Protocol, Mobility and Adversary Models for the Verification of Security

VOLKAN CAMBAZOGLU

UPPSALA UNIVERSITY
Department of Information Technology
Protocol, Mobility and Adversary Models for the Verification of Security

Volkan Cambazoglu
volkan.cambazoglu@it.uu.se

July 2016

Division of Computer Systems
Department of Information Technology
Uppsala University
Box 337
SE-751 05 Uppsala
Sweden

http://www.it.uu.se/

Dissertation for the degree of Licentiate of Philosophy in Computer Science

© Volkan Cambazoglu 2016
ISSN 1404-5117
Printed by the Department of Information Technology, Uppsala University, Sweden
Abstract

The increasing heterogeneity of communicating devices, ranging from resource constrained battery driven sensor nodes to multicore processor computers, challenges protocol design. We examine security and privacy protocols with respect to exterior factors such as users, adversaries, and computing and communication resources; and also interior factors such as the operations, the interactions and the parameters of a protocol.

Users and adversaries interact with security and privacy protocols, and even affect the outcome of the protocols. We propose user mobility and adversary models to examine how the location privacy of users is affected when they move relative to each other in specific patterns while adversaries with varying strengths try to identify the users based on their historical locations. The location privacy of the users are simulated with the support of the K-Anonymity protection mechanism, the Distortion-based metric, and our models of users’ mobility patterns and adversaries’ knowledge about users.

Security and privacy protocols need to operate on various computing and communication resources. Some of these protocols can be adjusted for different situations by changing parameters. A common example is to use longer secret keys in encryption for stronger security. We experiment with the trade-off between the security and the performance of the Fiat-Shamir identification protocol. We pipeline the protocol to increase its utilisation as the communication delay outweighs the computation.

A mathematical specification based on a formal method leads to a strong proof of security. We use three formal languages with their tool supports in order to model and verify the Secure Hierarchical In-Network Aggregation (SHIA) protocol for Wireless Sensor Networks (WSNs). The three formal languages specialise on cryptographic operations, distributed systems and mobile processes. Finding an appropriate level of abstraction to represent the essential features of the protocol in three formal languages was central.
Acknowledgments

I would like to thank my supervisors Christian Rohner, Per Gunningberg and Björn Victor. Among my three supervisors I spent the longest time with Christian Rohner whose dedication and encouragement have always been essential for my development as a PhD student. He is one of the important reasons why I decided to become a PhD student in the Communication Research (CoRe) group. I have always been very grateful to work with him.

It has been a great experience to interact with members of CoRe, Mobility, Uppsala Networked Objects (UNO), and Signals and Systems research groups.

Special thanks to my wife Deha and my family who have always supported me through good and bad times.
List of Papers

This thesis is based on the following papers


# Contents

1 Introduction .................................................. 3  
   1.1 Diverse Communicating Devices .......................... 4  
      1.1.1 Wireless Sensor Networks (WSNs) .................. 4  
   1.2 Security and Privacy .................................. 4  
      1.2.1 Security ............................................ 5  
      1.2.2 Privacy ............................................. 5  
      1.2.3 Security and Privacy Issues in WSNs ............... 6  
   1.3 Track of Research ..................................... 7  
      1.3.1 Simulation in Location Privacy ....................... 7  
      1.3.2 Experiment in Authentication ......................... 7  
      1.3.3 Verification in Secure Data Aggregation .......... 8  
   1.4 Contributions .......................................... 9  

2 Models for the Verification of Security Protocols ......... 11  
   2.1 Protocol Model .......................................... 12  
      2.1.1 Protocol Model for a Secure Aggregation Protocol . 13  
   2.2 Mobility Model ......................................... 17  
      2.2.1 Mobility Model for a Location Privacy Protocol .... 18  
   2.3 Adversary Models ....................................... 21  
      2.3.1 Adversary Model for a Location Privacy Protocol .. 22  
      2.3.2 Adversary Model for a Secure Aggregation Protocol . 24  

3 Papers ....................................................... 27  
   3.1 Summary of Papers ...................................... 27  

4 Research Methodology ....................................... 31  
   4.1 Simulation .............................................. 31  
   4.2 Real Experiment ........................................ 31  
   4.3 Formal Methods ........................................ 32  

5 Conclusions and Future Work ................................ 33
Chapter 1

Introduction

The overall theme of this thesis are security and privacy protocols in the context of communication networks. Security and privacy are two important topics in today’s technological world as our lives are surrounded by computers in different sizes from small smart devices to laptops. All of these computers are connected to each other and, collect and share data. While these connected data collecting devices are for the benefit of users, they should not expose the private data of the users to anyone. Security and privacy protocols play an important role in this situation by protecting devices and data. In order to fulfil their role of protection, security and privacy protocols have to be designed in a way that takes several factors into account. We distinguish between interior and exterior factors in order to analyse their influence on the protocols. Interior factors are related to the operations, the interactions and the parameters of a protocol. Exterior factors are related to entities that interact with the protocol such as users, adversaries and the environment of the protocol.

We study three security and privacy protocols through case studies with respect to both interior and exterior factors. In the first case study, we model the users’ mobility and the strength of an adversary in tracing a user in order to quantify their impact on the amount of the protection that a privacy protocol provides. In the second case study, we take advantage of the possibility that security protocols are configured by parameters which we used for a trade-off between the strength of the security and the performance of the computing and the communication resources. In the third case study, we make a statement about the security of a protocol with the support of formal languages and tools. We model the protocol in order to verify that its security objectives are fulfilled under the condition that there is an active adversary attacking the protocol.
1.1 Diverse Communicating Devices

The protocols that we consider in our case studies operate within specific contexts, where each context consists of technology with computing and communication capabilities that constrain what kind of security mechanisms and how they can be applied within the context. Therefore, the contexts comprise various devices with different features. The privacy protocol that we consider is K-Anonymity [57] and it deals with location information of users collected via their smart phones. Smart phones are connected to servers to handle location based services and also to achieve security and privacy in the service transactions. The security protocol that we consider for the trade-off between security and performance is the Fiat-Shamir identification protocol [30] and it deals with RFID systems and smart card applications. The protocol can be run on both resource constraint or rich systems, and both wired or wireless networks. The security protocol that we model and verify is the Secure Hierarchical In-Network Aggregation (SHIA) protocol [18] for Wireless Sensor Networks (WSNs). While the contexts of our case studies are heterogeneous, a particular interesting context is WSN due to its ubiquity and resource constraints.

1.1.1 Wireless Sensor Networks (WSNs)

WSNs are used for monitoring activity based on sensing in an area. A sensor node has a simple purpose such as doing sensing, taking action based on the sensing and interacting with other nodes nearby via wireless communication. A sensor node is relatively affordable and easy to deploy due to being battery operated and communicating via a wireless medium. However, it runs on minimal resources in terms of processor, memory, storage and battery. Those constraints lead to security and privacy protocol designs which are limited in terms of computational complexity, storage, network traffic and energy consumption. [47, 61]

1.2 Security and Privacy

WSNs have limited capabilities which security and privacy protocols have to take into consideration and be designed for. While security and privacy protocols have to be tailored for capabilities of each technology, they also have fundamental purposes that are explained in this section. The special issues related to the context of WSNs will follow due.
1.2. Security and Privacy

1.2.1 Security

Security is about protecting assets, such as computer systems, networks and data, from unauthorized parties regarded as adversaries. An adversary’s aim, capabilities and resources are unknown in most cases, and understanding what the adversary could accomplish is a hard problem. There are basic principles to start with when designing a solution for security [56] :

1. Which asset needs protection?

2. What is the threat?

3. How could the asset be secured?

As the questions imply, a security solution, which must deal with the questions above is specific to a scenario. An example scenario could consist of a service which delivers content only to authorised users. In this scenario, the information that needs protection is the delivered content, the threat is access of the delivered content by a non-authorised user, and the security mechanisms that need to be applied are encryption (confidentiality) and corroboration of identity (entity authentication). Encryption is applied to the content so that only authorised users can read it. Corroboration of identity is applied in order to validate the user’s identity.

1.2.2 Privacy

Privacy is defined as protection of personal data [34]. While security can be considered for any asset that needs protection, privacy is related to a user, whose data is collected from the applications in use. The user’s privacy can be achieved by protecting personal data that is provided to the applications by the user and also the data of the applications that might lead to the disclosure of the user’s identity. Here, the protection of personal data could mean anonymity, pseudonymity, unlinkability or unobservability of the personal data [56]. Anonymity ensures that the user’s identity will not be disclosed. Pseudonymity ensures that even though the user’s identity will not be disclosed, the user will be provided with a pseudonym which allows the application in use to track the private actions of the user. Unlinkability ensures that the user’s multiple private actions cannot be linked to each other and hence to the user’s identity. Unobservability ensures that it is not possible to observe the user’s private actions at all. If a user’s private data is encrypted, the data will be unobservable by a third party.
1.2.3 Security and Privacy Issues in WSNs

The lifetime of a WSN is often determined by the stored energy in the battery of a sensor node and its energy usage pattern. The most energy expensive operation performed by a sensor node is commonly the wireless communication [29]. A major design issue of WSN protocols is therefore to reduce the number of transmitted bytes exchanged between nodes. Furthermore, a sensor node’s computing and storage capabilities are significantly constrained to handle security protocols and cryptographic algorithms that require intensive computation, communication and storage [16]. Due to the resource limitations in a sensor node, bitwise exclusive-or operation (XOR) and cryptographic hash functions are among favourable operations in security protocols [16]. XOR is used in block ciphers, one-time pad and also as a mixing function. Cryptographic hash functions are used for authentication and integrity. In WSN, symmetric key cryptography is preferred over asymmetric due to lower computational complexity and better performance [16, 61]. However, there are also Elliptic Curve Cryptosystems which are examples of asymmetric key cryptography with shorter key lengths, using less resources and achieving same level of security as traditional asymmetric key cryptosystems [16].

The choice of using symmetric key cryptography raises the careful consideration of key management in a WSN. A sensor node is typically deployed unattended in a hostile environment where it can be physically captured and the secret key(s) stored inside could be revealed to an adversary. The characteristics of WSNs require new approaches for security protocols in comparison to the traditional ones. End-to-end (node-to-base station) security is not the main security goal in WSNs due to the interactions between sensor nodes. Hop-by-hop (node-to-node) communication needs to be secured in order to achieve confidentiality, authentication and message integrity as sensor nodes do in-network processing. Moreover, the communication primitives in a WSN include unicast, multicast, broadcast to one hop neighbours and broadcast to the whole network. Securing these communication primitives involves different types of keys: 1) A node key between a node and the base station, 2) A link key among a node and its one hop neighbours, 3) A pairwise key between two nodes paired in the network, 4) A cluster key among sensor nodes that are grouped together in the network, 5) A network key among all nodes and the base station. Managing these keys requires scalable and practical techniques to establish and distribute the keys to nodes. Key chains and one-time keys are used to mitigate key release through node capture, which would imply key revocation and update otherwise. [16]
1.3 Track of Research

The thesis consists of three case studies in location privacy, authentication and secure data aggregation.

1.3.1 Simulation in Location Privacy

Location based services (LbSs) [12] let users find useful information according to their location information. LbSs need to track the activity of their users in order to provide useful information. Tracking users is considered violating the privacy of the users even though it is for helping the users find useful information. There are location privacy preserving mechanisms [54] to protect the locations of the users while they retrieve useful information from the LbS. These mechanisms are not trivial and their effectiveness is influenced by several factors. We focus on two of these factors; users’ mobility patterns and adversaries who are interested in tracking the users.

We evaluate the K-Anonymity [57] location privacy preserving mechanism through the framework of the Distortion-based metric [54] to quantify the location privacy of a user. We come up with mobility traces in which users move relative to each other while being protected with the same mechanism and adversaries of varying strength (knowledge about the user) try to identify the user among others present at the same time. The details of this work will be explained in subsections 2.2.1 and 2.3.1.

1.3.2 Experiment in Authentication

We experimented with the Fiat-Shamir identification protocol [30], which is a zero-knowledge (ZK) authentication protocol. ZK protocols are examples of interactive proof systems in which a prover needs to successfully respond to the challenges that a verifier raises. [46] A ZK protocol is based on a secret that is used for proof of identity (authentication). The prover does not release any information about the secret during the protocol interaction [32]. As these protocols are probabilistic in nature, they are repeated in rounds to achieve the required level of security. With every additional round, the verifier thus can reduce the probability that a dishonest prover can convince the verifier without knowing the secret.

We identify two interior and two exterior factors related to the performance and the security of the Fiat-Shamir identification protocol. The interior factors are the number of rounds and the size of the variables in the protocol. While larger number of rounds reduces the probability of cheating, larger size of variables improves the strength of the authentication in one round of challenge-response, which relies on the square root modulo $n$ problem [46]. The size of the variables is directly related to the size of $n$ and thus to the
strength of the authentication. An increase in the number of rounds and the size of the variables leads to increased security of the protocol.

