stage malicious messages and avoid committing resources to deliver these messages to the destination is used as motivation. We recognise the desirability of early detection possibilities, but failure to address replay attacks —which is justifiably omitted because the required maintenance of state for each communications partner is infeasible for memory limited sensor nodes— raises doubts on the usefulness of investing in authentication of node to node communication. Moreover, a key deployment mechanism is assumed of which the mentioned examples are either not robust (i.e. a single captured node compromises the entire security of the system) or require knowledge of the network topology.

Numerous studies make further assumptions that are incompatible with the scenario we consider. Secure pebblenets [3] relies on a tamper resistance assumption to protect a shared secret key. SEKEN [17] requires the establishment and maintenance of the network topology by the base station.

4 Gossip primer

4.1 Introduction

We will present here a brief introduction to the epidemic communication model of [12]. The discussion in this chapter is intended as a summary providing enough information for subsequent chapters. See also the short description included at the end of Section 3.2. In the next chapter several shortcomings and ambiguities are identified and some suggestions for optimisation are presented. Chapter 6 then continues with a formalisation of the Gossip protocol which is proven to eliminate the identified defects.

4.2 Gossip protocol

In the following we will focus on the "Shuffle Protocol" of [12, Section 5] as simulation results indicate that it outperforms the "Random Gossiping Protocol" with respect to the dissemination speed (i.e. the average time required for an item to reach all nodes) with no apparent disadvantages.

In the Shuffle Protocol the dissemination of sensor information is achieved by a randomised swapping scheme in which neighbouring nodes engage in an exchange of items held in a local cache. Through simulation it can be observed that the protocol, although limited to the specification of local interactions between neighbours, has desirable properties at macro level. Of particular interest is the average time required for an item to reach the base station.

Each node has a unique identifier that is included in each item that it inserts into the network. Besides the identifier items may contain additional, not further specified, information that is application dependent. Nodes are equipped with storage sufficient for a certain preset maximum number of items.[1]

Time is divided into so called rounds. In each round each node initiates precisely once an exchange with one of the nodes within its communication range. The initiating node is termed the active node, the selected neighbour is referred to as the passive node. The active node selects from its cache a collection of items and sends this collection to the passive node. The passive node replies in a similar way with a collection of items chosen from its local cache. Both nodes remove duplicate items as well as the minimal number of items from the ones that were sent necessary to store the newly received items within the preset maximum. Here, duplicate means having the same identifier.

The following requirements for the protocol are mentioned:

1. At any point in time a cache must contain at most one item from each source.

[1]The storage requirement for the nodes is actually up to twice the storage required for the maximum number of items as during an exchange a node has to temporarily store items received from its peer.

2. The receiving node stores all items received from its peer.

3. An exchange conserves information. Each item present in either the cache of the active or the cache of the passive node prior to the exchange is also present in either the cache of the active or passive node after the exchange. Although not mentioned explicitly, the only allowable exception for an item not to be present in either cache after the exchange is when it has been removed as a duplicate. Hence, if both caches contain an item from the same source, at least one of those items is contained in either of both caches after the exchange.

4.3 Suitability analysis

As can be seen from the simulation results presented in [12] an item is subject to a large degree of duplication, even in the case dissemination is not performed in a completely arbitrary way and the relative position of the base station is taken into account in the selection of the peer with which to engage in an exchange.

In favour of the Gossip model it can be argued that with the choice to limit the specification to local interactions, the need for topology discovery and maintenance disappears. Also expensive route determination mechanisms can be avoided.

Evidently there exists a considerable gap between the message duplication caused by the Gossip exchanges on the one hand and the optimal, though practically unachievable, scenario where each message is delivered precisely once via the shortest path from source node to base station on the other hand. This situation is aggravated by the fact that energy consumption is reportedly dominated by transmission cost rather than computation [21, 30, 13]. Hence we are faced with a mismatch between the energy characteristics of the Gossip communication model and the identified goal of minimal energy consumption of security extensions to the Gossip communication to be introduced.

In a scenario where this much energy is devoted to redundancy in messages, it does not seem particularly useful to strive for a security architecture with minimal energy consumption. Nonetheless we have maintained our original goal of inexpensive security and have defined our solutions independent from the specifics of the Gossip model as much as possible. This holds in particular for extensions defined at the level of individual items as these items are of a very general form with just the identifier as prescribed element. As a result, an alternative and less expensive underlying communication model could be used where the reduction in security-related energy consumption is more significant relative to the overall energy consumption.

5 Gossip analysis and optimisations

5.1 Introduction

We identify several ambiguities in the specification of the protocol policies in [12] that violate the desired property of data conservation in an exchange (requirement 3 of Section 4.2). Also requirements 1 and 2 are not met by the specification. We analyse which interpretation best suits the desired properties. These interpretations will be used in the formalisation of the protocol to be presented in the next chapter.

In addition, we propose some optimisations that remove redundancy that does not contribute to the robustness.

5.2 Observations on the protocol description

The second step in method `selectItemsToKeep()` in [12, §5.1] reads "Remove repeated items". We can distinguish the following three sets of items:

1. items in the local cache that have been sent to the peer
2. items in the local cache that have not been sent
3. items received from the peer.

The first two sets are disjoint by definition. The intersection of 1 and 3 is possibly non-empty. When a duplicate item (i.e. an item from the intersection of 1 and 3) is removed from set 3 as a result of the second step of method `selectItemsToKeep()`, that same item can subsequently be removed from set 1 as well based on step 3 of method `selectItemsToKeep()`. This violates the guarantee that the items received from the peer are saved (requirement 2 of Section 4.2). What is probably intended here is that duplicate items are to be removed from set 1.

The second paragraph of [12, §5.1] claims that a passive node need only keep those items it did not already have and it need not return to the peer items from the local cache (sets 1 and 2) also occurring in the received items (set 3). Not saving duplicate items contradicts the obligation to save the items received from the peer. Loss of information can occur if the passive node discards the duplicate item and additionally excludes the duplicate from the set of items to be returned to the active peer as the latter is free to select the duplicate item for deletion as it is part of the items that were sent. What is probably intended here is that in case at least one of the items received from the peer also occurs in the local cache, at most $s - 1$ items (not including the duplicate item) need to be selected from the items outside the intersection with 3 in order to be able to accommodate the at most $s - 1$ received items not already present in the local cache (the union of sets 1 and 2). Indeed in this case there is an opportunity to add an own new item to the set of items in the reply to the active peer, however the obligation to also keep the element of which a duplicate was already present remains.

When inserting new items by a source node, the specification does not exclude the possibility of the introduction of duplicate items. The paragraph cited above mentions that an item with the own id is to be placed in the cache whenever there is a slot available. What is not mentioned is that the local cache might already contain an item with the own *id*, which should be overwritten to avoid the introduction of duplicates.[1] Hence, also a completely filled cache has room for a new item, if an item with the own *id* is already present.

Also the second method for inserting new items, whenever at least one of the received items also occurs in the local cache, could introduce duplicates. Among the $s-1$ items selected from the local cache could be one with the own *id*. This would imply that the receiving active peer has to verify set 3 for duplicates. It would seem better to have the passive peer exclude items with its own *id* when selecting the $s-1$ items from its local cache such that no energy is wasted on sending duplicate items and the passive peer can assume the received set of items is free from duplicates.

A congestion of communication paths is reported [12, §6.2] when the neighbour closest to the base station is selected with near-maximal probability. However, it is completely unclear how this congestion follows from the simulation results as there is no mention of e.g. channel capacity or average message size.

5.3 Optimisations

In case of Directed Shuffle [12, §6] and the exchange of the complete cache (i.e. $s = c$), let node m be the neighbour of node n closest to the base station. Then there is a high probability that n selected m as its peer in the exchange of the previous period. Let c_m be the cache n received from m in the previous period. In the current period, in which with high probability n will again select m, it seems not particularly useful to return to m the same cache c_m received in the previous period. It would seem better for n to select a different peer and/or wait until a fresh cache has been received from one of its predecessors.

A passive peer can use the received items to select which items to return to its passive peer. It is useless to return an item to the active peer that also occurs in the set of items received from the peer, as the protocol guarantees that this duplicate item will be deleted by the active peer. However, if such an item is omitted from the return message, it could be selected by the active peer for deletion. Should this be undesirable (no information would be lost since the passive peer will keep the item), then it would be cheaper to include a reference to the item (e.g. a sequence number in the set of received items or just the *id* of the item) as opposed to thoughtlessly echoing the complete item back to the peer.

The base station need not return any items when in the passive role and it should not initiate any exchange (i.e. assume the active role).

The base station could send to its active peer an authenticated command instructing the active peer to delete the items just sent. Considering the previous optimisation, this command would replace the return message with items from the passive peer from an regular exchange. This command would help to achieve secure deletion of old data that has become superfluous since it has already been delivered to the base station. With P the number of peers within the communication range of the base station up to $s \cdot P$ cache slots per round would be freed.

[1] [12] does not make a distinction between old and new items from the same source. Probably this is inspired by [18] which explicitly mentions that a newer item replaces an older one even if that item has never been sent.

Remark that although in the base station's direct neighbourhood a considerable amount of storage space is emptied, these empty slots could influence subsequent interactions with neighbours further from the base station. If node n has just deleted (part of) its cache on command of the base station and recieves from node m a new set of items, n will reply with a non complete set of items (i.e. of size less than s). The fact that the reply is smaller is an improvement and may be sufficient justification for the introduction of the removal mechanism. However, by definition of the Gossip protocol, node m will not discard items from the ones that it sent beyond what is necessary to accommodate the items it received. Hence, the removal mechanism does not influence the amount of free storage outside the immediate neighbourhood of the base station and it causes prolonged conservation of items that would otherwise have been discarded. It would be interesting to analyse further the effectiveness of the suggested removal mechanism, for instance in terms of the average time an item remains in the network after the first delivery to the base station, see also Section 12.1.

Authentication of the delete command could be achieved either by

- A signature under a shared key on the set of received items (a relevant question is now to what extent this set is unique during the lifetime of the network). For the sensor, verification of the signature might prove to be relatively costly as the signature encompasses the complete set of sent items.

- Having the peer include in its message a nonce over which the base station generates a signature under the shared key. A disadvantage of this approach would be the added computational overhead for the sensor to generate the nonce and the message expansion resulting form the addition of the nonce.

- Utilising the already available shared state between node and base station (e.g. a message counter used for replay protection and the *id* of the node) to construct a fresh signature.

The suggested method of a *lifetime limit* [12, §8] to achieve deletion of old items has the disadvantage that too high a value results in the unnecessary and energy consuming continued forwarding of items that have long been delivered to the base station whereas too low a value would cause items to be prematurely deleted without having reached the base station. Determining an appropriate value is likely to be difficult especially if that value would have to be determined prior to deployment of the network and setting this value dynamically would easily add significant overhead to communicate securely to the nodes the newly determined lifetime limit. Moreover, the lifetime limit would introduce a weakness. When an attacker is able to influence the value of these counters, she could prevent items from reaching the base station or cause items to be circulated longer than is necessary to reach the base station, which could possibly lead to a DoS attack. To protect against this weakness the counter would have to be authenticated by each node. This increases the cost of both computation and transmission due to the further increase in message size. Additionally, it is now required that nodes are able to verify signatures generated by other nodes, which makes securing just the node to base station channel insufficient and hence is unlikely to be achieved cheaply.

A node acting as a source should limit the number of times its own item is added to the local cache as well as the number of times its own item is added to the items returned to an active peer.

The method to select which neighbour to initiate an exchange with could be refined such that the probability gradually increases with the proximity to the base station (as opposed to just assigning a higher probability to the single neighbour closest to the base station). A node that can determine with sufficient certainty that the base station is among its neighbours (for example because it received an authenticated command as suggested above), assigns probability 1 to this neighbour.

A downstream node could wait for a connection from an upstream node before initiating a connection with a node further downstream (i.e. closer to the base station). This way, an item could bridge multiple hops towards the base station in a single round. Nodes at the boundary of the network (i.e. nodes furthest upstream) should not initiate a connection unless there is an own new item to be reported. Perhaps this even eliminates the need for the notion of a round.[2] For this modification it would be interesting to see simulations to analyse the impact on the robustness.

Perhaps improvements could be made that utilise the broadcast property of the wireless channel. It would be interesting to determine the typical size of the intersection of the items sent by a node in the current and previous period. When there is typically a high overlap then peer p, selected in the current period, could have obtained much of the received information already in the previous period when those items were sent to peer p'.

[2]We remark that the term round suggests some kind of (global) synchronisation, of which it is unfortunately unclear how it is obtained. Without security provisions, the synchronisation mechanism is likely to be vulnerable to adverserial manipulation which could either prevent nodes from initiating exchanges or result in increased and energy depriving exchange frequencies. Securing the synchronisation mechanism might prove to be expensive.

6 Gossip formalisation

6.1 Introduction

The following formalised version of the Gossip protocol corresponds as closely as possible with the description of [12, §5.1] as summarised in Chapter 4 and follows the interpretation of Chapter 5 to avoid the loss of information in an exchange. The suggested optimisations of Section 5.3 have not yet been applied.

6.2 Definitions

Let N be the collection of nodes in the sensor network, C the capacity of the cache, U the maximum number of items to be exchanged with $U \leq C$. The contents of the cache c_n of node n changes during the execution of the protocol and is an element of $\mathcal{P}(N \times \mathit{Data})$ for unspecified data domain Data. We require that at any time the number of items in the cache does not exceed the maximum capacity, $|c_n| \leq C$, and that there is at most one item for each node $n \in N$: $(n, d), (n, d') \in c \implies d = d'$.

For convenience we will be a bit sloppy and use arbitrary $n \in N$ to refer both to node n as to the unique identifier of node n used for identifying the source of items in the network. We assume a bijection $id : N \to \mathit{ID}$ that maps nodes to identifiers from ID.