The exterior factors are network latency and computing resources. Network latency is related to how fast the responses are transferred from the prover to the verifier. Computing resources are related to how fast the computations of the challenges are performed. Achieving increased security with increased number of rounds and size of variables leads to an increase in the delay in the protocol both communication and computationwise. We identified the communication delay to be more significant than computation. Communication delay is composed of network latency and upload bandwidth from the prover to the verifier. We experimented with four networks with different characteristics (wired-wireless, low-high latency and low-high bandwidth) and two computers in low-mid and mid-high range of computing power.

In the context of a WSN, the choice of parameters is important because of resource limitations and energy constraints. If the computing resources are poor and the network latency is high, then the interior factors of the protocol can be reduced considerably. A balance between the variable size and the number of rounds should be kept as they are both necessary to support the security of the protocol. We observe that the impact of the variable sizes on the delay of the protocol is less than the number of rounds’, as an increase in the number of rounds leads to an increase in computation and encoding time. A way of finding the balance between the two interior factors can be testing the optimal variable size on the node and pick the number of rounds according to the chosen variable size and acceptable delay.

The details of this work is included in Paper II in the thesis.

1.3.3 Verification in Secure Data Aggregation

Increasing WSN deployments and extensive data collection are two prevalent activities of the last decade. Due to the characteristics and limitations of the WSNs, the collection of data should be done as efficiently as possible. Data aggregation techniques are one of the ways in line with this goal of efficiency. Data of multiple sensor nodes are aggregated using operations such as sum, average, min and max. Therefore, an aggregation topology (typically a tree) is formed in which each node needs to send only one message to its successor. The collected sensory data could belong to an individual’s smart home or a company’s work environment. A secure data aggregation protocol integrates security measures into the data aggregation techniques in order to protect the private data communicated via the WSN. [18]

We model the SHIA protocol [18] for data collection in WSNs in order to verify interior factors related to the security of the protocol such as the
operations and the interactions of the protocol. This protocol reduces the load on the communication while collecting and aggregating data securely in a WSN. We verify the protocol for achieving successful secure aggregation in the case of honest participants (non-compromised sensor nodes) in the WSN and also against being deceived with a false aggregate by an adversary. We use three formal languages and tools with different specialities in customisability, security protocols and distributed systems. The details of this work will be explained in subsections 2.1.1 and 2.3.2.

1.4 Contributions

The thesis focuses on security and privacy protocols in the context of communication networks. The main contributions of the thesis are:

- Evaluated a framework to quantify location privacy. We propose user mobility and adversary models to challenge a location privacy preserving mechanism with different scenarios and explore its effectiveness to protect a user’s privacy.

- Through pipelining of an authentication protocol, demonstration of the trade-off between security and the performance in terms of computation and network parameters.

- Modelled and verified a secure data collection protocol in three formal languages and tools.
Chapter 2

Models for the Verification of Security Protocols

A security protocol is designed based on security requirements which are reflected in the specification of the protocol. This implies that the protocol specification needs to be tested so that the security requirements are fulfilled correctly in the protocol design. Testing can be done in two ways: validation and verification. Validation is based on domain knowledge and ensures that the protocol can be used as intended in its domain. A protocol model or implementation is validated according to its specification which defines the expected usage. In contrast to validation, verification questions the specification of the protocol whether it achieves its security requirements and includes more features beyond them [5]. Modelling protocols is a prerequisite to verify their specifications in order to discover design flaws if there are any.

The main idea behind modelling is to be able to, first, understand and, then, systematically control the protocol behaviour with respect to the use of the protocol in reality. Therefore, a model is a way of representing a protocol according to the description in its specification. A common feature of models is to abstract away from some of the details that are related to the protocol, its environment or use case. The reasons of these abstractions can be many fold such as being negligible among the operations of the protocol or reducing complexity in the model or focusing on a specific aspect in isolation. These reasons are specific to each study and part of the motivation.

A communication protocol brings several parties together so that they can interact with each other by using the protocol. One of the parties is the group of users. Because user input is an external factor that can affect the protocol, it is beneficial to model the protocol’s users’ behaviour. Furthermore, user input is related to the operations of the protocol as it has
to handle the input and return meaningful output.

In the context of security protocols, an adversary is an important party that is interested in the interaction that takes place via the protocol. A security protocol has to take adversaries into account so that the protocol and its rightful users are protected from them. Models that represent adversaries and what they can do are essential for the verification of security protocol models.

In this thesis, models are used for: 1) Verifying a protocol, 2) Representing user mobility and 3) Representing adversaries. The essential and the abstract aspects of each model are specific and will be explained in the following sections.

2.1 Protocol Model

A communication protocol consists of a set of rules that defines the type of data that will be exchanged, how this data will be represented in a syntax and the semantics behind it. Data is at the core of a communication protocol and is exchanged in form of messages. The syntax of the communication protocol defines its structure of representing the data and all other necessary control information in a message. The semantics of the communication protocol defines the operation to handle a message; thus a receiver can understand what the message includes and which actions need to be taken.

An established way of verifying communication protocols is to use formal languages that can express the rules of the protocol (data, syntax and semantics). One type of formal languages is known as process algebras, which are specification languages for reactive systems [4]. Those systems consist of processes that communicate and interact with each other. Processes have states and the interaction happens through transitions between different states. The transitions depend on well defined structural operational semantics and propositional logic. A process algebra allows to represent the states and the transitions of a communication protocol in terms of its mathematical constructs. Each communicating party in the protocol is represented as processes. The data and the syntax of the protocol are represented in data types available in the process algebra. The semantics of the protocol are expressed in terms of the semantics of the process algebra.
2.1. Protocol Model

A model of a communication protocol expressed in a process algebra represents the protocol specification. It is then analysed for the internal logic and the operations of the protocol such as logical requirements that need to be fulfilled to reach a certain state of the protocol. In order to verify a protocol, there are fundamental properties [11] to check:

**Reachability property**  A particular state of the protocol can be reached.

**Safety property**  Under certain constraints one or more defined states of the protocol can never be reached.

**Liveness property**  Under certain constraints one or more defined states of the protocol will ultimately be reached.

**Fairness property**  Under certain constraints one or more defined states of the protocol will or will not be reached infinitely often.

Each property is represented as a set of goal states and verified by searching a state space of the protocol for them. In order to check properties such as reachability and safety, the state space of the protocol model has to be explored by taking transitions beginning from the initial state.

2.1.1 Protocol Model for a Secure Aggregation Protocol

There are many related works within the area of formal methods in order to verify WSN security protocols. The following works are only a part of many, but found to be closer to our work. Ballardin and Merro [8] have formalised $\mu$TESLA, which is an authenticated broadcast communication protocol with symmetric cryptographic primitives, using a timed broadcasting calculus for wireless systems. Ballardin and Merro have proven that $\mu$TESLA’s time dependent authentication property that takes place in the broadcast holds. Macedonio and Merro [45] later extended their work of modelling $\mu$TESLA with formalisation of LEAP+ [63] and LiSP [49], which are among well known key management protocols for WSNs. Tobarra et al. [58, 59] has also used the AVISPA tool [7], which allows to specify security protocols with their properties via a high-level formal language, to model SNEP [50] and, then, TinySec [40], LEAP [62] and TinyPK [60]. SNEP (Secure Network Encryption Protocol) provides data confidentiality, authentication, integrity and freshness to a WSN. TinySec brings access control, message integrity and message confidentiality to link layer communication in a WSN. TinyPK allows the use of public-key based authentication and key agreement between WSNs.

There are also efforts in comparing formal verification tools for modelling security protocols. One example is [23], which compares use of state
spaces in AVISPA [7], ProVerif [1], Scyther [22] and Casper/FDR [43, 52]. Another work [39] compares Hermes [15] and AVISPA with respect to their complexity, front-end languages, verifiable security services, intrusion models and back-end analytical tools. Another work [19] compares Casper/FDR, STA [13], $S^3A$ [27], and OFMC [9] with respect to their features and ability to detect bugs under the same experimental conditions. Perhaps, the most related work among these works is [24], which implements six popular cryptographic protocols in ProVerif and Scyther in order to outline different characteristics of these tools.

Our Protocol Model

We modelled the Secure Hierarchical In-Network Aggregation (SHIA) protocol by Chan et al [18]. SHIA is a protocol especially tailored for WSNs to collect and aggregate data securely and, at the same time, reduce the load on the communication. We verified the model of the SHIA protocol by checking reachability property for achieving successful secure aggregation in the case of honest participants in the WSN and also safety property against being deceived with a false aggregate by an adversary.

The SHIA protocol makes use of reliable broadcast as a wireless communication method, authentication as a security mechanism and recursive aggregation of data as an efficient data collection method. All of these features are essential for the protocol, yet they are hard to model altogether in a formal language.

We modelled the protocol in three formal languages: Psi-calculi [10], mCRL2 [21] and Applied Pi calculus [2]. The Psi-Calculi Workbench (Pwb) [14] is a generic tool for implementing Psi-calculus instances, and it is an expressive and customizable tool to model a large set of protocols. mCRL2 is a toolset with a well documented and a rich specification language for analysing distributed systems and protocols. ProVerif [1] is a well known tool for modelling security protocols in Applied Pi calculus. Among the three languages and tools, only Pwb was able to represent all of the features of the SHIA protocol. mCRL2 and ProVerif came short in representing broadcast communication and recursive data structures in the model. In these two tools, we represented broadcast communication as multiple unicast communication and avoided recursive data structures for small scenarios where it is possible to provide the expected outcome to the model. Even after coming up with different ways of covering these shortcomings in mCRL2 and ProVerif, we were only able to obtain basic security results, i.e. secret keys are not disclosed in any of the messages, to the queries we made to these tools. While it is important to establish basic security results, we were interested in more advanced security results where it is possible to assess
the protocol through all of its operations.

We verified the model of the SHIA protocol in a Psi-calculus using Pwb. While we were able to verify the properties we were interested in, our verification was partial in comparison to exhaustive verification which covers all aspects of the specification. This is due to a common problem in labelled transition systems which is known as the state space explosion problem [35]. When evaluating a model, the state space of the protocol is built as transitions are taken and different states are visited. Each state keeps all the processes in the protocol, where in the execution of each process is reached and constraints related to the property being verified. The verification is done by a constraint solver which takes all of these information about the processes and the constraints into account and decides whether a state that satisfies the constraints can be reached or not. As the state space of the protocol becomes larger, the task of the constraint solver becomes harder. When there are too many states with too many constraints, the constraint solver cannot decide whether the state is reachable or not.

Our initial model of the SHIA protocol in Pwb covered all aspects of the protocol; however it was too large for the tool and, hence, could not be verified automatically. Therefore, it was necessary to settle for a level of abstraction so that the tool could manage verifying the model. Three abstractions were applied:

- **Removing arithmetic operations as part of data aggregation:** We represented data aggregation in an abstract way as $\text{Aggregation}(a, b)$, instead of $(a + b)$; where $a$ and $b$ are integer values. With this abstraction, it became possible to automatically evaluate the protocol model. In the end, we checked the final aggregate syntactically instead of semantically. For example, if $a$ is 2 and $b$ is 3, then we checked the result of the aggregation as $\text{Aggregation}(2, 3)$ instead of 5. This abstraction is applied for a negligible detail in the protocol in favour of having a simpler and automatically executable model; hence, it does not affect the security properties of interest.

- **Changing the model from general to specific for the roles of the sensor node processes:** A sensor node can do both sensing and aggregation in a network. One process model for reflecting both sensing and aggregating roles was general but too big. As we defined separate processes for sensing and aggregating roles, the protocol model became more specific and was evaluated faster for the same scenario, which meant that it became easier to evaluate the model for a larger scenario with more sensor nodes. This abstraction introduces only a difference in the representation of the protocol, not in its features; thus it does not affect the security properties of interest.
• *Limiting the number of sensor nodes in the protocol*: The SHIA protocol does not limit the number of sensor nodes in a secure aggregation. However, the protocol model has to limit the number of sensor nodes so that the model could be analysed within limited time and computing resources. This limitation in the model restrains the size of the state space of the protocol; thus making it more manageable. This abstraction is the main cause of our partial verification as we are limited to a certain amount of sensor nodes and cannot cover all possible sizes of WSNs.

The evaluation time of the verification depends on the network size as Table 2.1 shows the results for Pwb with an exponential growth and resulting in the order of hours already for a network of eight nodes. The purpose of sharing these performance results is to give an idea of the difficulty of verifying a secure aggregation protocol for WSNs with many nodes and also to show why we could only do partial verification. Verifying the protocol for all possible sizes of WSNs will require infinite amount of time and it is not possible to state a specific network size or a verification time that can guarantee that the protocol is secure for all scenarios. However, our work covers the base cases for the operations of the protocol as these operations are based on sending messages downwards and upwards in a tree data structure. Increasing the number of sensor nodes in the network will affect the number of levels in the tree while in those cases the top three levels of the tree are already covered by our work. As the tree is a binary tree, 2 nodes correspond to 1 level in the tree, 4 nodes to 2 levels and 8 nodes to 3 levels.

Paper III in this thesis explains the model and the verification of the SHIA protocol in detail.

<table>
<thead>
<tr>
<th>Network Size</th>
<th>Evaluation Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3.67</td>
</tr>
<tr>
<td>4</td>
<td>10.56</td>
</tr>
<tr>
<td>8</td>
<td>9266.33</td>
</tr>
</tbody>
</table>

Table 2.1: Timing results for evaluating the SHIA protocol model
2.2 Mobility Model

A mobility model reflects the movement of a user who uses a location dependent application. This type of application collects location information with some time and geographical precision from its users. Mobility studies include either a real data set or user mobility models. A real data set consists of traces of users’ location information collected from their mobile devices over a period of time via an application in use. User mobility models allow to generate traces of users’ location information, while the users are imaginary and the traces are synthetic. Furthermore, mobility models allow to control and reproduce the user mobility easier than in a real data set.

Synthetic traces are created with the help of probabilistic models. Random walk mobility model represents a user’s movement with random directions and speeds at each time a randomly selected point is visited. Random waypoint mobility model adds on the previous model with taking pauses at visited points. Random direction mobility model represents a user moving until the edge of the considered area before changing direction and speed. Probabilistic models allow to define a set of probabilities to determine the next point that will be visited and adjust the level of randomness. Camp et al. provides a thorough survey of mobility models in [17]. There are also models that add more features on top of these such as considering movement of a group of users and map based mobility. The purpose of modelling mobility is to be able to reproduce user mobility by choosing parameters and also reflect different user movements such as completely random or following a street plan.

Moreover, heuristics about user movements help to model user mobility closer to reality. For example, people generally start their daily activity from home, take a path towards a workplace, do activity around the workplace during the middle of the day, go out to a social event in the evening, return home via a path at the end of the day and spend time around home until the next morning [28]. This behaviour of people could be combined with the previously mentioned mathematical models. For instance, when modelling a daily trace of a user, the trace starts from home location and follows the shortest path map based mobility model until workplace location [41].
2.2.1 Mobility Model for a Location Privacy Protocol

We worked on how the location privacy of users are affected by the users’ mobility. This subsection first introduces location privacy and how a user’s location information can be protected, and then describes the mobility model we consider in this context.

Location Privacy

Location privacy [12] is about protecting the location information of a user which is considered personal as accessing it allows the opportunity to query the user’s previous actions and also stalk the user in real-time.

There are Location Privacy Preserving Mechanisms (LPPMs) to protect the location information of a user from unauthorised access, hence make it private. These mechanisms include anonymisation [31], obfuscation [6], elimination [38] and addition of dummy locations [20]. Anonymisation removes the user identifier from the location information. Obfuscation makes the location information less precise. Elimination is removal of locations where a user has been at. Addition of dummy locations aims to mislead an adversary with fake locations where a user has not been at.

In this context, there is a trade-off between privacy and the granularity of information; because making user mobility completely private prevents users from obtaining mobility dependent services such as traffic information whereas revealing user mobility precisely allows an adversary to follow the user.

In order to study the location privacy of users, we experimented with K-Anonymity and Distortion-based metric.

$K$-Anonymity is an LPPM which includes both anonymisation and obfuscation. It utilises a cloaking (Figure 2.1) mechanism which forms clusters of users, who are located close to each other within a limited distance, so that all the cloaked users appears as one. The activities of $k$ users, who benefit from K-anonymity, are observed to be identity-less, and occurring at the same area and the same period. [57, 36]

Distortion-based metric (DbM) is a location privacy metric that aims to measure the distortion in the location information after an LPPM is applied. In order to measure the distortion, DbM reconstructs an actual location from an observed location by hypothesising relationship among observed locations and replacing them with possible representative locations using probability. This metric takes the distance between the actual and the observed locations, and the probability of the observed location to be the actual location into account. [54]
2.2. Mobility Model

(a) Check if there are enough users to form a cloak within a distance of a user

(b) The star icon is the randomly selected location that represents the cloak for all three users

Figure 2.1: Three users cloaked together within a green rectangle where the star icon is the cloaked location for all three users

Our Mobility Model

We considered three mobility models on an abstract level that takes a group of users’ basic movement behaviour into account. We aimed to learn how the location privacy changes when users cross another user’s path, move parallel to another user or form a circular path as other users do the same. These actions of users could also be seen in real world scenarios depending on the surroundings or the daily routines of the users. In these three models, distances between users constitute an important factor, as we worked with K-Anonymity and DbM, which are both distance based. The three models place the locations within a unit circle, as illustrated in Figure 2.2. The unit circle could correspond to an area in a town where users move within a certain radius of a location. The locations of a user that take place in this area can belong to a segment of a longer trip of the user.

Cross traces follow chords that pass through the center of the circle (i.e., diameters). Users will therefore get closer to each other until they cross each other and move apart again. From a location privacy point of view, we evaluated the protection mechanism’s ability to protect against the ability to distinguish users in the beginning and end of the trace, and providing location privacy when all the users are located close to each other.

In parallel traces, users move parallel to each other, but will not necessarily have constant distance to each other as each user’s locations are distributed independently. However, we can guarantee that the minimum
distance between two users’ locations is bounded by the distance of the two trajectories. This was a base case for location privacy where we expected uniform behaviour over the trace.

In circular traces users start at a location on the circle, keep moving on it and stop at the start location again. Users move around the same physical area in the circular trace model. However, two users can be located on diametrically opposed points on the circle at a certain time, which means greater distance between those two users than the ones in cross and parallel trace models.

All of the locations are chosen randomly with uniform and normal distributions between the two end points located on the unit circle. All of the locations are generated at consecutive time instances, e.g., each half an hour. A user might cover a long distance by using a vehicle and then a short distance by walking. Figure 2.2a illustrates that a user’s path does not necessarily have to follow a straight line, but we assume that the locations along its trace to be on the straight line between the first and the last locations of the trace.

When traces of users are generated by using the three mobility models, they are used as actual traces of users which are considered private and must be protected with an LPPM, such as K-Anonymity. Each element of a trace is an event which is composed of a user’s identity, location, time stamp and application data. The set of actual events forms the actual trace of a user (Figure 2.3). When the LPPM is applied, an actual event is transformed into an observable event where identity, location, time stamp and application data might be altered to protect the user’s location privacy. An adversary could eavesdrop on the observable events, analyse them and then build observed traces (Figure 2.3) which represent the adversary’s view of the users. According to DbM, the location privacy of a user is calculated by comparing the distances between the actual trace of the user and each
2.3 Adversary Models

An adversary is an entity that attacks or is a threat to a system [53]. It is possible to predict such an entity’s intentions and capabilities, but it could be hard to know its complete set of activities. Therefore, an adversary model is defined to specify its capabilities from which users and protocols have to be protected.

Among adversary models in communication systems, the strongest one is the Dolev-Yao adversary model [26] which can overhear, intercept and synthesise any message.

There is also the attack graph model [51], in which each node of the graph represents an attack state and each edge represents a transition that has requirements to move further in attacks. The graph also includes a goal state that the attacker wants to achieve ultimately. There are several metrics to assess an attack graph and they are listed by [3] as structural, probabilistic and time-based metrics. Structural metrics consider cases such as the shortest attack path in the graph. Probabilistic metrics treat the graph as a stochastic process such as Markov chains and Bayesian network. Time based metrics consider cases such as how fast the system can be compromised.

We modelled adversaries for two different scenarios: 1) Location privacy protocol and 2) Secure aggregation protocol.
2.3.1 Adversary Model for a Location Privacy Protocol

Among location privacy studies, the related work on adversary modelling considers how an adversary can attack the location privacy of the users [25], what exploits there are to reveal protected users’ identities and locations [33, 42], how an adversary can come to a certain conclusion about a user and what the adversary needs to achieve his goals [44, 55].

We approach adversary modelling from another perspective: how the location privacy of the users is affected when the adversary’s knowledge reaches a certain level. As the adversary’s knowledge is unknown to us, we model it in terms of probability. In the context of location privacy, the adversary is perceived as an entity who acquires observed events of users, generates possible traces out of these events and then uses out-of-band knowledge about users and/or locations to estimate which traces or events could belong to whom. Estimating traces or events is interpreted as assigning probabilities to them according to the knowledge of the adversary [54]. Adversary’s knowledge about users and their location dependent events affect the adversary’s judgement of the observed events. If the adversary knows that a user has visited a location, the adversary will consider observed events with this location and nearby locations. This behaviour is interpreted as adversary collects a large set of observed events and focuses on probable events of a user within the set.
We consider four different levels of adversary model, where each one reflects the adversary’s confidence to link a subset of the observed events to the targeted user. The weakest adversary is modelled using a uniform distribution of confidence; because if the adversary has no knowledge of the user and acquires observed events, the adversary can already say that every event is equally probable to belong to the targeted user. On the other extreme, the strongest adversary is modelled using the unit impulse as probability distribution; the adversary can select an observed event as the targeted user’s actual event with 100% confidence. In between, we use normal distributed probabilities with different variance to model weak (large variance) and strong (small variance) adversaries. Figure 2.4 shows the probability distributions of the four adversaries when there are 12 possible events at a time instance for the targeted user. For instance, in the strong adversary model, the actual event of the targeted user at the specified time instance is numbered as 6; but the adversary also thinks that events that are numbered as 5 and 7 are also very likely. Therefore, the adversary does not have one choice as in the case of the strongest adversary; but also does not think that all the possibilities are equally probable. The numbering of events in Figure 2.4 is arranged to match the visual presentation of the probability distribution functions. The probability values and the event numbers can be sorted or arranged in different ways as long as the relationship between the events and their probabilities are preserved.

For each time instance, the euclidean distances \(d_i\) between the location of the actual event and the location of the observed events are calculated. The calculated distances are weighted according to the probability \(p_i\) of each observed event, as depicted in Figure 2.5. The location privacy of a user is calculated by considering the weighted distances for all time instances. If the adversary could build a possible trace which is very close to the actual one and assigns high probabilities to the events in it, then the adversary’s error would be very small. The adversary’s error is averaged over all traces it considers; hence, the location privacy of a user depends both on the correct and wrong decisions of the adversary. It is also possible to estimate the system-wide location privacy by taking the average of all users’ location privacy. [54]
2.3.2 Adversary Model for a Secure Aggregation Protocol

SHIA is a secure protocol that aggregates data in a wireless network by creating a virtual hierarchical binary commitment tree among sensor nodes, as shown in Figure 2.6. With this aggregation technique, each node sends only one message, which contains aggregated data, to its parent in the tree. The tree in Figure 2.6 is virtual; because a physical sensor node can have both sensing and aggregating role in the WSN; however these roles are separately indicated in the commitment tree.

The assumed attacker for the SHIA protocol in a WSN aims to deceive the querier with a modified aggregate. The attacker might also try to prevent the aggregation from happening with a denial of service attack; however, we focus on the case in which the attacker tries stealthy attacks on SHIA. In this case, the importance is on how SHIA manages to run with dishonest participants. The stealthy attacker may have physical control over an arbitrary number of sensor nodes in the network. After taking control of them, the attacker has knowledge of the secret keys of the sensor nodes and can try to tamper with the aggregation. The attacker could also attack by capturing messages, modifying them or even injecting messages, and once again try to disturb the final aggregate.
2.3. Adversary Models

Figure 2.6: A hierarchical binary commitment tree for a WSN with four sensor nodes (A, B, C and D) among which A and D both sense and aggregate while B and C only sense. BS is a base station that bridges between the querier and the WSN.
We model these two types of attackers in the following ways:

- **Captured node:** An attacker who has physical access to a sensor node is modelled as a sensing process with a different parameter than a non-captured one. This parameter is an integer value for a sensor reading. The captured sensor node will try to join in the aggregation as every other sensor node and not reveal its captured status. In this case the attacker would like to lower the final aggregate value by using the captured node. In our model, we represent this attacker which contributes in the aggregation with 0°C, which is the minimum allowed value by the protocol, while all other sensor nodes sense around 20°C. As SHIA allows this behaviour, this type of attack is successful within the defined minimum and maximum ranges; because the attacker provides valid values, possesses the secret key of the captured node, and thus will be able to provide the expected responses when challenged with the result checking messages by the querier.