We introduce selection operator $\sigma : (N \times \mathit{Data}) \times (N \times \mathit{Data}) \to (N \times \mathit{Data})$ that selects in some not further specified way one item from two items (n, d) and (n, d'). Now we can define the following set operators that preserve uniqueness of source identities (i.e. sets contain at most one item for each node):

$$\begin{array}{rcl}
V \cup_1 W & = & \{e \mid e \in V \cup W \wedge (\forall e_v \in V, e_w \in W : \pi_1(e) = \pi_1(e_v) = \pi_1(e_w) : \sigma(e_v, e_w) = e)\} \\
V \cap_1 W & = & \{e \mid e \in V \cup W \wedge (\exists e_v \in V, e_w \in W : \pi_1(e) = \pi_1(e_v) = \pi_1(e_w) : \sigma(e_v, e_w) = e)\} \\
V \setminus_1 W & = & \{e \mid e \in V \wedge \pi_1(e) \notin \pi_1(W)\}
\end{array}$$

where $\pi_1(T)$ is the projection to the first elements of tuples $t \in T$, in particular $\pi_1(\{(n, d)\}) = n$.

6.3 Gossip protocol

For active node $a \in N$ and passive $p \in N, a \neq p$ we define an exchange as follows:

$$\begin{array}{lll}
 & a & : s_a \text{ s.t. } s_a \subseteq c_a \wedge |s_a| = |c_a| \downarrow U \\
(1) & a \rightarrow p & : s_a \\
 & p & : s_p \text{ s.t. } s_p \subseteq c_p \cup_1 b \wedge |s_p| = |c_p \cup_1 b| \downarrow U \wedge \\
 & & \qquad (b = \emptyset \vee (b = \{(p, data)\} \wedge ((c_p \cap_1 s_a) \setminus_1 s_p \neq \emptyset \vee |c_p| < C \vee |s_a| < U))) \\
(2) & p \rightarrow a & : s_p \\
 & a & : s'_a \text{ s.t. } s'_a \subseteq s_a \setminus_1 s_p \wedge |s'_a| = (|c_a \cup_1 s_p| - C) \uparrow 0 \\
 & & \quad c_a := (c_a \setminus_1 s'_a) \cup_1 s_p \\
 & p & : s'_p \text{ s.t. } s'_p \subseteq s_p \setminus_1 (s_a \cup_1 (b \setminus_1 c_p)) \wedge |s'_p| = (|c_p \cup_1 s_a| - C) \uparrow 0 \\
 & & \quad c_p := (c_p \setminus_1 s'_p) \cup_1 s_a
\end{array} \tag{6.1}$$

Where $x := y$ is the assignment of the value of y to variable x and $\downarrow$ and $\uparrow$ are respectively the binary minimum and maximum operator such that $x \downarrow y = -(-x \uparrow -y) = \begin{cases} x & \text{if } x \leq y \\ y & \text{if } y \leq x \end{cases}$.

Explanation Both a and p perform two internal actions, in addition there are two communication actions where a sends a message to p and vice versa. The communication actions are numbered above ((1) and (2)). s_n is the selection from the local cache c_n of items to be sent to the peer, of which the size $|s_n|$ is limited by both U and the size of c_n, for the passive peer possibly augmented with new item b. b can only be non empty if at least one of the conditions $(c_p \cap_1 s_a) \setminus_1 s_p \neq \emptyset$, $|c_p| < C$, and $|s_a| < U$ is met, see also below. The items s'_n to be removed from the local cache are selected from s_n, in so far as these do not also occur in the items received from the peer.[1] Whenever $b \notin c_p$ the passive peer must also exclude b from s'_p. The size $|s'_n|$ of s'_n is determined by the extent to which the union of the local cache and the items received from the peer exceed the maximal capacity $(|c_a \cup_1 s_p| - C)$. Taking the maximum with 0 prevents negative sizes for s'_n. Lastly, the new contents of the cache are determined by removing s'_n from and adding s_n to cache c_n.

In [12] it is implicitly assumed that caches are completely filled and the behaviour in case of non full caches is left unspecified. In addition to the optimisations of Chapter 5 we note that when a passive peer receives less than U items from its active peer, the latter apparently sent the complete contents of its cache. This implies that the active peer has sufficient room to accommodate C new items, even when $C > U$. It would improve the dissemination speed when the passive peer used this room and returned C items instead of the prescribed U items. We will further ignore this optimisation and follow [12].

A non full cache $c_p, |c_p| < C$ at the passive peer offers an extra opportunity to insert a new item, since p need only accommodate at most $U - 1$ items received from the peer, and hence only at most $U - 1$ items from the local cache need to be selected for the reply to the peer. For the same reason p can add its own item whenever $|s_a| < U$. A complete formal justification is provided in the following section.

6.4 Validation

As identified in Section 4.2 the conservation of information is a requirement for an exchange. In Chapter 5 we showed that the specification from [12] does not meet this requirement. We will

[1] We refer to Chapter 5 for the explanation of the exclusion of the elements received from the peer.

now show that the protocol specification of (6.1) does meet the identified requirements.

After completion of an exchange between nodes a and p the information contained in the caches of both nodes is determined by the union of $c'_a = (c_a \setminus_1 s'_a) \cup_1 s_p$ and $c'_p = (c_p \setminus_1 s'_p) \cup_1 s_a$. We show that the information contained in both nodes prior to the exchange is preserved, ignoring duplicate items.

Theorem 6.4.1. $c_a \cup_1 c_p \subseteq c'_a \cup c'_p$

Proof.

$$\begin{aligned}
c_a \cup_1 c_p &\subseteq (c_a \setminus_1 s_a) \cup_1 s_p \cup_1 (c_p \setminus_1 s_p) \cup_1 s_a \\
&\subseteq (c_a \setminus_1 (s_a \setminus_1 s_p)) \cup_1 s_p \cup_1 (c_p \setminus_1 (s_p \setminus_1 s_a)) \cup_1 s_a \\
&= (c_a \setminus_1 (s_a \setminus_1 s_p)) \cup_1 s_p \cup_1 (c_p \setminus_1 (s_p \setminus_1 (s_a \cup_1 (b \setminus_1 c_p)))) \cup_1 s_a \\
&\subseteq (c_a \setminus_1 s'_a) \cup_1 s_p \cup_1 (c_p \setminus_1 s'_p) \cup_1 s_a \\
&= c'_a \cup_1 c'_p \\
&\subseteq c'_a \cup c'_p
\end{aligned}$$

□

Remark. For both $b \in c_p$ and $b \notin c_p$, $(c_p \setminus_1 (s_p \setminus_1 (s_a \cup_1 (b \setminus_1 c_p))))$ reduces to $(c_p \setminus_1 (s_p \setminus_1 s_a))$.

The preservation of unique source identities within the new caches c'_a and c'_p follows easily by definition of $\cup_1$. Obviously the preservation of information can trivially be obtained by not discarding any of the items, but then the capacity C of the cache would be exceeded. We show that the size of the new caches c'_a and c'_p is at most C.

Theorem 6.4.2. $|c'_a|, |c'_p| \leq C$

Proof. If $|s'_a| = |s'_p| = 0$ it follows immediately that $|c_a \cup_1 s_p| \leq C$ and $|c_p \cup_1 s_a| \leq C$, hence we assume $|s'_a|, |s'_p| > 0$:

$$\begin{aligned}
& |c'_a| \\
= \quad & \\
& |c_a \setminus_1 s'_a| + |s_p| - |(c_a \setminus_1 s'_a) \cap_1 s_p| \\
= \quad & \{s'_a \subseteq s_a \subseteq c_a, s'_a \cap_1 s_p = \emptyset\} \\
& |c_a| - |s'_a| + |s_p| - |c_a \cap_1 s_p| \\
= \quad & \\
& |c_a| - |c_a \cup_1 s_p| + C + |s_p| - |c_a \cap_1 s_p| \\
= \quad & \\
& |c_a| - |c_a| - |s_p| + |c_a \cap_1 s_p| + C + |s_p| - |c_a \cap_1 s_p| \\
= \quad & \\
& C
\end{aligned}$$

and similarly for c'_p:

$$\begin{array}{ll}
 & |c'_p| \\
= & \\
 & |c_p \setminus_1 s'_p| + |s_a| - |(c_p \setminus_1 s'_p) \cap_1 s_a| \\
= & \{s'_p \subseteq s_p \setminus_1 (b \setminus_1 c_p) \subseteq c_p, s'_p \cap_1 s_a = \emptyset\} \\
 & |c_p| - |s'_p| + |s_a| - |c_p \cap_1 s_a| \\
= & \\
 & |c_p| - |c_p \cup_1 s_a| + C + |s_a| - |c_p \cap_1 s_a| \\
= & \\
 & |c_p| - |c_p| - |s_a| + |c_p \cap_1 s_a| + C + |s_a| - |c_p \cap_1 s_a| \\
= & \\
 & C
\end{array}$$

□

Furthermore the protocol requires certain sizes of sets s'_a and s'_p. Naturally the sets from which s'_a and s'_p are chosen must be of sufficient size. We derive this as follows.

Theorem 6.4.3. $|s_a \setminus_1 s_p| \geq |c_a \cup_1 s_p| - C$

Proof.

$$\begin{array}{ll}
 & |s_a \setminus_1 s_p| \\
= & \\
 & |s_a| - |s_a \cap_1 s_p| \\
\geq & \{s_a \subseteq c_a \implies |s_a \cap_1 s_p| \leq |c_a \cap_1 s_p|\} \\
 & |s_a| - |c_a \cap_1 s_p| \\
\geq & \{(|s_a| = U \geq |s_p| \wedge C \geq |c_a|) \vee (|s_a| = |c_a| \wedge C \geq U \geq |s_p|)\} \\
 & |c_a| + |s_p| - |c_a \cap_1 s_p| - C \\
= & \\
 & |c_a \cup_1 s_p| - C
\end{array}$$

□

Theorem 6.4.4. $|s_p \setminus_1 (s_a \cup_1 (b \setminus_1 c_p))| \geq |c_p \cup_1 s_a| - C$

Proof. We distinguish between the two possible sizes of s_p:

$$\begin{aligned}
\textit{Case}\quad |s_p| = |c_p \cup_1 b| \qquad & \quad |s_p \setminus_1 (s_a \cup_1 (b \setminus_1 c_p))| \\
= & \quad \{s_p \subseteq c_p \cup_1 b \wedge |s_p| = |c_p \cup_1 b| \implies s_p = c_p \cup_1 b\} \\
& \quad |c_p \cup_1 b \setminus_1 (s_a \cup_1 (b \setminus_1 c_p))| \\
= & \quad \{b \notin c_p \implies |c_p \cup_1 b \setminus_1 s_a \cup_1 b| = |c_p \cup_1 s_a|\} \\
& \quad |c_p \setminus_1 s_a| \\
= & \\
& \quad |c_p| - |c_p \cap_1 s_a| \\
\geq & \quad \{|s_a| \leq C\} \\
& \quad |c_p| + |s_a| - |c_p \cap_1 s_a| - C \\
= & \\
& \quad |c_p \cup_1 s_a| - C
\end{aligned}$$

If $b \in c_p$ or $b \notin s_p$ then $|s_p \setminus_1 (s_a \cup_1 (b \setminus_1 c_p))|$ reduces to $|s_p \setminus_1 s_a|$. Since $b \in c_p \vee b \notin s_p \implies s_p \subseteq c_p$ and assuming $|s_p| = U \geq |s_a|$, the derivation of $|s_p \setminus_1 s_a| \geq |c_p \cup_1 s_a| - C$ is analogous to that of $|s_a \setminus_1 s_p| \geq |c_a \cup_1 s_p| - C$. Hence assume $b \notin c_p$ and $b \in s_p$:

$$\begin{aligned}
\textit{Case}\quad |s_p| = U \qquad & \quad |s_p \setminus_1 (s_a \cup_1 (b \setminus_1 c_p))| \\
= & \quad \{b \notin c_p\} \\
& \quad |s_p \setminus_1 (s_a \cup_1 b)| \\
= & \quad \{b \in s_p\} \\
& \quad |s_p| - |s_p \setminus_1 b \cap_1 s_a| - 1 \\
\geq & \quad \{b \notin c_p \implies (c_p \cap_1 s_a) \setminus_1 s_p \neq \emptyset \vee |c_p| < C \vee |s_a| < U, ①,②,③\} \\
& \quad |c_p| + |s_a| - |c_p \cap_1 s_a| - C \\
= & \\
& \quad |c_p \cup_1 s_a| - C
\end{aligned}$$

$$\begin{aligned}
① \qquad & \quad |s_p| - |s_p \setminus_1 b \cap_1 s_a| - 1 \\
\geq & \quad \{|c_p \cap_1 s_a| \geq |(s_p \setminus_1 b) \cap_1 s_a|\} \\
& \quad |s_p| - |c_p \cap_1 s_a| - 1 \\
\geq & \quad \{\bullet\, |c_p| < C, |s_p| = U \geq |s_a|\} \\
& \quad |c_p| + |s_a| - |c_p \cap_1 s_a| - C
\end{aligned}$$

$$\begin{aligned}
② \qquad & \quad |s_p| - |s_p \setminus_1 b \cap_1 s_a| - 1 \\
\geq & \quad \{|c_p \cap_1 s_a| \geq |(s_p \setminus_1 b) \cap_1 s_a|\} \\
& \quad |s_p| - |c_p \cap_1 s_a| - 1 \\
\geq & \quad \{\bullet\, |s_a| < U, |s_p| = U > |s_a|, |c_p| \leq C\} \\
& \quad |c_p| + |s_a| - |c_p \cap_1 s_a| - C
\end{aligned}$$

$$\begin{aligned}
③ \qquad & \quad |s_p| - |s_p \setminus_1 b \cap_1 s_a| - 1 \\
\geq & \quad \{\bullet\, c_p \cap_1 s_a \setminus_1 s_p \neq \emptyset, (s_p \setminus_1 b) \subseteq c_p \wedge c_p \cap_1 s_a \setminus_1 s_p \neq \emptyset \implies |c_p \cap_1 s_a| > |(s_p \setminus_1 b) \cap_1 s_a|\} \\
& \quad |s_p| - |c_p \cap_1 s_a| \\
\geq & \quad \{|c_p| = C, |s_p| = U \geq |s_a|\} \\
& \quad |c_p| + |s_a| - |c_p \cap_1 s_a| - C
\end{aligned}$$

□

6.4.1 Evaluation

Notice that $b \neq \emptyset \not\Rightarrow b \in s_p$, that is, the fact that there is an own new item and the possibility to insert it into the set of items to be returned to the active peer, does not guarantee that that item is actually included in s_p. The protocol is not formulated more restrictively than is necessary to meet the requirements. It defines precisely under which conditions an item *can* be inserted, but it does not enforce actual insertion of the item. Of course implementations can use this freedom and give preference to b when selecting s_p from $c_p \cup_1 b$.

7 Solution space analysis

We identify the following criteria at which the extensions to (6.1) to be presented in Chapters 8, 9, and 10 can be evaluated. Solutions will be evaluated with respect to both security and efficiency considerations.

7.1 Authentication

We evaluate the possibilities the solution offers to establish the authenticity of sensor information. We can distinguish authenticity verification at the level of individual cache items and at the level of the complete set of exchanged items. The advantage of verification at the level of individual items is that there is no need to generate a signature on the complete set of items at each exchange and it suffices to generate a signature just once when an item with a new sensor reading is constructed at a source node. The disadvantage is that the amount of data to be exchanged is increased by $U - 1$ signatures. Under the assumption that energy costs are dominated by communication [21, 30, 13], a signature over the complete set of exchanged items is preferable.

A disadvantage of authentication at the level of complete caches is that the base station is unable to verify that an element (n, d) actually originates from node n. As soon as an attacker has knowledge of an arbitrary key used for authentication of exchanged caches, she can construct items for any node $n \in N$ and send those to a peer. Eventually the base station will attribute the items to the sensors with the identities chosen by the attacker. Hence, just one compromised key used for authentication results in a potential compromise of the authenticity of items from all nodes in the network.

A second distinction can be made at the location where the authenticity can be verified. When nodes are able to verify signatures, illegitimate items can be detected and discarded in an early stage avoiding the useless forwarding of items through the network only to be identified as false after having reached the base station.

Note that it is conceivable that the level at which the authenticity can be established depends on the location. Perhaps nodes can only verify the authenticity of the complete set of received items whereas the base station can verify the authenticity at item level. Nodes could for instance be able to verify signatures generated by their neighbours, but not the individual items that can originate from anywhere in the network. In these cases it is desirable to combine authenticity of individual items with authenticity of sets of exchanged items.

7.2 Freshness

Authenticity alone does not prevent an attacker from inserting previously sent and authenticated information back into the network. Hence we also evaluate the proposed solutions with respect to the extent to which the freshness of information can be established. We can make a similar distinction as with authenticity, i.e. freshness at the level of individual items or at the level of sets of exchanged items. For the verification of freshness there must be some form of shared state. It is unlikely that nodes can maintain this state for all sensors with which it communicates during the lifetime of the network. The consequence is that old items inserted by an attacker are spread through the network because nodes are unable to detect the replay.

7.3 Confidentiality

In the adopted model where only the base station needs to process information, there is no need for nodes to decrypt information from other nodes. Hence all information can be encrypted for the base station. It does not seem advantageous to encrypt the complete set of items instead of encrypting individual items. An attacker would then be able to decrypt all incoming items after having captured a sensornode. Moreover, encrypting and decrypting the set of items at each exchange is considerably more expensive than the one time encryption of an item for the base station. It is important to note that the Gossip protocol requires the source identities of items to be visible to other nodes in order to be able to detect and discard duplicate items. When just the data element d of an item (n, d) is encrypted for the base station and the set of exchanged items remains unencrypted, a (passive) attacker can learn the identities of the nodes of which items are exchanged. Should this be undesirable, encryption at set level could be added. In that case it is preferable to maintain encryption at item level such as to avoid revealing data elements to intermediary and possibly compromised nodes (assuming encryption for the base station uses a different key).

When information is encrypted for the base station, nodes are preferably not able to learn that information. If additionally nodes should be able to verify the authenticity of items, the signature will have to be defined over the cipher text and not the plain text. This option offers the additional advantage that there is no need for a potentially expensive decryption of not yet authenticated information.

An additional evaluation criterion is the *forward secrecy* offered by the different solutions. We can further distinguish between the effect on previously encrypted information of the compromise of a cryptographic key from a sensornode and from the base station. Under the assumed invulnerability of the base station the latter option is a theoretical one.

7.4 Robustness

We evaluate the effect of the capture of one or more sensornodes by an attacker. We assume the attacker is able to extract from captured sensors all information, including key material, contained in them. Note that the Gossip communications model already offers some robustness against insertion of false or outdated information. The Gossip model allows at most one item for each source in a single cache. If an attacker after capturing a particular sensornode is limited to inserting items of just that node, the Gossip model guarantees that at most N storage slots will be used for those items.

7.5 Efficiency

We evaluate the efficiency of the proposed solutions at the following points:

Bandwidth Additional data might need to be sent as a result of added signatures, additional messages or data expansion resulting from encryption.

Storage Keys, nonces or counters require additional storage space. Furthermore, extensions at set level have relevance only to the peers engaged in an exchange. Encryption at set level needs to be removed in order to construct a new set of items for subsequent exchanges. Likewise, authenticity and freshness extensions at this level are only relevant for the current set of items. On the other hand, extensions at item level remain relevant throughout exchanges. Once encrypted and once appended with authenticity or freshness information, items need not be modified by intermediary nodes. Hence we can directly obtain the extra amount of storage needed from the analysis of the bandwidth requirements.

Computation Encryption, decryption, hash computation, and nonce generation will be evaluated with respect to computational complexity.

7.6 Introduction to solutions

In the next chapters we present three solutions that will be evaluated with respect to the criteria as defined above. Chapter 8 uses a single globally shared symmetric key, Chapter 9 uses a unique asymmetric key pair for each node, and Chapter 10 uses a unique continuously evolving symmetric key per node. Finally, Chapter 11 will summarise the efficiency characteristics of the three solutions.

8 Single shared network key

We assume that each node in the network knows network key k. This key can be placed securely on the nodes during initialisation.

8.1 Authentication

We can now define the following extensions to the protocol messages of (6.1):

$$\begin{array}{llll} (1) & a \rightarrow p & : & s_a, h_k(s_a) \\ (2) & p \rightarrow a & : & s_p, h_k(s_p) \end{array} \tag{8.1}$$

where h_k is a cryptographic MAC parameterised by key k. The properties of h will be elaborated on in Section 10.2.

This protocol offers authentication of sent cache items: if for received cache items s signature $h_k(s)$ matches the received signature, it can be assumed that the sending party knows key k. A passive node will abort an execution of the protocol whenever the received signature does not match the computed one. Similarly, an active node will not add the cache items received from the passive peer to its local cache if the signatures do not match.

An alternative is to authenticate individual cache items. Cache items will then become triples of the form $(n, d, h_k(n, d))$. Because only one key is used throughout the network, authentication of individual items does not offer advantages compared with authentication of sets of items. Once an attacker knows k, a valid signature can be constructed both for sets of items as well as individual items. Since one signature on the complete set saves $U-1$ signatures to be exchanged, this option is preferable.

8.2 Confidentiality

If besides authentication also confidentiality is desired, the protocol can further be extended as follows:

$$\begin{array}{llll} (1) & a \rightarrow p & : & E_k(s_a), h_k(E_k(s_a)) \\ (2) & p \rightarrow a & : & E_k(s_p), h_k(E_k(s_p)) \end{array} \tag{8.2}$$

where E_k is the symmetric encryption with key k. The properties of E will be elaborated on in Section 10.4.

For confidentiality it is irrelevant at what level encryption is performed. Knowledge of key k enables an attacker to decrypt both individual items and sets of items. An item can therefore also be encrypted once: $(n, E_k(d))$ and decrypted only at the base station. Encryption and decryption at each exchange is now no longer required. Efficiency considerations will determine the choice between both alternatives, see Section 8.5.

8.3 Freshness

In a scenario where a single network wide key is the best that can be achieved, it is unlikely that nodes are capable of generating nonces or can be equipped with a clock and additionally messages can be lengthened to include nonces or time stamps. Therefore we limit our analysis to the use of message counters to achieve freshness.

Nodes operating as a source node maintain a message counter c which is increased for each item added. The base station is supposed to store for each source node the highest received counter value and accept only items of which the counter value is greater than the stored value.[1] With authentication at the item level, the signature can be redefined such that also the counter value is authenticated: $(n, d, \mathsf{c}, h_k(n, d, \mathsf{c}))$. Note that the counter would not have to be included for the base station to establish the freshness, as the base station can anticipate a limited number of lost messages and calculate for a limited number of counter values following the highest stored value the signature on the data and compare it with the received signature. However, as noted earlier, it is probably not feasible for nodes to maintain the state of other nodes, here in the form of message counters. Hence when the counter is only indirectly included as input to the signature, other nodes without knowledge of the counter value are unable to verify the authenticity of the item. Authenticity verification by the nodes combined with freshness verification by the base station consequently comes at the price of an increase in the size of messages (1) and (2) with U counter values.

When encryption is used, the counter value can be implicitly embedded in the message without affecting the possibility of authentication by nodes. By supplying c as an extra parameter to encryption function E, the base station is able to verify the freshness.[2] Cache items are now of the form $(n, E_{k,\mathsf{c}}(d))$. Probably this additional input was already required for the underlying encryption algorithm in order to prevent that repeated occurrences of the same plain text result in identical cipher texts.

Verification of freshness at the level of sets of items hardly offers any protection against the insertion of replays by an attacker. As nodes are not capable of assessing the freshness, an attacker is able to send previously received authenticated sets with now outdated items to an arbitrary node. As the freshness information at set level will never reach the base station (as much as the cache in the current composition will not reach the base station), the base station will also be unable to detect the replay. The only limitation offered by freshness at set level is that an attacker is unable to directly send a replay message to the base station without the replay being noticed.[3]

We can somewhat limit the possibilities to spread replays of sets of items by using the shared state that follows implicitly from an execution of protocol (8.1). Although not necessarily the case, it seems natural for an implementation of the protocol with communication over a shared medium to utilise the unique identifiers from N to designate sender and receiver of a message. It would then not be unrealistic to assume peers a and p have knowledge of each other's identities. By extending the definition of the signatures to also encompass these identities:

$$\begin{array}{lllll} (1) & a \rightarrow p & : & s_a, h_k(a, p, s_a) \\ (2) & p \rightarrow a & : & s_p, h_k(p, a, s_p) \end{array} \tag{8.3}$$

an attacker without knowledge of key k is no longer able to insert previously obtained messages at an arbitrary location in the sensor network.

[1] We refer to section Section 10.5 for a discussion of a more sophisticated item acceptance policy.

[2] Concrete options for E where c can be used as additional parameter include a block cipher used in counter mode.

[3] Remark that with freshness at set level counter c_n of node n counts the number of exchanges of n.

With authentication at the level of sets of items, an attacker without knowledge of key k is unable to insert replays into sets authenticated with k. An attacker therefore has to know k to insert replays of individual items. In that case freshness information is useless as it can be modified by the attacker, assuming the attacker has obtained previous messages and knows the value of the message counter of the node such that acceptable counter values can be used. Without knowledge of k the possibilities of the attacker are limited to replaying previously received sets s. With freshness at item level the base station is able to detect the replay and discard the outdated items. Items are now of the form (n, d, c).

Lastly we note that message counters offer a weak form of freshness only. A counter does not provide information on the delay a message encountered under the influence of an attacker (or otherwise) since it was inserted into the network. The message counters merely introduce an order on the messages such that a message of node $n \in N$ with counter value $\mathsf{c}_n = i$ was inserted after messages with counter values $\mathsf{c}_n = 0 \ldots i-1$.

8.4 Robustness

Obviously the major disadvantage of a single network wide key is that an attacker need only capture a single sensor node and extract key k to be able to insert items and decrypt encrypted items. An advantage is that nodes are able to assess the authenticity, avoiding the needless spreading of illegitimate items through the network only to be detected as false at the base station.

A replay of a previously sent authenticated message will be spread through the network as nodes are unable to detect the replay. Although the base station will identify the message as a replay, valuable resources have been committed to deliver the replay to the base station.

As soon as an attacker knows k a simple and effective DoS attack is possible by constructing items with the highest possible counter value after which the base station will no longer accept any messages from the legitimate node as these will carry lower counter values.

8.5 Efficiency

8.5.1 Bandwidth

The cost of authentication of the exchanged cache items in (8.1) consist of two additional signatures to be sent per exchange. It depends greatly on U, the typical size of a cache item (n, d) and the size of the signature ([27, §9.3] advises 64 bit for a MAC) whether this will cause a significant increase in message size.

The choice of the encryption algorithm influences the extent to which cipher text $E_k(m)$ is greater than m. Obvious choices for E include stream and block ciphers. The latter has the disadvantage that the size of the cipher text is a multiple of the block length. Encryption will then result in an expected data expansion of half a block length. When encryption is performed at item level, the expected overhead per exchange is U block lengths. With encryption at set level the expected overhead is 1 block length.