- **Man-in-the-middle attack:** An attacker who captures, modifies and injects messages via the wireless medium is modelled such that one or more sensor nodes are not present in the network and the attacker decides which message to send to where as whom. An absent node in the network is a missing process in the model. The aggregation and the communication, which the missing process should have done, happens according to the attacker’s will. In this case, the querier is not aware of the absence of the sensor nodes; thus, the attacker targets certain nodes in the network, blocks their communication with other nodes and acts as those nodes. When the attacker targets a node that aggregates in the network, it will capture the aggregated value and send a false aggregate instead to the expecting parent. These attacks are successful if and only if the attacker are able to obtain the secret keys of all the sensor nodes that are under the aggregating node; because before concluding the aggregation SHIA has a distributed result checking phase which collects Message Authentication Codes from all sensor nodes. In the case that the adversary cannot obtain one or more of the secret keys of those sensor nodes, the attack will be detected and the aggregate will be rejected by the querier.
Chapter 3

Papers

The papers that are included in this thesis are briefly introduced in this chapter. Each paper is presented with its title, summary and scientific contribution.

3.1 Summary of Papers

Paper I - The impact of trace and adversary models on location privacy provided by K-anonymity


Summary

Privacy preserving mechanisms help users of location-based services to balance their location privacy while still getting useful results from the service. The provided location privacy depends on the users’ behaviour and an adversary’s knowledge used to locate the users. The aim of this paper is to investigate how users’ mobility patterns and adversaries’ knowledge affect the location privacy of users querying a location-based service.

We considered three mobility trace models in order to generate user traces that cross each other, are parallel to each other and form a circular shape. Furthermore, we considered four adversary models, which are distinguished according to their level of knowledge of users. We simulated the trace and the adversary models by using Distortion-based Metric [54] and K-anonymity [37].

The results showed that the location privacy of the users increases as more users are protected by K-Anonymity, regardless of the mobility trace.
model. The maximum location privacy achieved increases as we change the model from cross to parallel and then to circular. Since K-anonymity protects traces of users using spatial cloaking, the location privacy values decrease as users come closer to each other in the trace models. Furthermore, the results indicated that the impact of the adversary on the location privacy values decreases as more users are cloaked together.

With this work, we learned that even if the security protocol in use is sound and works accordingly, there are user and adversary dependent factors that can affect the level of security that the protocol provides.

Contributions of the paper
Evaluate a framework to quantify location privacy. We propose user mobility and adversary models to challenge a location privacy mechanism with different scenarios and explore its effectiveness to protect a user’s privacy.

My contributions in the paper
I did all of the concrete work and I am the main author of the paper.


Summary
Interactive zero-knowledge protocols are used as identification protocols. The protocols are executed in rounds, with security being increased with every round. This allows for a trade-off between security and performance to adapt the protocol to the requirements of the scenario.

We experimentally investigated the Fiat-Shamir identification protocol [30] on machines and networks with different performance characteristics. The experiments were conducted with two computers among which one represented low and the other one high performance. We considered a total of four networks with a variety of latency and bandwidth among which two of them were wired and the other two wireless. We identified two interior factors in the protocol that affected its performance which were memory size of the variables and number of rounds.
We found that the fat client nature of the protocol puts stress on network latency and upload bandwidth, making communication time the leading factor in the delay of the protocol as it can rise up to 99.85% of the total time. Computation time can take up to 11.66% of the total time, when the protocol generates little amount of data that will be sent via a low latency and constant bandwidth network. The impact of the memory sizes of the variables on the delay of the protocol is less than the number of rounds’, as an increase in the number of rounds leads to a significant increase in both encoding and computation time.

With this work, we learned to make a trade-off between security and the capabilities of the computing and communication resources that a protocol runs on. Aiming for the highest level of security might be assuring; however it will cost on the performance, and some platforms and networks simply may not afford the highest level of security.

Contributions of the paper

Through pipelining of an authentication protocol, demonstration of the trade-off between security and the performance in terms of computation and network parameters.

My contributions in the paper

I did all of the concrete work and I am the main author of the paper.

Paper III - Modelling and Analysing a WSN Secure Aggregation Protocol: A Comparison of Languages and Tool Support


Summary

A security protocol promises protection of a significant piece of information while using it for a specific purpose. Here, the protection of the information is vital and a formal verification of the protocol is an essential step towards guaranteeing this protection.

In this work, we modelled a secure aggregation protocol (SHIA) [18] for Wireless Sensor Networks and verified the protocol with three formal modelling tools (Pwb [14], mCRL2 [21] and ProVerif [1]). ProVerif and
mCRL2 are mature and well-established tools, geared respectively towards security and distributed systems; however, their expressiveness constrains modelling SHIA and its security properties. Pwb is an experimental tool developed by the authors; its relative immaturity is offset by its increased expressive power and customisability.

The results of formal verification heavily depend on the model specification and the ability of the tools to deal with the model. Among the three tools, there is difference in data representation, communication types and the level of abstraction in order to represent SHIA. Therefore, we were not able to model the SHIA protocol at the same desired abstraction level in the various tools. We needed to make some concessions on using two of the tools: reliable authenticated broadcast was modelled as forwarding of point-to-point messages; authenticity was modelled with public key cryptography; recursive data structures were avoided in small scenarios where it was possible to define the expected outcome of the recursion. At the abstraction level and size of the network that we modelled, we did not find any problems with the SHIA protocol.

With this work, we learned that while we take advantage of modelling, it also has disadvantage in handling complexity. It is necessary to simplify the complexity of a modelled protocol. Moreover, formulating a model in one formal tool might be different to another one since each formal modelling tool has its own speciality and focus.

**Contributions of the paper**
Modelled and verified a secure data collection protocol in three formal languages and tools.

**My contributions in the paper**
I modelled SHIA completely in Pwb and partially in ProVerif and mCRL2. I verified the model of SHIA in Pwb completely and also helped in the verification of the model in ProVerif partially. I am the main author of the paper.
Chapter 4

Research Methodology

Our research was conducted by utilising different methods. For the study of information security, possible research methods are simulation, real experiment and formal methods. This thesis covers all of these methods.

4.1 Simulation

Simulation is the execution of a model of a computer system. The model is executed as a guest on a host system. The real system can consist of multiple hardware configurations, however the model of the whole system can run on a single hardware configuration. Therefore, the simulator imitates the modelled entities, which could be more than one, and allows to observe the model as if it is real.

Simulation was used for the location privacy work, which required a trusted server, a location based service, an adversary and a set of users. All of these entities took place in just one physical machine in the simulation. It also saved time in several aspects, such as setting up and maintenance of a single machine instead of multiple, and waiting for user input. Moreover, finding users and depending on their input are hard tasks if one does not already have an application with existing users. Furthermore, with the help of simulation, we were able to systematically control and change the parameters of our mobility and adversary models; and repeat their simulations in an automated way.

4.2 Real Experiment

Real experiment is the case in which a system is built and deployed with all of its requirements. In comparison to simulation, real experiment consists of interaction between real resources hosted on separate devices.
Real experiment was used for the case study with Fiat-Shamir identification protocol. Multiple clients and servers were implemented on computers with different hardware configurations. Each interaction between a client and a server was tested over different networks with different features and characteristics. It was convenient to have a real experiment, because the requirements to establish the setup was manageable and the focus of the work was efficiency in performance; hence a real experiment allowed us to observe the execution of the protocol over different computers and networks.

A significant difference of a real experiment from a simulation is the difficulty of producing the same result twice. A simulation environment is isolated and controlled in terms of the parameters of the experiment. However, in reality a computer has many processes running on it and a non-isolated network carries traffic that is not part of the experiment. There are so many other factors that can impact the performance. In order to avoid extreme results in our experiments, we took the average of ten runs of the same experiment. As the number of repetitions of the same experiment is increased, the results will reflect an average with a lower variance; hence it would be even better to repeat more than ten times.

4.3 Formal Methods

Formal methods allow to model a computer system in an abstract way and prove that it works according to its specification. The aim of these methods is to create a model of the system that focuses on the characteristic properties of the system and abstracts away from implementation details. Depending on the support of the modelling tool, the model may for example be executed as a simulation.

A formal method was used for modelling and verification of a secure aggregation protocol for a WSN. This approach was chosen to focus on the description of the protocol instead of testing an implementation of the protocol. Formal methods are effective in revealing fundamental design problems. Our model was created by reading and extracting the properties and features of the protocol from its description in English language. Modelling with a formal method allowed us to specify and then verify the properties and the requirements of the protocol through logical relationships which take place between interacting processes within the protocol. We also simulated the protocol by executing the model as the tools allowed us.
Chapter 5

Conclusions and Future Work

The research that we include in this thesis consists of a variety of methods and topics related to security and privacy protocols for heterogeneous communicating devices. We worked on location privacy, authentication and secure aggregation protocols by using simulation, real experiment and formal research methods. While we cover different research methods, we achieve a strong base in modelling by expressing a protocol, three user mobility traces and two adversaries in models.

First, we evaluate an LPPM (K-Anonymity) through a framework (DbM) to quantify the location privacy of a user. We produce three mobility traces in which users move relative to each other while being protected under the same cloak that K-Anonymity provides. At the same time, adversaries of four different levels of strength (knowledge about the user) try to identify the user’s cloak among others present at the same time. The results indicate that the impact of the adversary on the location privacy values decreases as more users are cloaked together. The maximum location privacy achieved increases as we change the mobility trace model from cross to parallel and then to circular. With this work, we concluded that users and adversaries are two important exterior factors that can influence the effectiveness of a security protocol.

Then, we demonstrate a trade-off between security and performance of an authentication protocol (Fiat-Shamir identification protocol). We pipeline the rounds of the protocol and observe improvement in performance. Furthermore, we identify two interior (memory size of variables and number of rounds) and two exterior (network latency and upload bandwidth) factors that affect the amount of the improvement. With this work, we concluded that the strongest security that a protocol can provide might
not be applicable to every scenario due to the capabilities of the computing and communication resources that the protocol runs on. In those cases, a trade-off between available security and performance should be considered in order to optimise the gain.

Finally, we model and verify a secure data aggregation protocol (SHIA). We consider three formal languages and tools with different specialities in customisability (Pwb), security protocols (ProVerif) and distributed systems (mCRL2). While we are able to express the protocol completely in Pwb due to its customisability, in ProVerif and mCRL2 we needed to represent broadcast communication as multiple unicast communication and avoid recursive data structures for scenarios with small networks. With this work, we concluded that while modelling allows us to automatically verify a protocol, there are too many details that need to be considered for a complete verification but lead to unmanageable complexity for the automated tools. Some of these details need to be neglected, while others need to be formulated in different ways.

**Future Work**

Now that we have an understanding of different security scenarios, and research topics and methods, there is an interesting path to consider for future work. The path is to extend the modelling with the aspects of computing and networking resources to reflect the scenario of a security protocol closer to reality. Our experiments in trade-off between security and performance showed that computing and networking resources can have an impact on a security protocol. These are two of the exterior factors that can affect a security protocol. We already modelled other exterior factors such as user mobility and adversary knowledge, and analysed their impact on a privacy protocol. We also modelled and verified a security protocol. Considering all of these works it would be complementary to model computing and networking resources as exterior factors that are part of the environment that a security protocol runs within. Each of these exterior factors introduces requirements for applying a security protocol. For example, a resource constrained computing device will limit the number of bits that a security protocol will use for a key. As we understand the environment of a security protocol more in terms of these exterior factors through modelling, it could be possible to identify a suitable security protocol that can get along with the requirements of the environment.
Bibliography


The Impact of Trace and Adversary Models on Location Privacy Provided by K-anonymity

Volkan Cambazoglu, Christian Rohner, and Björn Victor

Department of Information Technology
Uppsala University, Sweden

Abstract

Privacy preserving mechanisms help users of location-based services to balance their location privacy while still getting useful results from the service. The provided location privacy depends on the users’ behavior and an adversary’s knowledge used to locate the users. The aim of this paper is to investigate how users’ mobility patterns and adversaries’ knowledge affect the location privacy of users querying a location-based service. We consider three mobility trace models in order to generate user traces that cross each other, are parallel to each other and form a circular shape. Furthermore, we consider four adversary models, which are distinguished according to their level of knowledge of users. We simulate the trace and the adversary models by using Distortion-based Metric and K-anonymity. The results show that the location privacy provided by K-anonymity decreases, as users are located closer to each other in the trace models. The impact of the adversary on location privacy is reduced as more users are cloaked together.

1 Introduction

New types of smart mobile devices enable the emergence of Location-based Services (LbS). A user of the service carries a mobile device that obtains its location via GPS or a WLAN. Then, with the help of a service provider, the device can discover nearby places or the whereabouts of a friend [1]. The advantage of this service is allowing users to find useful information based on their locations. While the LbS helps users to reach places or people easily, private information of users may be revealed not only to the LbS provider but to anyone. The aim of achieving location privacy is to prevent the disclosure of locations and mobility patterns of individual users to other entities. Thus, one should not be able to make an identity-location binding,
The TS protects the users and their locations
The LbS provides location dependent information to the users
The adversary is interested in the LbS data

The users have locations and use the LbS

Figure 1: System Overview

i.e. being able to tell that a specific user has been to a specific location [12]. There are existing Location Privacy Preserving Mechanisms (LPPM) [9] that provide location privacy to a user of a LbS. These mechanisms are applied to the private data of the user, such as location and identity, before the user communicates with the LbS. Since we consider a centralized architecture in this paper, the LPPM is applied on a Trusted Server (TS) that can securely communicate with the user and takes place in between the user and the LbS as depicted in Figure 1. When the LbS obtains the protected data of the user, it prepares location dependent information that the user asked for and sends it to the TS, which forwards it to the respective user. In this setting, we are concerned with the protection of users’ private data that the LbS obtains. Hence, we assume that the communication channel between the users and the TS is secure, and the protected data that is sent from the TS to the LbS is open to observation by an adversary, who can be the LbS provider or anyone else. The level of the location privacy a user can achieve in the LbS is measured by evaluation of the data that the adversary can obtain. There is also a trade-off between the quality of the service that the LbS provides to its users and the level of location privacy that the users experience [10]. For example, a user can obtain the best results that the LbS has to offer when there is no protection on the user’s data.