Use of a stream cipher is to be preferred as the length of the cipher text is exactly equal to the length of the plain text under any circumstances. An inexpensive option [30] that also offers freshness is the use of a block cipher in *Counter mode*, effectively creating a stream cipher. When the base station maintains for each node the last counter value used, the counter does not need to be included and encryption and (weak) freshness is obtained without any data expansion.[4]

[4]As each node uses the same k and message counters typically count from 0, without further measures the

Without encryption with implicitly embedded counter value, the possibility to establish the freshness requires per exchange an additional $2U$ counter values to be transmitted, independent of the level at which authentication is performed.

In summary, the amount of additional data to be sent can be limited to 16 byte per exchange when we opt for encryption with a stream cipher combined with indirect inclusion of the message counter.

8.5.2 Storage

Symmetric key k requires a few bytes (typically 8) of storage space at the sensor nodes. As noted previously, we can take the amount of additional storage required for the extensions at item level directly from the analysis above of the bandwidth requirements. Each source node requires a few bytes to store the message counter.

8.5.3 Computation

For each exchange both peers have to determine a signature on the complete set of items to be sent as well as the complete set of received items. According to the results of [31] for a HMAC this would require 1.16 μJ/B. Encryption with AES as block cipher with a key length of 128 bit costs somewhere between 1 and 2 μJ/B, depending on the particular mode in which AES is used. Derivation of the round keys consumes about 8 μJ.

Now we can make the choice between encryption at item and set level that was left open earlier. When encryption does not cause data expansion, preferably encryption is done once at item level by the source node inserting a new item into the network thus avoiding repeated encryption and decryption of complete sets of items at each exchange. Should encryption result in data expansion, an expected $U - 1$ block lengths of additional data to be sent per exchange can be saved by encrypting at set level. A disadvantage of encryption at set level is that the message counter can no longer be implicitly embedded in the item. The message counter will have to be included directly in the item. For 4 byte message counters an additional $8U$ byte of data would have to be transmitted which, for a 16 byte block length, halves the saving achieved by encryption at set level. Furthermore, in that case $4C$ byte of additional storage space is required.

8.6 Evaluation

For the sake of simplicity we have taken the use of a single network-wide key to the extreme and have used it for both authentication and encryption. This is generally considered bad practice (see e.g. [27, §13.5]) as weaknesses in either authentication or encryption could transfer to the other. We could mitigate this potential vulnerability by introducing some key derivation scheme to obtain different and, preferably, (computationally) independent authentication and encryption keys. This strategy is followed by [30] where a one way function is used to derive keys from a master key. However, this would add significantly to the computational requirements of the sensor nodes. Instead we suggest to simply place two keys k_a and k_e on each node during initialisation, adding just a few bytes to the storage space required on the node.

same key block is used N times, practically voiding all confidentiality. By using the unique identifier n combined with the message counter, reuse of the same key block in the network can be avoided.

9 Unique asymmetric key-pair

We assume each node $n \in N$ has a key pair PK_n, SK_n where PK_n is the public key and SK_n is the matching private key of node n. Furthermore each node knows the public keys of nodes within its communication range. We adapt the Needham-Schroeder-Lowe protocol such that we incorporate the sets of cache items in existing messages in order to limit the number of messages required for an exchange.

$$\begin{array}{llcl} (1) & a \rightarrow p & : & \{a, nc_a\}_{PK_p}, s_a, h_{nc_a}(s_a) \\ (2) & p \rightarrow a & : & \{nc_a, nc_p, p\}_{PK_a}, s_p, h_{nc_p}(s_p) \\ (3) & a \rightarrow p & : & \{nc_p\}_{PK_p} \end{array} \tag{9.1}$$

Where nc_n is the *nonce* of node n and $\{m\}_{PK_n}$ is the encryption of message m with the public key of node n. After completion of the protocol nodes a en p are communicating with each other and nc_a and nc_p are shared secrets between a and p. Only then both nodes can verify the authenticity and freshness of the received set of items, as the nonces are used as keys for h. The freshness of the messages is also guaranteed by using the nonces.

9.1 Authentication

As noted before, in case of authentication at set level it suffices to obtain one arbitrary SK_n to construct items with an arbitrary identifier and eventually have these accepted at the base station. We can limit the consequences of the compromise of a single private key by authenticating individual cache items. Cache items now become tuples of the form $(n, \{d\}_{SK_n})$. Where $\{d\}_{SK_n}$ is data d expanded with a signature under private key SK_n on d. For the moment we leave unspecified exactly how this signature is constructed. It is conceivable that a hash $h(d)$ is determined over which a signature under key SK_n is placed. A less expensive solution might be to use SK_n as a symmetric key: $h_{SK_n}(d)$. The assumption with the latter option is then of course that the party verifying the signature has knowledge of private key SK_n, which in practice implies that the signature can only be verified at the trusted base station as private keys, by nature, cannot be placed on other sensor nodes.

Above we have assumed that nodes have the public keys of other nodes within their communication range. As items can originate from any node in the network, nodes are unable to establish the authenticity of all incoming cache items. Therefore it seems useful to maintain the signature $h_{nc_n}(s_n)$ on the set of items. If this signature were to be omitted, an opportunity for a DoS attack is created. An attacker would then be able, without knowledge of any key, to replace from message (1) element s_a with a set of items of which the identifiers do not exist in the network. These false items would then be spread through the network only to be detected at the base station.

9.2 Freshness

So far we have accomplished that an attacker will not succeed without knowledge of SK_n to construct items of node n and have these accepted at the base station. However, the base station is unable to assess the freshness of items. An attacker can therefore, with knowledge of an arbitrary private key, replay previously sent authenticated items. We introduce a counter for each node acting as a source that is increased each time a new item is constructed. Cache items now become tuples of the form $(n, \{d, \mathsf{c}_n\}_{SK_n})$ where c_n is the counter of node n. The base station now has to maintain for each source node the maximal counter value seen so far such that items with a lower counter value can be ignored. Note that also the counter is authenticated by the signature under private key SK_n, since otherwise an attacker could simply change the counter value to have the replay accepted at the base station. This also prevents an attacker from constructing an item with the highest possible counter value which would result in no message from the legitimate node ever to be accepted again by the base station.

Probably it is not feasible to have nodes maintain counter values of all other nodes. Should this be feasible, nodes will typically have a relatively old counter value. This is particularly the case when nodes have knowledge of the relative position of the base station and items tend to follow the same paths from node to base station. Nodes outside this path will only rarely see an item from that source and hence will typically have stored a highly outdated counter value. An attacker will then likely have sufficient opportunity to mount a DoS attack by using relatively recent items, possibly in combination with the location in the network.

To avoid the spreading of replays through the network nodes would ideally be capable of assessing the freshness of all cache items. A possible solution would be to replace the counter values by time stamps. Items that are either too old or are too far from the future can now be excluded from the cache. This solution has the following disadvantages:

- Sensor nodes have to be equipped with a clock
- Clocks will have to be mutually synchronised throughout the network
- It is difficult to determine when an item is too old. The Gossip communication model derives its robustness from the fact that items reach the base station along different paths. An item following a long path should not be considered a replay too soon and be discarded. On the other hand, when old items continue to be accepted for too long, this could provide enough room for an attacker to flood the network with replays. See also the previous comments on the lifetime limit in Section 5.3.
- As time stamps are authenticated, nodes must be able to verify the signatures of other nodes. A possible solution to this problem is given below.

Alternatively, freshness verification could utilise the notion of rounds used by the Gossip communication model.[1] The advantage of this option that the need for an synchronised clock disappears. Instead of the message sequence numbers suggested above, the number rn of the round in which a new item is inserted into the network by a source node is added to the item: $(n, \{d, rn\}_{SK_n})$. As this number is shared by all nodes in the network, each node is able to assess the freshness. However, it remains a problem to determine exactly when an item is to be considered too old. To initialise the round number (e.g. after first deployment of a node) a node could request its neighbours to send the current round number. The node could take the average of received round numbers to initialise its local round counter. As an attacker's interest

[1]As noted before it is unclear how the synchronisation suggested by the term round is obtained and what security provisions are employed to protect the synchronisation mechanism from adversarial manipulation.

is to have this counter set to as low a value as possible in order to increase the probability that items from the new node are discarded before having reached the base station, outliers should be discarded from the received round numbers.

9.3 Confidentiality

When confidentiality of sensor information is desired the data element d can be encrypted for the base station $BS \in N$. Cache items are now of the form:

$$(n, \{\{d\}_{PK_{BS}}, \mathsf{c}_n\}_{SK_n}) \tag{9.2}$$

9.4 Discussion

So far it has been assumed that nodes are equipped with the public keys of nodes within their communication range. This assumes that sufficient storage is available for these keys at the nodes and there is prior knowledge of the precise distribution of the nodes such that for each node the set of neighbours that will (at any point in time) be in the communication range can be determined prior to sensor node deployment. Hence this assumption is expensive in terms of sensor hardware and excludes the dynamic addition of nodes to the network. Furthermore, we previously saw that it is desirable to verify in an early stage the authenticity and freshness in order to avoid the spreading of false information in the network. Ideally nodes would be able to verify the signatures of all nodes in the network without having to store or obtain for each of these nodes the public key.

Perhaps Identity Based Encryption could be of use here. Then the public key of an arbitrary node n can be directly derived from its identity n and a public parameter.

Alternatively, nodes could first exchange their public key, authenticated by the base station:

$$\begin{array}{lll} (1) & a \rightarrow p & : \{a, PK_a\}_{SK_{BS}} \\ (2) & p \rightarrow a & : \{p, PK_p\}_{SK_{BS}} \end{array} \tag{9.3}$$

When sensor nodes are typically immobile, the required additional communication can be reduced by storing the public keys of neighbours, effectively trading storage for communication. Note that this alternative does not provide the opportunity to verify the authenticity of items originating from nodes other than neighbours.

9.5 Robustness

A captured key pair PK_n, SK_n enables an attacker to complete a protocol run as defined by (9.1). When nodes are not capable of verifying the authenticity of (all) individual cache items, an attacker can compose a set of items of which the identifiers do not occur in the network and with arbitrary data. The intersection of such a set of items with that of a legitimate peer is empty such that a legitimate peer receiving such a set of items has to use U storage places to accommodate the false items. In the unlikely case that the attacker honours the restriction to initiate precisely once per round a connection with a peer and under the assumption that the attacker on average is contacted once per round by an active peer, each round $2U$ storage places of the network are filled with false elements and $2U$ legitimate items are removed from the network as these will not be stored by the attacker. Although the base station will identify these false items, much energy has been committed to deliver these items to the base station and legitimate items have been prematurely removed to make room for the false items.

The attacker is also able to construct items with identifier n and data d of her choice and add a valid signature under private key SK_n. Obviously these items cannot be identified as false by the base station, at least not at the cryptographic level.

When items are encrypted for the base station, the confidentiality is not violated once an attacker learns a sensor's key pair.

9.6 Efficiency

To analyse the efficiency we need to consider implementation options for the abstract encryption and decryption operations introduced above. For a concrete implementation there are various options of which the most obvious are RSA, Rabin and ElGamal. For RSA we choose a system wide low public exponent in order to reduce the cost of encryption and signature verification and to have a public key that is determined by just the modulus. ElGamal requires for encryption two modular exponentiations and the size of the cipher text is twice the size of the modulus. Therefore we reject ElGamal as implementation option. The size of the modulus determines the computational complexity and the amount of data expansion caused by encryption and signature. A modulus length of 1024 bit seems an absolute minimum, for data to be secured until the year 2010 a lowerbound of 1613 bit is advised [24].

9.6.1 Bandwidth

For protocol (9.1) an additional message is required for each exchange. Also nonces and identifiers need to be included. If nc_a, nc_p, p can be represented as an integer smaller than the modulus[2], then $\{nc_a, nc_p, p\}_{PK_a}$ requires the same number of bits as does the modulus. The amount of additional data to be sent is now for each exchange three times the size of the modulus and two signatures in messages (1) and (2).

Authentication at item level will result in data expansion of the cache items. Assuming that (d, c_n) can be represented in half the length of the modulus, the extent of the data expansion is limited by opting for signature with *message recovery* combined with ISO/IEC 9796 formatting [27]. The size of $\{d, \mathsf{c}_n\}_{SK_n}$ is now the size of the modulus. Hence each cache element is subject to a data expansion of at least half the modulus length. Per exchange minimally an additional U modulus lengths have to transmitted.

In case of encryption for the base station according to (9.2), $\{d\}_{PK_{BS}}$ will have size equal to that of the modulus and as a result can no longer be authenticated directly with message recovery. Note that when we define the signature on the plain text: $\{d, \mathsf{c}_n\}_{SK_n}$, the direct use of message recovery remains possible, but the confidentiality is violated in that case as d can be determined from $\{d, \mathsf{c}_n\}_{SK_n}$. We now adapt (9.2) as follows:

$$(n, \{d\}_{PK_{BS}}, \{g(\{d\}_{PK_{BS}}), \mathsf{c}_n\}_{SK_n}) \tag{9.4}$$

for *collision resistant* hash function g. Note that for a typical size of 160 bit of the hash value and 1024 bit for the modulus there is sufficient room to include counter c_n in the signature with message recovery under private key SK_n. Per exchange we now have an overhead of $2U$ modulus lengths.

In summary, for 1024 bit moduli and authentication at item level the amount of additional data to be exchanged will minimally amount to $128U+384+16$ byte. Encryption further increases this by $256U$ byte.

[2]RFC2437 [20] defines RSAES-OAEP to map bitstrings of length maximal the bitlength of the modulus $-16-2$ hash lengths to an integer in the appropriate range.

9.6.2 Storage

As a bare minimum, each node n has to store key SK_n. For encryption for the base station also PK_{BS} has to be stored. Furthermore it is desirable to store for each node within the communication range the public key to avoid having to exchange it repeatedly. When the public keys of the neighbouring nodes cannot be placed on the sensor nodes prior to network deployment, neighbours have to (either once or repeatedly) exchange their public keys authenticated by the base station. This requires the storage of $\{PK_n\}_{SK_{BS}}$, which could be implemented with a RSA or Rabin signature under key SK_{BS} on some hash of PK_n. For 1024 bit moduli 256 byte of storage space is required for the certified public key.

With authentication at item level the cache requires minimally an additional $C/2$ modulus lengths of storage space (see the analysis above). Encrypted cache items further require C modulus lengths of storage space.

To summarise, for 1024 bit moduli with authentication at item level, B the number of neighbours and 4 byte message counters[3] a minimal additional amount of $128B + 64C + 512 + 4$ byte of storage is required. Encryption further increases this by $128C$ byte of storage.

9.6.3 Computation

The protocol requires three times an encryption and decryption operation. An RSA encryption with a 1024 bit key costs 15.97 mJ and a decryption 546.5 mJ [31].[4] Notice that a passive node must execute two expensive decryption operations before the authenticity of the active node can be established. Perhaps this provides an attacker with enough opportunity to mount a DoS attack by exhausting the energy source.

Per exchange two hash values have to be determined over the complete set of items to be exchanged. Remark that, contrary to the RSA operations, these hash computations are also required in protocol (8.1). We argue that the hash computations are negligible compared to the RSA operations. If we assume a cache size $C = 50$, all items are exchanged ($U = C$), identifier n can be represented in 16 bit and we choose 1024 bit moduli, then the size of the set of items to be exchanged is $50(16 + 1024) = 52000$ bit. Determining an HMAC on this data costs 7.5 mJ, just over 1% of the costs of an RSA decryption.

For authentication at item level it is desirable for verification of signatures to be inexpensive compared to generation of the signature. After all, generation of the signature is done only once at the source node when the new item is inserted, while all intermediary nodes from source to base station have to verify the signature. Moreover it is favourable that the amount of energy spent on false items is limited. Although RSA with a small public exponent already has an advantageous balance between the cost for encryption and decryption, a modified Rabin [27, §11.3.4] is preferable as the verification of a signature is limited to just one modular squaring, while the cost for generating the signature is comparable with an RSA decryption. If we take 10 mJ as estimate for the verification of a Rabin signature, the cost for verification of the exchanged items are $20U$ mJ.

Rabin is also an interesting option for the encryption of the data for the base station. Encryption then requires a one time investment by the source node of 10 mJ per item to be inserted, based on the same estimate for the cost of the Rabin public operation.

For each new item to be inserted approximately 0.55 J has to be spent to generate the signature under private key SK_n. Encryption requires another 10 mJ and a negligible amount of energy for determining the hash value $g(\{d\}_{PK_{BS}})$.

[3]With 4 byte message counters a node can generate an item each second for a period of 136 years.

[4]Although [31] neglects to mention it, it seems a small public exponent is assumed, which would explain the difference in costs for decryption and encryption

In summary, for 1024 bit moduli and authentication at item level per exchange $1.7+0.02U$ J is required for the protocol extensions of which 1.6 J is used for RSA decryptions. Per item at least a one time investment of 0.55 J is needed. Note that here we only consider the additional energy required for computation while the additional data to be sent at each exchange (as analysed above) is likely to also require a significant amount of energy.[5] Finally, we remark that we have ignored the generation of nonces in the efficiency analysis.

9.7 Optimisations

As follows from the analysis above, the RSA decryption is by far the most expensive operation. We propose a number of modifications to the protocol of (9.1) with the aim of reducing the energy requirements without affecting the security. We remark that the suggested modifications are in a sense conservative, but that in general subtle changes can introduce vulnerabilities.

9.7.1 Replacing asymmetric encryption in message (3)

From message (3) in (9.1), p can deduce that its nonce nc_p is known to a. The encryption with PK_p ensures that nc_p remains a shared secret between a and p. We can accomplish the same by replacing message (3) with:

$$(3) \quad a \rightarrow p \quad : \quad g(nc_p)$$

for non invertible hash function g. We could also use the previously defined $h : h_{nc_p}(a)$. The choice of a as argument to h is arbitrary, but strictly speaking the non derivability of m from $h_k(m)$ is not a design requirement for a MAC whereas non derivability of k obviously is required. As a precaution m in $h_k(m)$ should not contain confidential information.

As an alternative the active node can simply reveal nonce nc_p in message (3):

$$(3) \quad a \rightarrow p \quad : \quad nc_p$$

Obviously, in this case nonce nc_p is no longer a shared secret between a and p, but p can still conclude from message (3) that its nonce nc_p is known to a.

With this optimisation an expensive decryption by the passive peer and an encryption by the active peer are avoided. The hash computation now required for message (3) (by both a and p) is insignificant compared to the elimination of the asymmetric operation. In case message (3) is the nonce nc_p, no computation is required for message (3). This optimisation also significantly reduces the amount of data to be sent. As previously noted, the size of (3) in (9.1) is the size of the modulus, for the non invertible hash value a length of 80 bit suffices [27]. Lastly, the amount of energy to be spent is divided more equally between active and passive peer and the passive peer is no longer required to execute two expensive RSA decryptions before being able to establish the authenticity of the active peer.

9.7.2 Modification message (2)

We can replace the nonce used as key for the signature on s_p by the nonce nc_a of a:

$$(2) \quad p \rightarrow a \quad : \quad \{nc_a, nc_p, p\}_{PK_a}, s_p, h_{nc_a}(s_p)$$

[5]Sending and receiving 1024 bit of data consumes according to [6] under specific circumstances 21.5 mJ and 14.3 mJ respectively.

The advantage is that the active node is able to perform the inexpensive verification of MAC $h_{nc_a}(s_p)$ before executing the expensive RSA decryption operation. In case the MAC does not pass verification, a can avoid the decryption. One might now be tempted to omit nonce nc_a from the first element of (2) encrypted with PK_a. However, an attacker would then be able to modify message (2) to $\{nc, p\}_{PK_a}, s_p, h_{nc_a}(s_p)$ with nonce nc to be freely chosen by the attacker. Although p will detect this manipulation upon receipt of (3), a will wrongly assume that nc is a shared secret between a and p.

9.7.3 Symmetric key derivation

When nodes are immobile and hence the set of peers is rather static, the property that the nonces in (9.1) are shared secrets between both peers can be exploited. After completion of a protocol run according to (9.1) peers can derive a symmetric key from the exchanged nonces. Let $nc_{m\,n}$ be the nonce of node m used in an interaction with node n according to protocol (9.1). We define the symmetric key $k_{m\,n}$ between nodes m and n as $k_{m\,n} = nc_{m\,n} \oplus nc_{n\,m}$ where $\oplus$ is the bitwise exclusive or operation. At a next exchange the expensive asymmetric operations can be avoided by using the following protocol:

$$\begin{array}{lll} (1) & a \rightarrow p & : & s_a, h_{k_{a\,p}}(a, s_a) \\ (2) & p \rightarrow a & : & s_p, h_{k_{a\,p}}(p, s_p) \end{array} \tag{9.5}$$

This protocol offers authentication of the set of cache items s_n, yet the freshness cannot be established. We solve this by introducing message counters. Let $\mathsf{c}_{n \rightarrow m}$ be the counter of node n counting the number of messages sent to node m since the establishment of shared key $k_{m\,n}$. Further let $\mathsf{c}_{n \leftarrow m}$ be the counter value of the last message from node m accepted by node n. Now we can define the following protocol:

$$\begin{array}{llll} (1) & a \rightarrow p & : & s_a, \mathsf{c}_{a\rightarrow p}, h_{k_{a\,p}}(a, \mathsf{c}_{a\rightarrow p}, s_a) \\ & a & : & \mathsf{c}_{a\rightarrow p} := \mathsf{c}_{a\rightarrow p} + 1 \\ & p & : & \mathsf{c}_{a\rightarrow p} \overset{?}{>} \mathsf{c}_{p\leftarrow a} \\ & & & \mathsf{c}_{p\leftarrow a} := \mathsf{c}_{a\rightarrow p} \\ (2) & p \rightarrow a & : & s_p, \mathsf{c}_{p\rightarrow a}, h_{k_{a\,p}}(p, \mathsf{c}_{p\rightarrow a}, s_p) \\ & p & : & \mathsf{c}_{p\rightarrow a} := \mathsf{c}_{p\rightarrow a} + 1 \\ & a & : & \mathsf{c}_{p\rightarrow a} \overset{?}{>} \mathsf{c}_{a\leftarrow p} \\ & & & \mathsf{c}_{a\leftarrow p} := \mathsf{c}_{p\rightarrow a} \end{array} \tag{9.6}$$

Where $n : x \overset{?}{>} y$ is the verification by n of assertion $x > y$. Whenever $x \leq y$, n will abort the protocol run.

Note that the freshness offered by (9.6) is a weak form of freshness, contrary to the nonces used in (9.1) that offer strong freshness.

The reduction achieved by protocol (9.6) compared to protocol (9.1) is per exchange regarding bandwidth at least 2 modulus lengths minus the length of 2 counter values for messages (1) and (2). Message (3) of (9.1) disappears completely. Compared with the original prototcol of (9.1) this is a reduction of another modulus length, with the optimisation of Section 9.7.1 the reduction is limited to the size of the hash output. Regarding computation, at least two expensive encryption and decryption operations are saved.

The savings with respect to bandwidth and computation do come at the cost of increased storage space on the sensor nodes to accommodate symmetric key $k_{m\,n}$ for each neighbour and two counters per neighbour.

To summarise, with 2 byte counters and 128 bit symmetric keys, at the cost of $20B$ byte additional storage space a reduction per exchange of at least $2(128-2)+10=262$ byte of data to be sent and at least 1.1 J of computations to be performed is achieved with this optimisation.

10 Unique symmetric key

In the context of privacy-friendly Radio Frequency Identification protocols Ohkubo et al, [29] propose the following solution based exclusively on hash functions. The (single) goal of the protocol is identification of a *tag* to a base station and only to that base station. Each tag t is supplied with a secret randomly chosen x_t^0 which is also stored by the base station and associated with a unique identifier. For identification the tag sends at the i-th identification hash $\mathcal{G}(x_t^i)$ to a *reader* where $\mathcal{G}$ is a non invertible hash function. The reader authenticates to the base station after which the base station supplies the reader with the identifier associated with the hash, if it could be found.

After each identification the tag renews its key x_t as follows: $x_t^{i+1} = \mathcal{H}(x_t^i)$ for non invertible hash function $\mathcal{H} \neq \mathcal{G}$. As the sent identification values appear random without knowledge of an element from $\{x_t^i \,|\, i \geq 0\}$, an attacker cannot correlate different identifications of the same tag. Also, based on the non invertibility of $\mathcal{H}$ an attacker is not able, after having captured x_t^j, to match the particular tag with previously learnt identifications $\mathcal{G}(x_t^i), 0 \leq i < j$.

Contrary to the scenario of [29], we are not interested in anonymity of the sensor nodes. Because we do not require anonymity, the associated scalability problem disappears. Instead, we use the pseudo random sequence $\mathcal{G}(x_n^i), i \geq 0$ as a keystream and as noted in [2], x_n^i can be used as a secret session key. As x_n^i is a secret shared between node and base station, it can only be used for authenticity and confidentiality at item level. After all, as previously noted, authentication extensions at set level will never reach the base station just as the particular composition of the set during an exchange between peers will not be known to the base station.

We refer the reader to [4] for an extensive discussion of using key evolution to obtain forward security.

A revised version [26] of the security provisions presented in this chapter has been accepted for the 3rd International Conference on Security in Pervasive Computing.

10.1 Definitions/assumptions

Each node is initialised with an initial key, that will be used as the seed for the hash chain. More precisely, for each node $n \in N$ the base station chooses an initial key x_n^0 uniformly at random: $x_n^0 \in_R \{0,1\}^{\mathsf{k}}$ for security parameter k. The key is securely distributed to the sensor node such that x_n^0 is a shared secret between node n and the base station.

We define $\mathcal{H}$ to be a non-invertible collision resistant hash function with domain restricted to k-bit strings: $\mathcal{H} : \{0,1\}^{\mathsf{k}} \rightarrow \{0,1\}^{\mathsf{k}}$.

10.2 Authentication

We elaborate on the definition of MAC h introduced in Section 8.1 and define $h : \{0,1\}^{\mathsf{k}} \times \{0,1\}^* \rightarrow \{0,1\}^{\mathsf{t}}$ to be a MAC function that on input of a k-bit key x and arbitrary d outputs a t-bit signature $h_x(d)$ on d. We require h to resist adaptive chosen-message attacks.

We redefine the i-th message of node n to include a MAC under ephemeral key x_n^i as follows:

$$(n, d, h_{x_n^i}(d)) \tag{10.1}$$

Informally we argue that under the assumption that x_n^0 is unknown to the attacker and the fact that h offers key non-recoverability, an attacker is unable to learn any key $x_n^j, j \geq 0$ without physically tampering with the sensor node. Therefore an attacker will have a negligible probability of success in creating a valid signature for data other than data previously signed by n. We provide a more thorough security analysis in Section 10.6.

10.3 Key evolution

We define for any $n \in N$ the i-th key x_n^i to be simply the image under $\mathcal{H}$ of the previous key x_n^{i-1}:

$$x_n^i = \mathcal{H}(x_n^{i-1})\,,\, i > 0 \tag{10.2}$$

We assume nodes erase key x_n^{i-1} once the new key x_n^i has been determined and the erasure is such that we can assume that physical inspection cannot reveal erased keys.

10.4 Confidentiality

We elaborate on the definition of encryption function E introduced in Section 8.2 and define $E : \{0,1\}^{\mathsf{k}} \times \{0,1\}^l \rightarrow \{0,1\}^l$ that on input of a k-bit key x and arbitrary plain text d of length l outputs cipher text $E_x(d)$ also of length l. As noted before, in our scenario it is of particular importance that encryption does not result in data expansion since energy costs are reportedly dominated by transmission rather than computation [21, 30, 13]. For the moment we require E to resist adaptive chosen plain text attacks, implementation options and specific security properties will be explored in Section 10.8.