The location privacy problem can be broken down to private data of users, the protection mechanism, knowledge of the adversary, and the evaluation metric [9]. Users may repeat the same behavior over time, such as being at home during the night, that a protection mechanism cannot protect against or an adversary can pinpoint. All protection mechanisms can have strengths and weaknesses. Different adversaries can have different amounts of information on users. The evaluation metric for location privacy can capture certain details; but miss others. Therefore, each of these components can be perceived as variables that can affect the location privacy of a user. In this paper, we select the protection mechanism and the evaluation metric
from the set of existing ones. We focus on how these protocols are influenced by the private data of users and the knowledge of the adversary.

The contributions of this paper are the generation of the mobility traces of users based on abstract models and the consideration of different probability distribution functions for modeling the knowledge of the adversary. We simulate the effects of different trace and adversary models on K-anonymity [3] protection mechanism and evaluate the results using the Distortion-based Metric (DbM) [9]. The results show that the location privacy that K-anonymity provides increases as the size of the cloaked area increases. Perhaps more surprisingly, the effectiveness of K-anonymity decreases as users are located closer to each other, even if more users are cloaked at once by K-anonymity. Furthermore, the difference that the adversary’s level of knowledge can make on the location privacy of the users decreases as K-anonymity cloaks more users together.

2 Related Work

Most of the works on location privacy simulate their protocols on real world traces. These real world traces are extracted from different scenarios such as mobility traces of a person [7], taxi cabs [11] and cars [6]. Some of the works consider building synthetic traces of users by using the random waypoint mobility model and its variants [8]. There are also works in which the traces are generated as a result of traffic simulations based on survey data [5]. In comparison to the related work, we consider the trace models on an abstract level that takes users’ basic movement behavior into account. We aim to learn how the location privacy changes when users take certain actions, such as cross another user’s path, move parallel to another user or form a circular path as other users do the same. These actions of users can also be seen in real world scenarios depending on the surroundings or the daily routines of the users.

The related work on adversary modeling approaches the problem from various perspectives. There are works that aim to define the attack model that the adversary can exploit [2]. For example, when the queries of the users are not anonymized, the adversary can check the presence or absence of the user at a place or when the queries are anonymized, the adversary can still attack by knowing a place [4] or making an observation [7]. In [8], an active adversary, who can physically trace the users, is also considered. Furthermore, there are works [11] that combine all of these aspects, where adversaries model user movements using Markov Chains. The related work on adversary modeling, thus, considers how an adversary can attack the location privacy of the users, what exploits there are to reveal protected users’
identities and locations, how an adversary can come to a certain conclusion about a user and what the adversary needs to achieve his goals. We approach the problem from another perspective, which is asking the question how the location privacy of the users is affected when the adversary’s knowledge reaches a certain level. We consider four levels of adversary model and investigate how much location privacy the users can lose according to the adversary model.

3 The Private Data of the Users

In this paper, we choose to define the private data of the users as their locations and mobility patterns. This corresponds to a trace [9], i.e. a sequence of events which are placed in their order of generation. An event [9] is composed of a user’s identity, location, time stamp and, optionally, message content. The events are distinguished as actual, observable and observed events. An actual event is an event that is generated by a user when querying the LbS; hence, a user’s trace consists of the user’s actual events and is called the actual trace of the user. When the LPPM is applied at the TS, an actual event is transformed into an observable event where identity, location, and time stamp might be altered to protect the user’s location privacy. Observable events are sent from the TS to the LbS. The adversary that eavesdrops on the LbS observes a subset of the observable events and acquires the observed events. The adversary analyzes the observed events and then builds the observed traces which represent the adversary’s view of the users.
3.1 Generation of Traces of Users

We generate actual traces of users before applying the LPPM on them. In order to study location privacy properties of specific mobility patterns we generate the events using abstract models where the paths of the users cross each other, are parallel to each other, and form a circular shape, respectively. The models place the events within a unit circle, as illustrated in Figure 2. In all trace models, the algorithm picks the first event location at random on the unit circle. The other events are then distributed along a trajectory along either a chord departing from the first event, or the circle itself. The last event of a trace is located on the circle again. Cross traces follow chords that pass through the center of the circle (i.e., diameters). Users will therefore get closer to each other until they cross each other and move apart again. From a location privacy point of view, we will evaluate the LPPM’s ability to protect against the ability to distinguish users in the beginning and end of the trace, and providing location privacy when all the users are located close to each other. In parallel traces, users move parallel to each other, but will not necessarily have constant distance to each other as each user’s events are distributed independently. However, we can guarantee that the minimum distance between two users’ events is bounded by the distance of the two trajectories. This is a base case for location privacy where we expect uniform behavior over the trace. In circular traces users start at a location on the circle, keep moving on it and stop at the start location again. Users move around the same physical area in the circular trace model. However, two users can be located at the opposite poles of the circle at a certain time, which means greater distance between those two users than the ones in cross and parallel trace models.

All of the event locations are chosen randomly, where randomness is achieved on uniform and normal distributions between the two end points located on the unit circle. All of the events are generated at consecutive time instances, e.g., each half an hour. A user might cover a long distance by using a vehicle and then a short distance by walking. Figure 2(a) illustrates that a user’s path does not necessarily have to be on a straight line, but we assume that the events along its trace to be located on the straight line between the first and last events of the trace. The unit circle could correspond to an area in a town where users move within a certain radius of a location. The events of a user that take place in this area could belong to a segment of a longer trip of the user.
4 Adaptation of K-anonymity to Distortion-based Metric (DbM)

K-anonymity [13] is a privacy protection mechanism that was considered first in database systems. The private information of \( k \) users is protected by having the same values for their unique identifiers before being publicly released. K-anonymity was later considered within the context of the location privacy [5]. In order to achieve the same goal as before, K-anonymity applies anonymization and obfuscation mechanisms on users’ actual events. The events of the \( k \) users, who benefit from K-anonymity, are observed to be identity-less, and occurring at the same area and the same period. The efficiency of K-anonymity in protecting the location privacy of users is discussed in [12].

We selected [3] as an example of K-anonymity solution that considers a personalized approach for each user. The values \( k \), \( t_x \), \( t_y \) and \( t_t \) are kept as personal values of each user. The \( k \) value represents the number of users that will be cloaked together, \( t_x \) is the toleration in x-coordinate, \( t_y \) is the toleration in y-coordinate and \( t_t \) is the toleration in time space. The toleration values are used to check if a cloak of \( k \) users can be composed. When all users are confirmed to be inside the toleration values of their neighbors, the smallest area that includes all of the \( k \) users is calculated. If there are not enough users to form a cloak of \( k \) users within the toleration values, then the users are clearly visible to an adversary. The personalized approach allows users to benefit from the service more frequently, as the scenarios in which K-anonymity does not work can be by-passed. In our simulations, the aim is to protect every user with K-anonymity; thus, we choose to use system-wide values for simplicity of simulations. The toleration in x and y coordinates are chosen as 2 units, which is the smallest distance that can cover \( k \) of the users at once within the unit circle. The toleration in time space is chosen as 28 minutes, as each new event is generated in half an hour.

We used the DbM [9] as evaluation model for the location privacy in order to assess our simplified K-anonymity implementation. The DbM is a location privacy metric, which aims to estimate the expected distortion that an adversary will encounter when trying to reconstruct the actual traces of the users by hypothesizing the relationships among observed events and replacing them with possible representative events using probability. The DbM is based on traces, which are sequences of locations. However, K-anonymity works with areas and is not suitable for traces. In order to adapt the cloaked area that K-anonymity provides to a location, which DbM expects, we picked a random point inside the cloaked area. Selecting a random point in an area might lead to underestimation or overestimation in the location privacy re-
sults. However, we take the average and calculate the standard deviation from 100 simulation runs, in order to avoid such extremes. After having the locations, the observed traces that the adversary builds are created according to the time instances and the number of cloaks at each time instance. The number of observed traces is calculated by the formula \((\frac{U}{k})^T\); where \(U\) is the number of all users, \(k\) is the number of users in each cloak and \(T\) is the number of time instances. If \(k\) is equal to the number of users, then it appears as if there is only one trace in the system.

5 Adversary Modeling

The adversary is one of the key variables in the location privacy problem. As the adversary’s knowledge is unknown to us, it is modeled in order to assess how much location privacy a user can experience. The adversary acquires observed events of users, generates possible traces out of these events and then uses out-of-band knowledge about users and/or locations to estimate which traces or events could belong to whom. Estimating traces or events is interpreted as assigning probabilities to them according to the knowledge of the adversary. According to the DbM [9], the adversary selects a beginning time instance, starts considering possible events from on that time instance, places each event in separate traces weighted by different probabilities and repeats this process until the last time instance chosen. When this process is over, the adversary has the observed traces.

We consider four different levels of adversary model, where each one has approximately half the confidence of the previous one on the targeted event/trace as they change from the strongest to the weakest. The weakest adversary is modeled using a uniform distribution; because if the adversary has no knowledge of the user and acquires observed events, the adversary can already say that every event is equally probable to belong to the targeted user. On the other extreme, the strongest adversary is modeled using the unit impulse as probability distribution; the adversary can select an event as the targeted user’s event with 100% confidence. In between, we use normal distributed probabilities with different variance to model weak (large variance) and strong (small variance) adversaries. Figure 3 shows the probability distributions of the four adversaries when there are 12 possible events at a time instance for the targeted user. For instance, in the strong adversary model, the event of the targeted user at the specified time instance is numbered as 6; but the adversary also thinks that events that are numbered as 5 and 7 are also very likely. Therefore, the adversary does not have one choice as in the case of the strongest adversary; but also does not think that all the possibilities are equally probable.
Figure 3: Probability distribution functions over 12 possible events at a time instance

After building the observed traces and assigning probabilities to the events that are included in them, the adversary makes a decision among the set of possibilities to track down the targeted user. The location privacy of the user is calculated by comparing the distances between the actual trace of the user and each of the observed traces that the adversary considers [9]. The calculations start from the beginning time instance as in the building of the observed traces. At each time instance, the euclidean distance between the location of the actual event and the location of the observed event is calculated. The calculated distance is weighted according to the probability of the observed trace that the event belongs to. This process continues until all possible observed traces and all time instances are considered. For example, if the adversary could build a possible trace which is very close to the actual one and assigns high probability to it, then the adversary’s error would be very small. The adversary’s error is averaged over all traces; hence, the location privacy of a user depends both on the correct and wrong decisions of the adversary. When the location privacy of all users are calculated, it is also possible to estimate the system-wide location privacy. As DbM is based on weighted distances between the actual and the observed traces, the range of the location privacy values is not necessarily from 0.0 to 1.0; it is scenario dependent. For instance, the range of the location privacy values can be from 0.0 to 2.41, when the trace model is circular and to 2.24, when
the trace model is cross or parallel. The highest values are calculated by considering the maximum possible distances within the cloaks.

6 Results

We evaluate the location privacy of the users for different mobility trace and adversary models. The evaluation is based on the simulations where users’ mobility traces are protected by K-anonymity and the protected traces are assessed for location privacy by DbM while different adversary models are considered. We consider 12 users and 10 time instances when generating the mobility traces of the users. The \( k \) values that we considered are 1, 2, 3, 4, 6 and 12, in order to serve all of the users with K-anonymity. The plotted location privacy values correspond to system-wide location privacy, which is the average location privacy over all users. Since the metric is DbM, the location privacy values indicate the adversary’s expected error in locating the users.

The simulation results, depicted in Figure 4, are consistent with the papers on K-anonymity [3]: when \( k \) value increases, the location privacy of the users also increases. They evaluate the location privacy differently from us; however, we can observe the increase in location privacy because of the widened area in order to cloak more users at once, as shown in Figure 5. The results show that location privacy is improved, regardless of the trace model, when more users are cloaked together.