Messages that offer confidentiality in addition to authentication are now defined by:

$$(n, E_{x_n^i}(d), h_{x_n^i}(E_{x_n^i}(d))) \tag{10.3}$$

10.5 Freshness

The key evolution as defined in (10.2) offers implicit weak freshness. Similar to the implicitly embedded message counter in Section 8.3, the base station should maintain for each node the current position in the hash chain. Since our scenario does not exclude the possibility of message loss or delay in message delivery, the base station should anticipate non-sequential message receipt and verify the MAC for a limited number of keys from the key chain.

We introduce an acceptance window of length $2w + 1, w \geq 0$. Let X_n be the element of the hash chain $x_n^i, i \geq 0$ stored by the base station. Initially X_n is set to x_n^0. Upon receipt of a message (n, d, sig) with $n \in N$ the base station searches for a $j, 0 \leq j \leq 2w$ such that $h_{\mathcal{H}^j(X_n)}(d) = sig$. If it finds such a j it accepts d as authentic and fresh originating from node n.

The base station updates the stored element of the hash chain as follows:

$$X_n := \mathcal{H}^{j-w}(X_n) \quad \text{iff} \quad j > w \tag{10.4}$$

Hence, apart from a short initialisation phase where the backward window is empty, $\mathcal{H}^w(X_n)$ is the maximal element of the chain $x_n^i, i \geq 0$ for which a message has been accepted. Once a message signed with an element further down the chain is received, the acceptance window is shifted forwards such that messages with an offset of at most $\pm w$ are accepted as fresh. Obviously the non-invertibility of $\mathcal{H}$ prevents us from storing $\mathcal{H}^j(X_n)$ rather than $\mathcal{H}^{j-w}(X_n)$.

Of course there is a trade-off to be made here, the base station can reserve $2w\mathsf{k}$ additional bits per node to avoid having to compute up to $2w$ elements of the hash chain for each received message (n, d, sig).

We will ignore the handling of duplicate messages here. It is trivial to record for which elements of the hash chain a message has been received such that duplicates can be ignored if so desired.

We remark that there is ample opportunity for refinement of the base station's policy for accepting messages. The base station could, for instance, dynamically change the value of w perhaps even for each node individually. It could use previous communication patterns and the current network traffic to determine an appropriate value. Should the base station detect prolonged disruption of communication it could temporarily increase the length of the forward window or update the stored hash chain elements using previous patterns to avoid loss of synchronisation. These refinements are context dependent and will further be ignored here.

10.6 Verification

10.6.1 Signature forgery

We define a signature forgery attack to be successful if the attacker without knowledge of key x succeeds in constructing some message (n, d, sig) that is accepted by the base station as authentic and fresh originating from node n using key x, when no such signature under key x has been generated bij node n.

We show that the probability of success of such a forgery is negligible in security parameters k and t. In our analysis we consider h to be a black box for which the probability of constructing a valid signature $sig = h_x(d)$ for arbitrary data d and unknown but fixed key x is at most $2^{-\mathsf{t}}$, assuming $\mathsf{t} \leq \mathsf{k}$.

Note that for specific node $n \in N$ the set of acceptable keys is determined by $\{\mathcal{H}^i(X_n) \,|\, 0 \leq i \leq 2w\}$. It now follows easily that the probability of successful forgery is at most $2^{-\mathsf{t}+\log(2w+1)}$.

As expected, the width of the acceptance window influences the probability of successful forgery. When determining an appropriate value for t also an upper bound on the value of w should be established such that the probability of successful forgery as determined above is sufficiently low. We refer to Section 10.8 for practical values for both t and w.

A notable difference with [29] is that an attacker is forced to choose a particular node $n \in N$ whereas [29] excludes node identifiers in order to preserve anonymity. This increases the probability of successful forgery by a factor $|N|$.

10.6.2 Confidentiality violation

We define oracle $\mathcal{O}_C$ that on input of arbitrary plaintext d, node $n \in N$, and iteration number $i \geq 0$ outputs the cipher text under key x_n^i:

$$\mathcal{O}_C(d, n, i) = E_{x_n^i}(d) \tag{10.5}$$

We consider an attacker that has queried $\mathcal{O}_C$ at a number of inputs (d_j, n_j, i_j) that is polynomial in security parameter k but not at input (d, n, i). We define an attack on the confidentiality of $E_{x_n^i}(d)$ to be successful if the attacker is able to determine any information on d other than its length $|d|$.

First, we show by contradiction that the key that is input to E is unique. Assume $x_n^i = x_{n'}^j$ with $i < j$. Note that we do not require the keys to be element of the same hash chain. Then we have established x_n^{i-1} and $x_{n'}^{j-1}$ as colliding inputs to $\mathcal{H}$, contradicting its collision resistance.

If we consider E to be a Random Oracle and the input to E is unique, we have that $E_{x_n^i}(d)$ cannot be distinguished from a truly random $r \in_R \{0,1\}^{|d|}$ without knowledge of key x_n^i. Hence the probability distribution is uniform over the plain text space $\{0,1\}^{|d|}$.

10.6.3 Forward security

Suppose an attacker is able to physically extract or otherwise obtain at some point key x_n^j. We argue that knowledge of this key does not lead to a compromise of any previous messages.

We define oracle $\mathcal{O}_F$ that provides for given $n \in N$ and $i \geq 0$ key x_n^i:

$$\mathcal{O}_F(n, i) = x_n^i \tag{10.6}$$

We consider an attacker that has queried $\mathcal{O}_F$ at input (n, j) but not at any of the inputs (n, i), $0 \leq i < j$. We show that knowledge of the obtained key x_n^j does not help the attacker in forging signatures or deciphering messages under any key $x_n^i, 0 \leq i < j$.

Under the assumed non-invertibility of $\mathcal{H}$ key $x_n^{j-1} = \mathcal{H}^{-1}(x_n^j)$ cannot be determined from x_n^j in less than 2^{k} operations. Obviously we then have for any $x_n^i, 0 \leq i < j$ that it cannot be determined from x_n^j in less than 2^{k} operations since any such x_n^i would lead to $x_n^{j-1} = \mathcal{H}^{j-1-i}(x_n^i)$.

10.7 Robustness

Since each node has a unique key, an attacker is limited to constructing items on behalf of node n after having captured key x_n^j. Moreover, as shown above, capturing this key does not enable an attacker to decrypt items previously sent by node n and encrypted with x_n^i for some $0 \leq i < j$. Also the generation of valid signatures is limited to keys from the chain starting at x_n^j.

As nodes are not capable of establishing the authenticity of items from other nodes, an attacker is able to spread false items or replay previously learnt authentic items. Although the base station will detect the forged items, the attacker is able to flood the network with false information.

10.8 Implementation

So far our discussion has been on an abstract level and mainly focused on the security properties. In order to analyse the resource requirements we will now consider implementation options for functions $\mathcal{H}$, h and E as defined earlier.

10.8.1 Hash function

Although recently it has been shown [37] that finding collisions for SHA-1 requires far less effort than the ideal strength of 2^{80} operations, we propose to take SHA-1 to implement $\mathcal{H}$. We note that the non-invertibility of SHA-1 is not affected and we have only used its collision resistance property in support of our confidentiality proof of Section 10.6.2. Even there, the inputs to $\mathcal{H}$ cannot be influenced by the attacker which makes the attack inapplicable.

The code size for a software implementation on various embedded processors is around 2000 byte based on data from [34] which is less than half the amount required for MD5. Presumably, in a hardware implementation several optimisations can be applied to reduce the required chip area, however the SHA-1 implementation is likely to remain significant compared to the simple sensor node logic required for e.g. a smoke detecting sensor. It is our objective to reuse SHA-1 for functions h and E in order not to further increase code size/chip area.

10.8.2 MAC function

We choose to use HMAC-SHA1-t as implementation of MAC function h. Here parameter t is used as described in RFC2104 [22] to denote that only the first t output bits are used as the MAC.

We remark that the HMAC construction involves basically two applications of the hash function and some simple padding operations. The added complexity in terms of code size/chip area is therefore limited.

10.8.3 Encryption function

Similar to [29] we define $\mathcal{G}$ to be a non-invertible collision resistant hash function independent from $\mathcal{H}$ with domain restricted to k-bit strings: $\mathcal{G} : \{0,1\}^{\mathsf{k}} \rightarrow \{0,1\}^{\mathsf{k}}$. We use the hash values generated by $\mathcal{G}$ as a keystream. E can now be defined in terms of $\mathcal{G}$:

$$E_{x_n^i}(d) = d \oplus \mathcal{G}(x_n^i) \tag{10.7}$$

Where, as before, $\oplus$ denotes the bitwise exclusive-or operation that is understood to discard the remaining suffix of $\mathcal{G}(x_n^i)$ of length $\mathsf{k} - |d|$ bits and operate on the first $|d|$ bits of $\mathcal{G}(x_n^i)$ only.

As a first observation note that the length of the plain text is now limited to k bits. We remark that for the scenarios we consider the value of k (to be determined shortly) is more than sufficient to accommodate typical message sizes. We could generate a longer keystream, for instance, by successively applying $\mathcal{G}$ to simple variations of x_n^i, e.g. construct a keystream $\mathcal{G}(x_n^i \oplus 0)|\mathcal{G}(x_n^i \oplus 1)|\ldots|\mathcal{G}(x_n^i \oplus l-1)$ and define items as

$$(n, d\oplus\mathcal{G}(x_n^i\oplus 0)|\mathcal{G}(x_n^i\oplus 1)|\ldots|\mathcal{G}(x_n^i\oplus l-1), h_{x_n^i}(d\oplus\mathcal{G}(x_n^i\oplus 0)|\mathcal{G}(x_n^i\oplus 1)|\ldots|\mathcal{G}(x_n^i\oplus l-1))) \tag{10.8}$$

where $|$ is the concatenation operator and l is determined by $\lceil |d|/\mathsf{k} \rceil$.

Earlier we showed uniqueness of key x_n^i. If we consider $\mathcal{G}$ to be a Random Oracle, we can view $\mathcal{G}(x_n^i)$ as a One Time Pad offering perfect secrecy. Non-invertibility of $\mathcal{G}$ also guarantees that a known plaintext attack in which an attacker obtains $\mathcal{G}(x_n^i)$ does not lead to a compromise of x_n^i.

Notice that the construction we suggest here relies rather heavily on the assumptions that we make on the properties of $\mathcal{G}$. Once we choose a concrete hash function to implement $\mathcal{G}$ that has some predictable bias, this bias would directly reveal (partial) information on the plain text.

We have defined $\mathcal{G}$ to be independent from $\mathcal{H}$. In the original scenario of [29] this is essential as the output of $\mathcal{G}$ is sent in the clear such that in case $\mathcal{G} = \mathcal{H}$, the sent identification value $\mathcal{G}(x_t^i)$

is equal to the new key $x_t^{i+1} = \mathcal{H}(x_t^i)$. But also in our scenario, an attacker with knowledge of some plain text is able to determine (some prefix of) keystream $\mathcal{G}(x_n^i)$ resulting again in the compromise of subsequent keys if $\mathcal{G} = \mathcal{H}$. However, from the point of chip area/code size it were desirable if $\mathcal{G}$ and $\mathcal{H}$ could use the same concrete hash function. Therefore, we define $\mathcal{G}(x_n^i) = \mathcal{H}(x_n^i \oplus msk)$ for fixed and public $msk \neq 0$ of length k bit. Knowledge of $\mathcal{G}(x_n^i)$ now does not allow an attacker to determine the subsequent key $\mathcal{H}(x_n^i)$.

10.8.4 Choice of security parameter sizes

Throughout we have used k to denote the size of the key and t to denote the size of the signature. As mentioned before the amount of message expansion is of particular importance in our scenario.

We adopt a keylength of $\mathsf{k} = 160$ bit as suggested for a collision resistant hash function in [27]. Choosing a lower value does not seem to offer significant savings as the key is expanded to full block length when used as input to SHA-1.

The suggested lowerbound of $\mathsf{t} = 64$ bit for the size of the MAC could add significant overhead for typical message sizes. As mentioned in [27] for specific applications this value could be reduced if the probability of successful forgery $2^{-\mathsf{t}+\log(2w+1)}$ is acceptable, taking into account the total number of forgery attempts an attacker is able to perform during the lifetime of the sensor network and the implications of a single successful forgery.

10.9 Efficiency

For the sake of concreteness, let us assume that we would like to accept messages with an offset of at most $w = 8$, resulting in an acceptance window of length 17. Assuming we like to maintain a security level of approximately 64 bit, we would have to set $\mathsf{t} = 68$.

10.9.1 Bandwidth

The signature appended to each item results for 68 bit signatures in an amount of additional data to be sent of $17U$ byte per exchange. Encryption does not cause data expansion and hence has no performance penalty with respect to bandwidth consumption.

10.9.2 Storage

The current element x_n^i of the hash chain has to be stored by node n. With $\mathsf{k} = 160$ this will require 20 byte of storage space.

10.9.3 Computation

The energy required to establish a new key as per (10.2) is about 15 μJ for $\mathsf{k} = 160$ according to experimental results from [31].

The amount of energy required for the one-time generation of a signature for a new item is estimated by [31] to be around 1 μJ/byte.

Encryption requires for each data block of 160 bit (or part thereof) an evaluation of hash function $\mathcal{G}$. Hence encryption consumes $\lceil |d|/160 \rceil \cdot 15, 2\mu$J.

As we chose SHA-1 as hash function for the HMAC construction, there is a great amount of code reusage for key evolution, authentication and encryption.

11 Evaluation and comparison

Table 11.1 summarises the efficiency characteristics of the scenarios presented in the preceding three chapters.

	Bandwidth (B)	Storage (B)	Energy per exchange (J)	Energy per item (J)
(8.1)	$16 + 8U$	$4C + 12$	$1.16\mu \cdot 4U \cdot \lvert(n, d, \mathsf{c})\rvert$	0
(8.2)	16	12	$1.16\mu \cdot 4U \cdot \lvert(n, E_{k,\mathsf{c}}(d))\rvert$	1-2 $\mu \cdot \lvert d\rvert$
(9.1)	$128U + 400$	$64C + 128B + 516$	$1.7 + 0.02U$	0.55
(9.4)	$384U + 400$	$192C + 128B + 516$	$1.7 + 0.02U$	0.56
(9.6)[1]	380	$-24B$	1.7 [2]	0
(10.1)	$17U$	20	0	$15.2\mu + 1.16\mu \cdot \lvert d\rvert$
(10.8)[3]	0	0	0	$\left\lceil \frac{\lvert d\rvert}{20} \right\rceil \cdot 15.2\ \mu$

Table 11.1: Summary performance characteristics. Here $|d|$ denotes the length in byte of d, all solutions have been taken to include freshness at item level and counters have been taken to be 4 byte.

We evaluate for two extreme values for the size of cache items the consequences with respect to the performance characteristics. We consider items for which the length of the data element is 2 byte and items with a 1 kB data element. In the latter case, the assumption made earlier that d, c_n can be represented in half the length of the modulus is no longer valid. Hence the direct use of signature with message recovery is no longer possible. Analogous to the modification from (9.2) to (9.4) we now define cache items to be tuples of the form

$$(n, d, \{g(d), \mathsf{c}_n\}_{SK_n}) \tag{11.1}$$

The amount of additional data to be sent is in this case the length of the modulus and additional energy is required to determine hash $g(d)$.

The size of d also poses a problem for the direct asymmetric encryption of d for the base station: $\{d\}_{PK_{BS}}$. We use the common hybrid solution to encrypt d with a random symmetric key k and encrypt k asymmetrically for the base station:

$$(n, \{k\}_{PK_{BS}}, E_k(d), \{g(\{k\}_{PK_{BS}}, E_k(d)), \mathsf{c}_n\}_{SK_n}) \tag{11.2}$$

[1]Reported are the savings relative to (9.1) and (9.2). Note that the savings can only be achieved after having established a shared key by an (expensive) execution of (9.1) or (9.2), or the somewhat less expensive version according to the optimisation of Section 9.7.1.

[2]With the disappearance of the expensive asymmetric encryption, computing the MAC on the complete set of exchanged items is possibly no longer negligible for large items. This computation amounts to approximately $1.16\mu \cdot (|s_a| + |s_p|)$ J.

[3]Reported are the additional costs relative to (10.1).

	Bandwidth (B)	Storage (B)	Energy per exchange (J)	Energy per item (J)
(8.1)	$16 + 8U$	$4C + 12$	37.12 μU	0
(8.2)	16	12	18.56 μU	24 μ
(9.1)	$128U + 400$	$64C + 128B + 516$	$1.7 + 0.02U$	0.55
(9.4)	$384U + 400$	$192C + 128B + 516$	$1.7 + 0.02U$	0.56
(9.6)^1	380	$-24B$	1.7	0
(10.1)	$17U$	20	0	17.5 μ
(10.8)	$17U$	20	0	32.7 μ

Table 11.2: Summary efficiency characteristics for $|d| = 2$ byte.

	Bandwidth (B)	Storage (B)	Energy per exchange (J)	Energy per item (J)
(8.1)	$16 + 8U$	$4C + 12$	4.78 mU	0
(8.2)	16	12	4.76 mU	1.5 m
(9.1)/(11.1)	$256U + 400$	$128C + 128B + 516$	$1.7 + 0.028U$	0.55
(11.2)	$512U + 400$	$256C + 128B + 516$	$1.7 + 0.029U$	0.56
(9.6)[a]	$256U + 20$	$128C + 152B + 516$	$0.028U$	0.55
(9.6)[b]	$512U + 20$	$256C + 152B + 516$	$0.029U$	0,56
(10.1)	$17U$	20	0	1.2 m
(10.8)	$17U$	20	0	2.0 m

Table 11.3: Summary efficiency characteristics for $|d| = 1$ kB.

[a]Reported are the values for the optimisation as in (9.6) combined with authentication at item level as in (11.1).

[b]Reported are the values for the optimisation as in (9.6) combined with authentication and confidentiality at item level as in (11.2).

This adds 2 modulus lengths to the size of cache items and requires the computation of $E_k(d)$ and $g(\{k\}_{PK_{BS}}, E_k(d))$.

In Tables 11.2 and 11.3, SHA-1 was taken for g. Furthermore it has been assumed that node identifiers can be represented in 16 bit, message counters require 4 byte and symmetric encryption does not result in data expansion. The fact that at least a block length has to be computed for symmetric encryption has been taken into account. Note that also the computation of an HMAC involves a certain amount of fixed costs and hence the estimate from [31] of 1.16 μJ/B is probably not entirely accurate for very small messages.

12 Conclusions

With regard to the choice of a Gossip as the communication mechanism in a wireless sensor network, we have identified a mismatch between its inherent highly redundant nature and the desire to minimise the energy requirement for wireless sensor nodes. Nonetheless we have maintained minimal resource requirements as our guiding principle in the construction of the three different solutions we presented.

The simplest conceivable solution (beyond the trivial one) was presented in Chapter 8 and used a single globally shared symmetric key. Although favourable with respect to resource requirements, the lack of robustness against node capture attacks was identified as its main disadvantage. We can use this solution as an approximate lowerbound of what is required to secure epidemic communication in wireless sensor networks.

A natural second option was presented in Chapter 9 and used asymmetric keys for each node. The robustness is improved significantly over that of the single network key. Several optimisations were suggested in an attempt to reduce its significant computational and bandwidth requirements. Even with all optimisations applied, the resource requirements remain significant. Especially for small messages, the data expansion resulting from asymmetric signature or encryption introduces a quite considerable overhead.

Of particular interest is the solution presented in Chapter 10 using a continuously evolving unique symmetric key per node. Its distinguishing property is high robustness against node captures and in particular forward security of information. However, security is limited to node to base station communication and no in-network authentication is possible. The associated resource requirements are modest.

The resource requirements of the three solutions were compared in the previous chapter. The comparison was elaborated for two different message sizes. We note, however, that the security properties of the solutions differ to such an extent that a careful analysis should be performed in which context dependent information is used to assess the feasibility of attack strategies, the sensitivity of information, the damage caused by a successful attack and so forth. This analysis would help to determine if the use of a more expensive option can be justified. Also note that numerous context specific network parameters greatly influence the required resources. The number of exchanged items, the size of the cache, the typical size of the data element of an item, and the number of neighbouring sensor nodes all determine to various extents the overall resource requirements. Given these parameters, Table 11.1 can be used to select an appropriate solution.

12.1 Open problems and future research

We include here a list of remaining open problems and suggestions for further research. We identify several security related questions and questions relating to the Gossip communication model. As previously mentioned, there is a considerable mismatch between the inherent high redundancy of the Gossip communication model and the desire to minimise resource usage. To bridge this gap, several optimisations have been suggested in Section 5.3, some of which made use of security extensions.

As the Gossip model is limited to the specification of local interactions and the macro level behaviour of the network is observed through simulations, it has proven to be difficult to analyse the impact of modifications to the Gossip exchanges. We would like to assess in relative terms the reduction achieved by an optimisation as well as the effect on other aspects such as the robustness. For instance, it is easy to make the qualitative observation that a reduction is possible if the base station does not initiate exchanges and does not return any items to its peers. Obviously the robustness is not affected in this case. It is, however, difficult to answer the quantitative question whether this would constitute a significant reduction in the overall energy consumption of the network. Further simulations would help to answer these questions.

Most notable among the security related open problems is the mitigation of denial of service attacks. Even if nodes are able to verify the authenticity of information from other nodes, an attacker is able to flood the network with replays of previously observed authentic messages. We discuss here briefly a sketch of a possible solution that has not been investigated further yet.

The base station could generate a hash chain of which the length determines the number of exchanges a node is able to initiate. An authentic commitment to the chain would have to be distributed to neighbouring nodes and the seed of the chain should be privately communicated to the node. A node then has to pay for each exchange by releasing a preimage of the chain. A neighbour verifies if in fact the released value is a preimage of the value stored for the particular node and if so, replaces the stored value with the preimage. The possibilities to insert messages are now limited by the length of the hash chain and the number of neighbours (we have to assume a malicious node is able to spend each hash chain element once at every neighbour). In order to prevent double spending, the commitment should be tied to a particular location in the network and to a particular time. This could be achieved for instance by a Rabin signature on the last element of the chain, the identities of neighbouring nodes and the round number. Alternatively, the shared secret state as in Chapter 10 can be used to construct a MAC on the last hash chain element and the round number for each of the neighbours. See Appendix A for a mechanism to distribute secure round numbers. Confidential distribution of the chain's seed could use the secret shared state of Chapter 10 as well. The identities of the neighbouring nodes would have to be included in the message to the base station requesting the generation of a hash chain. We remark that the overhead introduced is considerable. Nodes have to maintain hash chain elements for each neighbour and have to perform a verification for each exchange. Additional communication to distribute authenticated commitments and confidential seeds and a communication path from base station to node is now required.

As can be observed from the simulation results of [12], knowledge of the relative position of the base station used in the selection of the peer reduces the time required to deliver an item to the base station. However, [12] does not provide any hints how nodes could learn the relative position of the base station. Hence, the interesting question is how this knowledge can be inexpensively and securely provided to the nodes. Appendix A presents a possible solution to this problem.

We mentioned before that it is unclear how the (global) synchronisation suggested by the term "round" is obtained. We also noted the vulnerability resulting from an insecure synchronisation mechanism. Again, this leads to the question how to obtain inexpensive and secure global synchronisation. We refer to Appendix A for a discussion of a possible synchronisation mechanism.

In Section 5.3 we proposed an authenticated removal command from base station to neighbouring nodes. Simulation could show the effectiveness of this removal mechanism and determine whether the reduction is sufficient to justify the investment in the mechanism (e.g. verification of the command's authenticity by the nodes). A relevant question in this regard might be how long a message remains in the network after first delivery to the base station.

A related question is whether other removal mechanisms can be applied. And, more generally, what other reductions in redundancy are possible.

Of great practical interest is the latency introduced by the Gossip communication model. Consider for instance concrete applications where a sensor network is used to detect fires or burglars. Relevant questions are the maximum number of rounds required to traverse the network and the maximum round interval.

The optimal solution would be to remove all redundancy and deliver each item precisely once via the optimal route from source to base station. Evidently the desirable property of robustness would be lost and a presumably expensive route determination mechanism would have to be implemented. However, comparing the highly redundant Gossip communication with this hypothetical optimal solution would establish an (admittedly loose) upperbound to the investment in a removal mechanism or a mechanism to determine the relative position of the base station. More specifically, it would be interesting to see a comparative analysis of the reduction in redundancy between the directed (i.e. knowledge of the relative position of the base station is used in the selection of the peer) and undirected shuffle protocol.

Appendix A: Inexpensive authenticated round numbers

We present here a mechanism that introduces loose synchronisation in the network by means of authenticated round numbers. We distinguish three different uses for the construction we suggest. First, the Gossip exchanges are governed by rounds, the implementation of which is left unspecified in [12]. A loosely synchronised global notion of the number of the round can be used to achieve the division of time in intervals as required in the Gossip model.[1] Second, the round numbers could be used as a basic building block to establish freshness. This would be particularly useful in scenarios where in-network verification of authenticated round numbers is possible such that the spread of replays through the network can be avoided, see also Section 9.2. Recall, however, that in a Gossip context it is difficult to assess the specific age limit from where an item should be considered a malicious replay. Third, the construction we suggest here offers the discovery of the relative position of the base station as a bonus. This enables the use of the more efficient "Directed shuffle" protocol of [12].

The mechanism has only modest resource requirements. The design is based around a hash function which makes it particularly useful in combination with the scheme of Chapter 10.

A.1 Construction

We choose a random secret y_0 and generate a hash chain $y_i, 0 \leq i \leq l$ such that $y_i = g(y_{i-1}, l-i)$, for $1 \leq i \leq l$ and non-invertible hash function g. The last element y_l is the commitment to the chain y and is placed on each sensor over an authenticated channel, for instance during initialisation. To announce round number rn the base station releases the tuple y_{l-rn-1}, rn. Nodes should store the last received element of the hash chain and verify on receipt of the new value if $g(y_{l-rn-1}, rn) \overset{?}{=} y_{l-rn}$. By the non-invertability of g and the secrecy of y_0, only the base station could have released y_{l-rn-1} which provides authentication for the round number. It is not strictly required to include the round number in the message from the base station as the round number is implicitly determined by the position in the hash chain of the released y_{l-rn-1}. Including the round number enables nodes to disregard outdated round number messages such that the evaluation of the hash function can be avoided. Similarly, (forged) round number messages that have too high a round number compared with the current number can be discarded. Furthermore, based on the amount of available energy or other criteria a node could adopt the round number without verifying that it belongs to the hash chain. Also, nodes can be deployed without commitment to the hash chain.