The location privacy that each user experiences is the lowest in the cross trace model, where the area of the cloaking box becomes small as users come closer to each other. When cloaking boxes get smaller, distances between actual and observed events become shorter which leads to smaller distortion and in return less location privacy. In the parallel trace model, the cloaking box covers a larger area, in which users could enjoy higher location privacy values. However, the users are still located inside the circle most of the time, thus the location privacy values are slightly higher than the ones in cross traces. The location privacy of the users is the highest in the circular trace model where the actual events of the users are located on the unit circle. The adversary experiences higher distortion in circular trace model than in the other ones, as the distances between actual and observed events are maximized on the unit circle. The highest location privacy achieved in different trace models are roughly 25%, 33% and 48% for cross, parallel and circular respectively.

Figure 6 compares the amount of change in the location privacy with respect to the area of the cloaking box when the events are normally and uniformly distributed in the cross trace model. The location privacy values
Figure 4: Location privacy achieved in different trace models when the adversary is the strongest.

Figure 5: Change of location privacy with respect to change of area of the cloaking box; when the trace model is circular, the adversary is the strongest and different $k$ values are used.
achieved in uniformly distributed events are 10-15% higher than the values achieved in normally distributed events, even with the strongest adversary model. This change in the location privacy is observed specially in cross traces; because the events are more closely located when they are normally distributed. When the events are normally distributed, most of the events take place in the middle of the trace where the users cross each others’ paths. The distances between the events are wider when they are uniformly distributed; hence the size of the cloaked areas become larger. Thus, we again see that being closely located to other users causes location privacy to decrease in the case of K-anonymity.

Moreover, Figure 7 shows the decrease in the impact of the adversary on the location privacy results as $k$ increases. The adversary models strongest, strong, weak and weakest are represented with standard deviation as 0, 1, 2 and 4 respectively on the x-axis of Figure 7. The cloaked area becomes larger as more users are cloaked together, which increases the distortion that the adversary could encounter. Furthermore, the number of possibilities decreases as $k$ increases, which leaves the adversary with fewer options. For example, all of the adversary models have the same impact on the location privacy results when all of the users are cloaked together, because there is only one trace and each event of the trace has 100% probability.

Figure 8 shows how system-wide location privacy values change in the circular trace model with respect to the adversary’s knowledge of users. When K-anonymity is the LPPM applied on the actual events, one can see the highest and the lowest location privacy values achieved with respect to increasing $k$ values from Figure 8. As the strength of the adversary weakens, the corresponding location privacy curve will increase from the curve of the
Figure 7: Change of location privacy with respect to change of standard deviation, when different $k$ values are used and the trace model is circular.

Figure 8: Location privacy achieved in circular trace model, while different adversary models are considered.
strongest adversary up to the curve of the weakest adversary.

Furthermore, Figure 9 reflects the claim of Shokri et al. [9] that the adversary’s probability assignment of users is not visible when all of the observed events seem equally probable. They also say that K-anonymity provides a reasonably constant level of location privacy independent of the adversary’s choices. When the adversary model is the weakest, the level of location privacy does not change much; however this assumption may not hold in reality.

7 Conclusions

The results show that the location privacy of the users increases as the $k$ value increases, regardless of the mobility trace model. The maximum location privacy achieved increases as we change the model from cross to parallel and then to circular. Since K-anonymity protects traces of users using spatial cloaking, the location privacy values decrease as users come closer to each other. Furthermore, the results indicate that the impact of the adversary on the location privacy values decreases as more users are cloaked together.

If, by examining the generated traces of users and the adversary models,
we could deduce how the location privacy results are affected, how effective
the protection mechanism is, and what kind of patterns are matched in the
location privacy results, it may be possible to take precautions in certain
situations by learning from these patterns.

This work was a starting point to study the location privacy problem. In
the future, we would like to consider mobility models where actual city plans
restrict user mobility. Moreover, we would like to study the impact of trace
and adversary models on other LPPMs with the aim of creating a system in
which users can select the most suitable protection mechanism according to
the scenario they are in.

References

tweakersoft.com/aroundme.html.

International Conference on Mobile Business, Washington, DC, USA,

ized anonymization model. In Proceedings of ICDCS ’05, Washington,

Springer-Verlag.

services through spatial and temporal cloaking. In Proc. of MobiSys
’03, New York, NY, USA, 2003. ACM.

Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for dis-
tributed privacy-preserving traffic monitoring. In Proc. of MobiSys ’08,
New York, NY, USA, 2008. ACM.


of published anonymous mobility traces. In Proc. of MobiCom ’10, New
York, NY, USA, 2010. ACM.


Paper II
Towards Adaptive Zero-Knowledge Protocols: A Case Study with Fiat-Shamir Identification Protocol

Volkan Cambazoglu and Christian Rohner
Department of Information Technology
Uppsala University, Sweden

Abstract

Interactive zero-knowledge protocols are used as identification protocols. The protocols are executed in rounds, with security being increased with every round. This allows for a trade-off between security and performance to adapt the protocol to the requirements of the scenario. We experimentally investigate the Fiat-Shamir identification protocol on machines and networks with different performance characteristics. We find that the delay of the protocol highly depends on network latency and upload bandwidth. Computation time becomes more visible, when the protocol transmits little amount of data via a low latency network. We also experience that the impact of the sizes of the variables on the delay of the protocol is less than the number of rounds’, which are interior factors in the protocol.

1 Introduction

In a zero-knowledge (ZK) protocol, a party (the prover) can prove to another party (the verifier) possession of a secret $s$ without revealing any information about the secret [4]. This property can be used for proof of identity (authentication) where the prover has to compute values using its secret $s$ and transmit the computed values to the verifier. One example is the Fiat-Shamir identification protocol [3]. The protocol is executed over several rounds, as in every round there is a chance of 50% for a dishonest prover to successfully compute the challenges given by the verifier without knowing the secret. With every additional round, the verifier thus can reduce the probability of cheating.

While increase in the number of rounds give rise to increase in the security of the protocol, it also leads to increase in computation and communication time. Furthermore, the memory size of the variables (e.g. secret $s$), can
be adjusted and it also has a positive correlation with the security and the
delay of the protocol. These properties allow a trade-off between the level
of security and the performance involved with the protocol, depending on
the scenario. In this paper we study how the choice of parameters for the
number of rounds and the memory size of variables, with respect to different
computing and network conditions, affect the performance of the protocol to
find an optimal trade-off.

In the next section, the Fiat-Shamir protocol’s internal structure and use
of vectors to improve its performance will be explained. In Section 3, the
impact of vector of rounds, variable size of $s$, number of rounds, network
latency and computing resources on the performance of the protocol will be
presented. Later, we will discuss our findings. The related work on the Fiat-
Shamir protocol will be covered in section 5. We will conclude the paper
with our findings, an idea of adapting a ZK protocol and future work.

2 The Fiat-Shamir Identification Protocol

The protocol has two phases: (1) one time set-up and (2) rounds of three-way
challenge-response.

At the beginning, one time set-up takes place between a prover and a
trusted center. The purpose of the one time set-up is to register the prover’s
public key ($i = s^2 \mod n$, where $s$ is the secret and $n = p \times q$ is the product
of two prime numbers) at the trusted center.

After the one time set-up, there are a number of rounds of three-
way challenge-response (Figure 1) between the prover and the verifier.
At each round, the prover makes a commitment by generating a random
number $r$ and computing $x = r^2 \mod n$, hence the rest of that round depends
on the value of $r$. When the verifier receives the prover’s commitment, it
challenges the prover with a random bit $e$. Then, the prover has to reply
with $y = r \times s^e \mod n$, which includes both the random number $r$ of the
round and the secret $s$. The verifier verifies the response of the prover by
computing $y^2 = x \times i^e \mod n$, where $i$ is the registered public key of the
prover, $x$ is the commitment of the round and $e$ is the random bit generated
by the verifier. If the verification of $y^2$ holds, the prover is approved for that
round by the verifier. The three-way interaction is repeated for $k$ rounds,
where the prover has to provide the correct response at each of the $k$ rounds
in order to be authenticated by the verifier.

The security of the Fiat-Shamir identification protocol relies on
two factors: (1) the sizes of the variables and (2) the number of rounds.
The sizes of the variables affects the security of the protocol; because
“The security relies on the difficulty of extracting square roots modulo large
composite integers $n$ of unknown factorization, which is equivalent to that of factoring $n$" [6]. Therefore, the protocol becomes more secure as the size of $n$ increases. In the protocol, the largest variable size belongs to $n$ and is four times the size of the smallest one ($s$). For example, if $n$ should be 1024 bits, then $s$ must be 256 bits. The security of the protocol also relies on the number of rounds ($k$). There is 0.5 probability of cheating by a dishonest prover at each round. The cheating can occur when $e = 0$, as the dishonest prover can compute $x$ according to the chosen $r$. However, the dishonest prover cannot cheat when $e = 1$, because the knowledge of $s$ is required and it is infeasible to take the square root of $i \mod n$. If the dishonest prover also does not know the random $r$ of the round, then it cannot cheat even when $e = 0$, as it is also infeasible to take the square root of $x \mod n$. Thus, increasing the number of rounds increases the level of security as the possibility of cheating decreases with each round.

The efficiency of the Fiat-Shamir protocol can be improved by using vectors collapsing $k$ rounds into one. Executing the rounds in parallel eliminates time consuming waiting due to network latency. When vectors are used, each message consists of $k$ many values of each variable separated by a delimiter; thus, replacing transmissions of small size $4 \times k$ packets with large size 4 packets. For example, when the number of rounds is 3, the vector of $x [x_1 \ x_2 \ x_3]$ will be sent from the prover to the verifier. The computations of the protocol are not affected due to the independence of each round from others. While it is not possible to terminate in the middle of the communication due to an incorrect response from the prover as in the iterative approach, the vector approach increases the efficiency of successfully authenticating after

Figure 1: A round in the Fiat-Shamir identification protocol
<table>
<thead>
<tr>
<th>Network Name</th>
<th>Avg. RTT (ms)</th>
<th>Stddev. RTT</th>
</tr>
</thead>
<tbody>
<tr>
<td>3G</td>
<td>459.13</td>
<td>37.0</td>
</tr>
<tr>
<td>home</td>
<td>18.5</td>
<td>1.15</td>
</tr>
<tr>
<td>eduroam</td>
<td>2.9</td>
<td>1.1</td>
</tr>
<tr>
<td>office</td>
<td>0.72</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Table 1: Network latency

$k$ rounds. Moreover, the vector approach allows to improve the computation time of the protocol using threading on a computer with a multi-core CPU as each round can be computed in itself on a single core.

3 Results

We evaluate the performance of the Fiat-Shamir identification protocol with the goal of adaptive performance in the context of ZK protocols. We observe that four factors have impact on the performance of Fiat-Shamir: (1) the variable size of $s$, (2) the number of rounds, (3) network latency and (4) computing resources. While the variable size and the number of rounds form the interior aspects of the protocol, network latency and computing resources are exterior factors which influence the protocol. In order to understand the impact of these four factors on the protocol, we carry out 450 experiments in total where 360 are obtained from a laptop (MacBookAir4,2) with access to four different networks (Table 1) and 90 from a desktop computer (Acer Aspire M3970) with access to the home network. In the experiments, we consider three different variable sizes and number of rounds, and also repeat each experiment ten times to handle variation.

The protocol is implemented in C using the OpenSSL [1] library. The implementation is done in a client and server architecture, where the client is the prover and the server is the verifier. The results are obtained from the runs of client code, which are executed on different machines and from different networks, while the server is always connected to the same network.

3.1 Vector of rounds

As it is mentioned in [5], using vectors to send all rounds at once is more efficient than sending one variable at a time. We observed the same result such that the iterative approach is always slower than the vector approach. Furthermore, in our experiments, the gain of using vectors varies from 3-
85 times faster depending on the four factors. These limits are observed when the data, which is generated by the protocol, is the largest (512 bits sized $s$ and 512 rounds) in our experiments. The lowest limit (3) is experienced with lowest latency network (office) and poor computing resources (MacBookAir4,2) whereas the highest limit (85) is reached with high latency network (home) and rich computing resources (Acer Aspire M3970). The details of these factors will be explained in the following sections, where the results are extracted from the experiments with the vector approach.

### 3.2 Components of delay

The total time to authenticate the prover can be delayed due to two main reasons: (1) computation and (2) communication. The computation time consists of the times spent on calculations of variables $r$, $x$, $e$, $y$ and $y^2$. Most of the computation time is spent on the prover side as the variables $r$, $x$ and $y$ are computed by it. The communication time is two-fold as it includes encoding/decoding on each side and also sending/receiving data from one side to the other via Transmission Control Protocol (TCP) sockets. Once again, most of the communication time is spent on the prover side due to $k$ many values of $x$ and $y$. Therefore, the protocol requires commitment from the prover by all means. In order to show the interplay between computation and communication times, the lowest and the highest bounds experienced in our experiments are:

- 0.15% computation time + 99.85% communication time (3.5% encoding + 96.35% socket) : observed with largest data (512 bits sized $s$ and 512 rounds) sent via the highest latency network (3G)

- 11.66% computation time + 88.34% communication time (37.13% encoding + 51.21% socket) : observed with smallest data (128 bits sized $s$ and 128 rounds) sent via the lowest latency network (office)

When the prover communicates with the verifier using a low latency network and sending small amount of data, the computational delay of the protocol becomes highlighted. On the contrary, as the latency in the network and the size of the data increases, the computation time becomes negligible.