Inclusion of the round number in the computation of the hash value helps to mitigate the

[1]Note that Gossip does not strictly require a round *number*, but merely a division of time in intervals.

risk of collisions in the hash chain. Suppose we exclude the round number from the input to the hash function. As the hash function is deterministic with a finite output domain, the sequence of hash outputs is guaranteed to cycle at some point. In other words, there is a non-zero probability that some element $y_j = g(y_{j-1})$ from the chain of elements has a second preimage $y_{j+l_c-1} \neq y_{j-1}$ where l_c is the length of the cycle in y. This fact could theoretically be exploited by an attacker by blocking the message in which the base station releases hash chain element y_{j-1} and replacing it with hash chain element y_{j+l_c-1}. Once nodes have verified y_{j+l_c-1} to be a preimage of the last seen element y_j and have updated their stored element of the chain to become y_{j+l_c-1}, subsequent legitimate round number messages from the base station will be identified as false by the nodes that received y_{j+l_c}. Moreover, the attacker has gained the ability to determine the round interval. She could either not release any further round number messages which would at best disrupt synchronisation in the network and at worst prevent any further communication in case round number messages trigger peer exchanges. Alternatively, she could release preimages $y_{j+l_c-1}, y_{j+l_c-2}, \ldots$ at increased pace possibly resulting in a DoS attack through energy deprivation.[2]

If we include the number of the round as input to g and we assume the round number does not cycle, cycles in y occur with negligible probability as a repeated occurrence of some hash output is coupled with a different round number the second time it is input to g resulting with high probability in a different output. This argument obviously extends to the case where the latter output is again identical; the probability diminishes extremely fast for increasing length of the sequence of identical hash outputs. Even if the attacker were able to determine some $y_{j-1+l_c} \neq y_{j-1}$ with $j + l_c \leq l$ such that $y_j = y_{j+l_c}$, she would be unable to substitute $y_{j-1+l_c}, l - j - l_c$ for $y_{j-1}, l - j$, if nodes perform the simple verification of increasing round numbers. Inclusion of the round number in the input to g thwarts the precomputation attack mentioned in the footnote of the previous paragraph.

A.2 Distribution

Thusfar, at a conceptual level, we have only considered unidirectional communication from node to base station. Conversely, the distribution of round numbers requires a communication path from base station to node. Perhaps the most natural option would be for the base station to generate Gossip items of its own, include these in return messages to its neighbours (and perhaps even disregard the suggested optimisation of Section 5.3 and initiate exchanges with neighbours) and have the nodes inspect incoming messages for items from the base station. However, for reasons outlined below, we propose an alternative distribution mechanism that offers the additional advantage that the relative position of the base station can be determined at almost no additional cost.

A.2.1 Distribution through Gossip

In the Gossip model, the initiation of exchanges is conditioned by the transition to a next round. If round numbers were to be distributed by means of peer exchanges a circular dependency (i.e. a deadlock) could be introduced as the initiation depends on the receipt of the round number and delivery of the round number depends on an exchange being initiated.

[2]We remark that in case the attacker is not interested in releasing round number messages after side tracking the nodes, she does not depend on the hash chain being part of a cycle, but merely has to find a second preimage for one of the elements in the chain. However, she would have to find this preimage before the base station releases the next round number. Hence, for any reasonable round interval and hash output length it is entirely impractical to find such a preimage in real time. However, with precomputed hash values a birthday attack might be feasible.

Unavoidably, synchronisation in an absolute sense cannot be achieved. Due to propagation delays, typically in the direct neighbourhood of the base station the round number will be higher than at the boundary of the network. Distributing the round number messages through Gossip exchanges is suboptimal if the intent is to minimise this synchronisation difference. Probably this difference would not pose a problem if round numbers are merely used as trigger for peer exchanges. However, if the round number is used to verify freshness, it is desirable to minimise the synchronisation difference.

A.2.2 Distribution through broadcasts

We propose a round number distribution mechanism that achieves rapid delivery of round numbers to all nodes and requires just a single message transmission by each node. We utilise the broadcast property of the wireless medium. The base station broadcasts a message announcing the new round number as described above. Upon receipt of such a message a node verifies its authenticity and if it proves to be authentic the node itself rebroadcasts the message. Subsequent copies of the round number message received as a result of broadcasts from other nodes are ignored. Hence, the round number message floods quickly through the network. Moreover, as the message is broadcast at most once by each node, the message also disappears quickly from the network, unlike the case where the round number would be included in a Gossip item that would linger in the network long after it has reached all nodes.

The mechanism is robust against occasional message loss or corruption as a subsequent copy broadcast by another node will simply be regarded as the first received round number.

If a node first receives the round number message from a particular node, that implies that the fastest route to the base station is via that node. Similarly, the second fastest route is via the node from which the second copy of the round number message was received. This information could be used in the peer selection function such that when initiating an exchange, a higher probability of selection is assigned to nodes closer to the base station. Assuming, as was done before in Section 8.3, that the node identifiers are used to designate sender and receiver in peer exchanges, the identifier should be included in the broadcast messages such that their origin can be determined. Ideally we would like to have origin authentication of the broadcast messages if the origin is used to introduce bias in the peer selection function as we suggest here. This can only be achieved if nodes are able to verify each other's signatures. If the origin cannot be verified, an attacker is able to divert messages from the optimal path by manipulating the source identifier in broadcast messages. The risk of origin forgery of broadcast message should be analysed carefully and too strong a bias in the peer selection function should be avoided.

We remark that in this scenario attacks using *wormholes* [36] are applicable.

A.3 Evaluation

Variations of the distribution scheme are possible where there is only a periodic broadcast of the round number (e.g. every 10 rounds) and nodes are assumed to be able to remain sufficiently synchronised for the number of rounds between broadcasts. This would be particularly useful if the round time is short and frequent broadcasts consume too much energy.

The size of the broadcast message is modest; if birthday attack are inapplicable, the size of the hash output can be limited to about 8 bytes for practical security [27] and 3 or 4 byte for the round number is more than sufficient for any reasonable scenario. Verification of the round number requires a single evaluation of hash function g, assuming at least one copy of the previous round number was received. Otherwise, a few evaluations would be required to arrive at the stored last received hash chain element. In any case, there is a fixed cost of one evaluation of the hash function on a small size input per round of the network. A few bytes of storage should be reserved for the last received element of the chain. These costs should be compared with the efficiency gain obtained as a result of knowing the relative position of the base station.

Should the hash chain approach exhaustion (i.e. should the round number approach the hash chain's length l), the base station could release something of the form $h_{y_0}(y'_l)$ as authenticated commitment to the new hash chain y'. When subsequently y_0, y'_l is released, nodes can verify the authenticity of y'_l and the new chain can be used. It is now crucial to release the commitment to the new chain timely such that it can be excluded that it was constructed by an attacker after publication of y_0 by the base station. This is similar to the delayed release of authenticators in μTESLA [30].

Bibliography

[1] IEEE Standard 802.15.4-2003: Wireless medium access control and physical layer specifications for low-rate wireless personal area networks, May 2003.

[2] Avoine, G., and Oechslin, P. A scalable and provably secure hash based RFID protocol. In *The 2nd IEEE International Workshop on Pervasive Computing and Communication Security – PerSec 2005 (To appear)* (Kauai Island, Hawaii, USA, March 2005), IEEE, IEEE Computer Society Press.

[3] Basagni, S., Herrin, K., Bruschi, D., and Rosti, E. Secure pebblenets. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing* (New York, NY, USA, 2001), ACM Press, pp. 156–163.

[4] Bellare, M., and Yee, B. Forward-security in private-key cryptography. Cryptology ePrint Archive, Report 2001/035, 2001.

[5] Buttyán, L., and Vajda, I. Towards provable security for ad hoc routing protocols. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks* (New York, NY, USA, 2004), ACM Press, pp. 94–105.

[6] Carman, D., Kruus, P., and B.J.Matt. Constraints and approaches for distributed sensor network security. Tech. Rep. 00-010, NAI Labs, June 2000.

[7] Chandrakasan, A., Amirtharajah, R., Cho, S., Goodman, J., Konduri, G., Kulik, J., Rabiner, W., and Wang, A. Design considerations for distributed microsensor systems, 1999.

[8] Deb, B., Bhatnagar, S., and Nath, B. Information assurance in sensor networks. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications* (New York, NY, USA, 2003), ACM Press, pp. 160–168.

[9] Eschenauer, L., and Gligor, V. D. A key-management scheme for distributed sensor networks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security* (New York, NY, USA, 2002), ACM Press, pp. 41–47.

[10] Ganeriwal, S., and Srivastava, M. B. Reputation-based framework for high integrity sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks* (New York, NY, USA, 2004), ACM Press, pp. 66–77.

[11] Gaubatz, G., Kaps, J.-P., and Sunar, B. Public key cryptography in sensor networks - revisited. In *ESAS* (2004), pp. 2–18.

[12] Gavidia, D., Voulgaris, S., and van Steen, M. Epidemic-style monitoring in large-scale sensor networks. Tech. Rep. IR-CS-012.05, Vrije Universiteit Amsterdam, March 2005. `http://www.cs.vu.nl/pub/papers/globe/IR-CS-012.05.pdf`.

[13] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. System architecture directions for networked sensors. In *ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems* (New York, NY, USA, 2000), ACM Press, pp. 93–104.

[14] Hjorth, T. Supporting privacy in RFID systems. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2004. Supervised by Assoc. Prof. Christian D. Jensen, and Prof. Jan Madsen.

[15] Hu, F., and Sharma, N. Security considerations in ad hoc sensor networks. *Ad Hoc Networks 3*, 1 (2005), 69–89.

[16] Itkis, G. Cryptographic tamper evidence. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security* (New York, NY, USA, 2003), ACM Press, pp. 355–364.

[17] Jamshaid, K., and Schwiebert, L. Seken (secure and efficient key exchange for sensor networks). In *Performance, Computing, and Communications, 2004 IEEE International Conference on* (2004), pp. 415–422.

[18] Jelasity, M., and van Steen, M. Large-scale newscast computing on the Internet, Oct. 2002.

[19] Jones, K., Wadaa, A., Olariu, S., Wilson, L., and Eltoweissy, M. Towards a new paradigm for securing wireless sensor networks. In *NSPW '03: Proceedings of the 2003 workshop on New security paradigms* (New York, NY, USA, 2003), ACM Press, pp. 115–121.

[20] Kaliski, B., and Staddon, J. RFC 2437: PKCS #1: RSA cryptography specifications version 2.0, Oct. 1998. Obsoletes RFC2313. Status: INFORMATIONAL.

[21] Karlof, C., Sastry, N., and Wagner, D. Tinysec: a link layer security architecture for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems* (New York, NY, USA, 2004), ACM Press, pp. 162–175.

[22] Krawczyk, H., Bellare, M., and Canetti, R. HMAC: Keyed-Hashing for Message Authentication, Feb. 1997. RFC 2104.

[23] Law, Y. W., Dulman, S., Etalle, S., and Havinga, P. Assessing security-critical energy-efficient sensor networks. Tech. Rep. TR-CTIT-02-18, University of Twente, The Netherlands, July 2002. http://purl.org/utwente//38381.

[24] Lenstra, A. K., and Verheul, E. R. Selecting cryptographic key sizes. *Journal of Cryptology: the journal of the International Association for Cryptologic Research 14*, 4 (2001), 255–293.

[25] Liu, D., Ning, P., and Li, R. Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inf. Syst. Secur. 8*, 1 (2005), 41–77.

[26] Mauw, S., van Vessem, I., and Bos, B. Forward secure communication in wireless sensor networks. In *SPC '06: Proceedings of the 3rd International Conference on Security in Pervasive Computing* (York, United Kingdom), Lecture Notes in Computer Science. To appear.

[27] Menezes, A. J., Vanstone, S. A., and Oorschot, P. C. V. *Handbook of Applied Cryptography.* CRC Press, Inc., Boca Raton, FL, USA, 1996.

[28] Messerges, T. S., Cukier, J., Kevenaar, T. A. M., Puhl, L., Struik, R., and Callaway, E. A security design for a general purpose, self-organizing, multihop ad hoc wireless network. In *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks* (New York, NY, USA, 2003), ACM Press, pp. 1–11.

[29] Ohkubo, M., Suzuki, K., and Kinoshita, S. Cryptographic approach to "privacy-friendly" tags. In *RFID Privacy Workshop* (MIT, MA, USA, November 2003).

[30] Perrig, A., Szewczyk, R., Tygar, J. D., Wen, V., and Culler, D. E. Spins: security protocols for sensor networks. *Wirel. Netw. 8*, 5 (2002), 521–534.

[31] Potlapally, N. R., Ravi, S., Raghunathan, A., and Jha, N. K. Analyzing the energy consumption of security protocols. In *ISLPED '03: Proceedings of the 2003 international symposium on Low power electronics and design* (New York, NY, USA, 2003), ACM Press, pp. 30–35.

[32] Sastry, N., and Wagner, D. Security considerations for ieee 802.15.4 networks. In *WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security* (New York, NY, USA, 2004), ACM Press, pp. 32–42.

[33] Stankovic, J. A. Security in wireless sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks* (2004), ACM Press, pp. 65–65.

[34] Venugopalan, R., Ganesan, P., Peddabachagari, P., Dean, A., Mueller, F., and Sichitiu, M. Encryption overhead in embedded systems and sensor network nodes: modeling and analysis. In *CASES '03: Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems* (New York, NY, USA, 2003), ACM Press, pp. 188–197.

[35] Wadaa, A., Olariu, S., Wilson, L., Eltoweissy, M., and Jones, K. Training a wireless sensor network. *Mob. Netw. Appl. 10*, 1-2 (2005), 151–168.

[36] Wang, W., and Bhargava, B. Visualization of wormholes in sensor networks. In *WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security* (New York, NY, USA, 2004), ACM Press, pp. 51–60.

[37] Wang, X., Yin, Y. L., and Yu, H. Finding collisions in the full SHA-1. Technical report, Shandong University, Shandong, China, June 2005.

[38] Wang, Y. Robust key establishment in sensor networks. *SIGMOD Rec. 33*, 1 (2004), 14–19.

[39] Yuval, G. Reinventing the travois: Encryption/mac in 30 rom bytes. In *FSE '97: Proceedings of the 4th International Workshop on Fast Software Encryption* (London, UK, 1997), Springer-Verlag, pp. 205–209.