### 3.3 Interior aspects

**Variable size of $s$:** Figure 2 shows the change in time to authenticate the prover as the size of $s$ changes. Each curve corresponds to runs of the protocol inside the office network (Table 1) with a certain number of rounds. We observe that increasing the size of $s$ from 128 bits to 256 bits results in
Figure 2: Impact of the size of $s$ on the time to authenticate the prover via the office network.

96% increase in time on average, whereas increasing from 256 bits to 512 bits results in 66% increase in time on average. Therefore, when RTT is low, 512 bits sized $s$ can be more favourable than 256 bits in order to achieve higher security with better efficiency. The curves follow a logarithmic growth, which means that, as the size of $s$ increases, the total time to authenticate the prover can scale up to a limit.

**Number of rounds:** Figure 3 shows the change in time to authenticate the prover as the number of rounds in the protocol changes. Each curve corresponds to runs of the protocol inside the office network (Table 1) with a certain variable size of $s$. Increasing the number of rounds from 128 to 256 results in 161.35% increase in delay of the protocol; while an increase from 256 rounds to 512 leads to 219.97% increase in delay. The curves are exponentially growing, thus increasing the number of rounds will only consume more time; because of generating more data with each increase.

### 3.4 Exterior aspects

**Network latency:** Figure 4 shows the change in time to authenticate the prover as the network latency changes. The values in the plot are obtained
Figure 3: Impact of the number of rounds on the time to authenticate the prover via the office network

from a laptop computer when placed in four different networks. Total data transfer, on the x-axis, represents the size of the problem, which is defined by the size of $s$ and the number of rounds that are presented in Table 2. The order of the rows of the table matches the order of plotted points in the graph. One can observe that the number of rounds generally has more impact than the size of $s$, as it is visible from the steps at 29000 and 116000 bytes in Figure 4; because when the number of rounds increase, it leads to increase in both computation time and encoding/decoding time even though the socket time remains the same. For example; in the office network, when 512 bits sized $s$ and 256 rounds take 0.038 seconds in total (0.012 sec. in socket + 0.002 sec. in computation + 0.024 sec. in encoding), 256 bits sized $s$ and 512 rounds take 0.074 seconds in total (0.012 sec. in socket + 0.003 sec. in computation + 0.058 sec. in encoding).

Network latency is the most crucial aspect in this problem as the total time to authenticate the prover is highly dependent on it. As mentioned in subsection 3.2, up to 99.85% of the total time can be consumed by the communication time in the case of 3G in Figure 4. The curve of 3G follows the same trend as the office network, but the values are amplified. 3G curve reaches up to 2.29 seconds when the largest data is sent between the
prover and the verifier. The reasons of observing delay around 2 seconds is the dependency of the protocol on the upload link and 3G’s low upload bandwidth compared to the other networks.

Computing resources: In our experiments, we executed the client side (the prover) of the protocol on two different machines (MacBookAir4,2 and Acer Aspire M3970) via the home network (Table 1). The results indicated that as the problem size (Table 2) widened, the performance of the two machines clearly drifted apart from each other. MacBookAir4,2 performed 17.56% better when the problem size is the smallest, while at other problem sizes Acer Aspire M3970 performed 63.69% better on average. Furthermore, we observed that while it always consumes more time to process larger variable sizes on MacBookAir4,2, 256 bits sized $s$ takes less than or equal time to 128 bits on Acer Aspire M3970.

4 Discussion

In this section, we discuss various ways to approach our findings of the four factors in the context of ZK protocols. We think that a ZK protocol can be used in a time critical setting such that the parameters of the protocol can
be optimized according to the constraints of the setting. Hence, network conditions, computing resources and the time limit become the major parameters to consider. According to the network conditions and computing resources, a table of variable sizes and number of rounds can be composed and, then, the optimal values from the composed table can be computed using the time limit.

In one scenario, where the prover has low latency and high upload bandwidth network connectivity to the verifier, it could be a better choice to select larger variable sizes as it will provide higher security with a short delay. However, a balance between the variable size and the number of rounds should be kept as they are both necessary to support the security of the protocol. A way of finding the balance between the two interior factors can be testing the optimal variable size on the machine/device and pick the number of rounds according to the chosen variable size and the time limit. Therefore, instead of merely reducing the problem size in a time critical scenario, the present computing resources are taken into account to achieve balance between the security and the performance of the protocol.

In another scenario, the network between the prover and the verifier can have high latency and/or low upload bandwidth. In this case, the number of rounds should be kept as low as possible while the variable sizes should be as large as possible to balance the security within the time limit. If the computing resources are also poor while the network latency is high, then the interior factors of the protocol can be reduced considerably. Still, it can be a good choice; because, as observed in the comparison of MacBookAir4,2 and Acer Aspire M3970, a machine with poor computing resources can still perform better on a simple problem than a resource rich one.

<table>
<thead>
<tr>
<th>Size of s</th>
<th>Number of Rounds</th>
<th>Total Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>128</td>
<td>14500</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>29000</td>
</tr>
<tr>
<td>128</td>
<td>256</td>
<td>29000</td>
</tr>
<tr>
<td>256</td>
<td>256</td>
<td>58000</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>116000</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>116000</td>
</tr>
<tr>
<td>512</td>
<td>512</td>
<td>232000</td>
</tr>
</tbody>
</table>

Table 2: Problem sizes with respect to variables and rounds
5 Related Work

The Fiat-Shamir identification protocol [3] was considered as a signature scheme for microprocessor-based devices. The protocol was modified several times both by Fiat-Shamir and others to address either shortcomings or different scenarios. For example, the protocol had to be used memory-efficient [7] and time-efficient [2] due to limited resources in smart-card applications. Use of vectors instead of single value rounds was another performance focused alteration in the protocol [5]. In order to have an understanding of adaptivity in terms of performance of the protocol, we investigate the degree of influence of memory and time dependent factors in the protocol, in light of the previous works.

6 Conclusions

In this study, we investigated the Fiat-Shamir identification protocol, which is a ZK proof, with the idea of an efficient security protocol. After running the protocol on different machines and networks, we find that the fat client nature of the protocol puts stress on network latency and upload bandwidth, making communication time the leading factor in the delay of the protocol as it can rise up to 99.85% of the total time. Computation time can take up to 11.66% of the total time, when the protocol generates little amount of data that will be sent via a low latency and constant bandwidth network. The impact of the sizes of the variables on the delay of the protocol is less than the number of rounds’, as increase in the number of rounds leads to increase in computation and encoding time.

All of these aspects can be combined by testing network latency with ping, computing resources with a benchmark, such as “openssl speed -evp aes-256-cbc”, and/or building a table of sizes of s and number of rounds to choose among a set for the special case the prover is in. The level of security and the performance expectations can be agreed upon before the first run or as profiles to run in different situations. Therefore, a ZK protocol such as Fiat-Shamir can be made adaptive in terms of performance by assessing network conditions and computing resources of a prover’s machine, and also the critical time that the prover has to conform in the scenario.

In the future, we would like to automate performance adaptation of the Fiat-Shamir identification protocol by optimizing the protocol with respect to the parameters studied in this paper. We also would like to study other ZK protocols in order to support our findings.
References


Paper III
Modelling and Analysing a WSN Secure Aggregation Protocol: A Comparison of Languages and Tool Support

Volkan Cambazoglu, Ramūnas Gutkovas, Johannes Åman Pohjola, and Björn Victor

Department of Information Technology
Uppsala University, Sweden

Abstract

A security protocol promises protection of a significant piece of information while using it for a specific purpose. Here, the protection of the information is vital and a formal verification of the protocol is an essential step towards guaranteeing this protection. In this work, we study a secure aggregation protocol (SHIA) for Wireless Sensor Networks and verify the protocol in three formal modelling tools (Pwb, mCRL2 and ProVerif). The results of formal verification heavily depend on the model specification and the ability of the tools to deal with the model. Among the three tools, there is difference in data representation, communication types and the level of abstraction in order to represent SHIA. ProVerif and mCRL2 are mature and well-established tools, geared respectively towards security and distributed systems; however, their expressiveness constrains modelling SHIA and its security properties. Pwb is an experimental tool developed by the authors; its relative immaturity is offset by its increased expressive power and customisability. This leads to different models of the same protocol, each contributing in different ways to our understanding of SHIA’s security properties.
1 Introduction

A communication protocol consists of a set of rules that defines the type of data that will be exchanged, how this data will be represented in a syntax, the semantics behind it and the type of communication. Data is at the core of a communication protocol and is exchanged in form of messages. The syntax of the communication protocol defines its structure of representing the data and all other necessary control information in a message that will be sent from one side to the other. The semantics of the communication protocol defines the operation to handle a message; thus a receiver can understand what the message includes and what to do with it. For example, in a file transfer protocol, the receiver should be able to figure out which part of the file is received and whether it should wait for more parts or build the file from the parts by observing the control information that comes with the message. The type of the communication defines the reliability of the communication and the communication primitives that are required for the protocol. For instance, a file transfer protocol runs on a reliable point-to-point connection.

Formal verification is a way of certifying the description of such a protocol by using formal methods that are based on mathematical specification. The rules of the protocol (data, syntax, semantics, communication primitives) are expressed with a certain formal specification and, then, the specification is analysed for the properties of the protocol. When the formal specification of a protocol is analysed, intrinsic properties of the protocol can be identified.
clearly. These properties belong to the internal logic and operation of the protocol such as logical requirements that need to be fulfilled to reach a certain state of the protocol. These properties play a crucial role in verifying the correctness of the protocol.

In this work, we formally model and verify a secure aggregation protocol for Wireless Sensor Networks (WSN), the Secure Hierarchical In-Network Aggregation protocol (henceforth SHIA) by Chan et al [14]. In addition to modelling SHIA to verify its security, we also consider suitability of three different languages and tools for this goal. The tools that are used are Psi-Calculi Workbench (Pwb), mCRL2 and ProVerif. We choose Pwb, because it is expressive and customisable, and we aim to develop it further. mCRL2 is well used for modelling and verifying distributed applications, while ProVerif for security applications; however both tools have not been tried for WSN and using broadcast communication. As each of the three tools targets a different application, they do not have a large intersection on representing the model of SHIA. Fundamentally, these tools differ in data representation, communication types and level of abstraction in modelling SHIA. These differences lead to slightly different models of the same protocol, where Pwb comes on top in representing the model of SHIA completely, and mCRL2 and ProVerif come short in representing broadcast communication and recursive data structures in the model. Even after coming up with different ways of covering these shortcomings in mCRL2 and ProVerif, we are unable to obtain interesting results to the queries we made to these tools. These different models and their justifications will be explained in this paper. Considered from the different perspectives of three formal modelling tools, we could not observe any situation that implies SHIA’s insecurity.

The modelling of the SHIA protocol was a significant driver for improving the Pwb tool. The result of this is a solid first-order algebraic specification for the term language of the processes with battle-tested improvements on the interface with SMT solvers, pragmatic trade-offs on improving the constraint generation and solving run-times, and bringing the implementation of communication primitives closer to what is required by the protocol.

Before going into the details of the secure aggregation protocol, we first present a short introduction to the context of the protocol.

1.1 Data Collection in WSN

A Wireless Sensor Network (WSN) serves for a querier, who wants to obtain values from a specific region or area where the WSN is deployed. The WSN consists of resource constrained sensor nodes and at least one base station, which is generally more resource rich than the other nodes and acts as a gateway between the WSN and the querier. In this setting, efficiency is of
high concern and the most expensive operation for a sensor node is transmitting a message. Arranging the WSN in a spanning tree structure and using in-network data aggregation for obtaining values from the WSN are ways of reducing the number of messages exchanged in the network [41]. These features allow to limit the number of neighbours a node has and also the number of messages a node has to transmit. In some cases, the values that are aggregated in the WSN need to be protected such that they are kept confidential or intact or both.

In the next section, we will describe the SHIA protocol and how we modelled it. Then we will introduce the three modelling tools and their languages in Section 3. How the SHIA model is developed in these tools will be explained in Section 4. Our efforts in developing Pwb to improve its applicability and reflect the SHIA model as close as possible will be described in Section 5. Related work will be presented in Section 6. Our conclusions and what can be done for future work will be shared in Section 7.

2 The Secure Hierarchical In-network Aggregation Protocol

SHIA is a secure protocol that aggregates data in a wireless network by creating a hierarchy among sensor nodes. In-network data aggregation is a technique that creates a spanning tree, Figure 1b, based on a network topology, Figure 1a, in order to associate neighbouring nodes that will send data towards the root of the tree and aggregate data on the go. With this technique, each node sends only one message, which contains aggregated data, to its parent in the tree. Aggregation of data can be done in different ways depending on the type, however, the idea is combining multiple data into one. SHIA [14] provides security on top of this technique by introducing authentication and integrity. SHIA achieves these security goals by using commitments which are aggregated over a virtual binary tree structure, Figure 1c, on top of the aggregation tree, thus it has two levels of abstraction from the physical WSN. Overall SHIA has three phases: (1) query dissemination, (2) aggregation commit, and (3) result checking.

1. Query dissemination: One round of the protocol starts with the querier sending a query to all of the sensor nodes to start sensing and aggregation in the WSN. The query is sent via authenticated broadcast [34], which adds to the security of SHIA by allowing identity confirmation.
Figure 1: SHIA organizes the nodes of the WSN, Figure 1a, into a hierarchical binary commitment tree, Figure 1c
2. **Aggregation commit:** Once the query reaches all of the sensor nodes, a binary commitment tree is constructed iteratively so that the sensed values will be aggregated towards the querier via the tree. In the terminology of [14], a *node* is a real wireless sensor node and a *vertex* is an entity with sensing or aggregating role in the binary commitment tree. Each sensing vertex places its value in a label and sends it to the associated aggregating vertex in the commitment tree. An aggregating vertex can be one physical node or can co-exist with a sensing vertex within the same physical node. For example, in Figure 1b, node A is both sensing and aggregating within one physical sensor node, while node B is only sensing. Each aggregating vertex receives labels from its children, aggregates them into one label and sends it to its parent. Here, an aggregated label contains not just the aggregation of two labels but also a commitment to the protocol, which supports the security of SHIA. The aggregation continues until the base station, which creates the label of the root of the tree and passes it to the querier.

3. **Result checking:** After receiving the label of the root of the tree, the querier starts the distributed result checking phase by sending the root label to the whole network. As the label of the root of the tree is disseminated downwards in the tree, each aggregating vertex should also disseminate off-path labels downwards. Off-path labels of a vertex consist of all the labels of siblings from every other branch on the path from the vertex to the root of the tree. For example, in Figure 1c, the off-path labels of vertex $A_s$ are labels of vertices $B$ and $D_a$. When a sensing vertex receives the label of the root of the tree and all of its off-path labels, it checks whether its contribution in the final aggregate is in place or not; therefore, the off-path labels also contribute to the security of SHIA. If the verification succeeds, each sensing vertex sends an okay message to its parent, which also aggregates these messages and send towards the querier. When the querier receives the final okay message that represents the whole network, the querier accepts the final aggregate; otherwise the aggregation fails.

SHIA fulfils its security goal during these phases of the protocol by achieving *optimal security*, which was defined by Chan et al. [14] as follows:

**Definition 1.** *(Definition 2 in [14])* An aggregation algorithm is optimally secure if, by tampering with the aggregation process, an adversary is unable to induce the querier to accept any aggregation result which is not already achievable by direct data injection.

**Definition 2.** *(Definition 1 in [14])* A direct data injection attack occurs when an attacker modifies the data readings reported by the nodes under its direct control, under the constraint that only legal readings in $[0, r]$ are reported.
There are three properties that support the optimal security of SHIA: (1) authenticated broadcast, (2) off-path labels, and (3) boundary checking.

**Authenticated broadcast** originates from the querier and is sent to every node in the network. Each node that receives an authenticated broadcast can check whether the source is the querier or someone else; thus, the nodes take action for query and result-checking only after they make sure that the querier is the one that requests them. If the authenticated broadcast does not exist, an adversary can trick the sensor nodes into sensing and aggregating as if there is a query from the querier or disturb the actual result checking phase.

**Off-path labels** are components that are used in the result checking phase while building the label of the root of the tree in a bottom-up fashion. Each sensing vertex can compute the final aggregate by starting with its label and aggregating each off-path label on the way towards the root of the tree. If and only if all of the labels are the actual ones, then a node should be able to successfully verify that its contribution to the final aggregate is in place. For example, if vertex $B$, in Figure 1c, needs to check that its contribution is included in the final aggregate, it needs to compute the label of vertex $A_a$ by aggregating its own label with the label of vertex $A_s$, and then compute the final aggregate by bringing the labels of vertices $A_a$ and $D_a$ together. If the computed final aggregate is the same as the received final aggregate, then vertex $B$ confirms that its contribution is included correctly in the aggregation.

**Boundary checking** is about setting the minimum and the maximum values that can be sensed and aggregated in the commitment tree, and also making sure that no value is out of those limits. After being queried, each sensing vertex creates a label of $\langle \text{count, value, complement, ID} \rangle$, where count is one, value is the sensed value, complement is $(r - value)$, $r$ is the maximum allowed value and ID is the identifier of the node. The minimum allowed value is zero; thus, the sensed values and their complements have to be non-negative at all times. When an aggregating vertex receives labels from its children, it creates a new label in which the label of the children vertices are aggregated such as

\[
\text{Sensing vertices: } A_s \rightarrow A_a : \langle 1, v_{as}, \bar{v}_{as}, ID_a \rangle \\
B \rightarrow A_a : \langle 1, v_b, \bar{v}_b, ID_b \rangle \\
\text{Aggregating vertex: } A_a \rightarrow BS : \langle 2, v_{aa}, \bar{v}_{aa}, H[N, 2, v_{aa}, \bar{v}_{aa}, A_s, B] \rangle
\]

where the aggregation operation is addition, $v_{aa} = v_{as} + v_b$, $\bar{v}_{aa} = \bar{v}_{as} + \bar{v}_b$, $H$ is a hash function and $N$ is nonce of the round. The hash value in the aggregating vertex’s label is a way of commitment to the protocol since it allows verification of the values afterwards. When the querier receives the
label of the root of the tree, it checks that the final value and complement are non-negative and their sum equals to $n \times r$ where $n$ is the total number of sensor nodes in the network. These boundaries allow to restrict the magnitude of the impact that an adversary can have on the final aggregate.

2.1 Protocol Model

In SHIA, there are two major types of entities: the querier and the sensor nodes. The querier is straightforward and has only one role, which is to communicate with the sensor nodes. However, the sensor nodes are more complicated and their roles need to be outlined clearly here.

2.1.1 Sensor Nodes

First of all, the base station is interpreted as a sensor node, though it may be more resource rich than the rest of the nodes. It conceptually does data aggregation as any other sensor node and forwards the final aggregation to the querier.

The model for sensor nodes is divided into two: (1) LeafVertex and (2) InternalVertex.

1. **LeafVertex** is the process model of leaf vertices in the binary commitment tree of SHIA. These vertices have *sensing* role and need to communicate with an InternalVertex process.

2. **InternalVertex** is the process model of vertices that are located between leaf vertices and the querier in the binary commitment tree of SHIA. These vertices have *aggregation* role and need to carry communication from LeafVertex processes to the Querier process and vice versa.

The protocol model involves the Querier, the LeafVertex and the InternalVertex processes. These processes communicate with each other via channels in which they input (receive) or output (send) their messages. The channels between processes are treated in an abstract way; hence, a channel could represent both an internal channel within a wireless sensor node and the wireless medium that the sensor nodes uses to communicate with each other. We assume that a binary commitment tree for a round of SHIA is built; therefore, we model the interaction between the querier and the sensor node processes over this tree. Each vertex of the tree represents a process and each edge represents a channel that two processes can communicate over it.

In addition to the communication between processes, it is necessary to model specific features of SHIA; such as types of messages, authenticated messages, aggregation of data and secure result-checking.
We tag the channels to differentiate their uses. In particular, we define *Query*, *Verify* and *Offpath* tags to relate to the three phases of the SHIA protocol. The *Query* tag is only used at the beginning of a round of SHIA, when the querier disseminates a nonce for starting the aggregation session to the WSN. When the querier wants to start the result checking phase, it disseminates the final aggregate to the WSN using the *Verify* tag. The verify tag is used from the beginning of the result checking phase until the end of it, so it carries the final aggregate to every vertex in the tree and also the MACs from every vertex to the querier. The *Offpath* tag is used for carrying the exchanged off-path labels between vertices and also not to confuse these labels with the label of the root of the tree as all of these labels take place in the communication during the result checking phase.

Authenticated broadcast is modelled as one hop communication from the querier to all of the vertices via a separate channel. This is an abstraction from the real authenticated broadcast that is suggested in SHIA to be achieved with µTESLA. The main reason for this abstraction is µTESLA is a symmetric key based protocol and every recipient can verify the source of the broadcast, i.e. the querier. Moreover, the number of hops that are necessary to reach all of the vertices should not affect the content of the authenticated broadcast. Therefore, a separate channel represents the symmetric key based authentication between the querier and all the vertices, and one hop communication represents the nature of the broadcast.

The data is aggregated using a recursive label structure. A basic label consists of a sensed value and when two labels are aggregated, there is one resulting label. As a simple example, a label could be either \( \text{Label} (\text{SensedValue}) \) or \( \text{AggrLabel} (\text{LabelLeft}, \text{LabelRight}) \), where \( \text{AggrLabel} \) takes two labels from children vertices in the tree and aggregates them into one label. Therefore, the label of the root of the tree reflects the structure of the tree instead of representing just the final aggregated value.

The result checking phase of SHIA depends on distributed security by aggregating MACs from all leaf vertices. This is modelled similar to the label structure such that a leaf vertex provides \( \text{MAC} (\text{nonce}, \text{key}) \) to its parent and the parent provides \( \text{XOR} (\text{macLeft}, \text{macRight}) \) to its parent. Here, \( \text{nonce} \) is a one time bit sequence for a round of SHIA, \( \text{key} \) is the symmetric key that is specific to each leaf vertex and shared with the querier, and \( \text{XOR} \) represents the exclusive or operation that takes two MACs and outputs one MAC.

**2.2 Attacker Model**

The assumed attacker aims to trick the querier into accepting a different value than the actual aggregate. The attacker might also try to prevent the aggregation from happening as in a denial of service attack; however, we fo-
focus on the case in which the attacker tries stealthy attacks on SHIA. In this case, the importance is on how SHIA manages to run with dishonest participants. The stealthy attacker may have physical control over an arbitrary number of sensor nodes in the network. After taking control of them, the attacker has knowledge of the secret keys of the sensor nodes and can try to tamper with the aggregation. The attacker could also attack by capturing messages, modifying them or even injecting messages, and once again try to disturb the final aggregate.

We model these two types of attackers in the following ways:

- An attacker who has physical access to a sensor node, is modelled as a LeafVertex process with different parameters. The captured sensor node will try to join in the aggregation as every other sensor node and not reveal its captured status. For example, if all the sensor nodes would sense around 20 °C under normal conditions and the attacker would like to lower the aggregate, then the captured node contributes in the aggregation with 0 °C.

- An attacker who captures, modifies and injects messages via the wireless medium, is modelled as a gap in the SHIA model such that one or more sensor nodes are not defined in the network and the attacker decides which message to send to where as whom. For example, an InternalVertex process is missing in the executed model, so the aggregation and the communication, which the missing InternalVertex should have done, happens according to the attacker’s will. In this case, the querier is not aware of absence of the sensor nodes; thus, the attacker targets certain nodes in the network, blocks their communication with other nodes and acts as those nodes. For instance, the attacker targets a node that aggregates in the network, so it will capture the aggregated value and send a false aggregate instead to the expecting parent.

3 Languages and Tools

Modelling and verification of communication protocols require formal languages and tools that can capture the interaction between processes that execute the protocol. The goal of this study is to explore and compare the capabilities and complexity of three formal languages involved in the verification of SHIA. The three tools are Psi-Calculi Workbench (Pwb), mCRL2 and ProVerif. We briefly introduce their capabilities and features that are related to modelling the SHIA protocol in this section. By this way we cover possible ways of representing the data, the operations and the process
interaction that is necessary to reflect the protocol model we envision for SHIA.

The three tools that are used in this study are all based on process algebras, which are prototype specification languages for reactive systems [3]. In these systems, there are processes that communicate and interact with each other. These processes have states and the interaction happens through transitions between different states. Furthermore, the transitions have well defined structural operational semantics and take place around a logic. While based on this (very high level) process theory, the three tools differ in their process algebras, thus they have different operational semantics, logic, data representation and evaluation framework. All of these tools have their strengths, hence they are chosen for this study: Pwb is an expressive and customizable tool to model a large set of protocols, mCRL2 has a well documented and a rich specification language for analysing distributed systems and protocols, and ProVerif is a well known tool for modelling security protocols.

3.1 Psi-Calculi and Pwb

Psi-calculi [10] is a parametric semantic framework based on the pi-calculus [30], adding the possibility to tailor the data language and logic for each application. The framework provides a variety of features, such as lexically scoped local names for resources, communication channels as data, both unicast and broadcast communication [13], and both first- and higher-order communication [33].

The Psi-Calculi Workbench (Pwb) [12] is a generic tool for implementing psi-calculus instances, and for analysing processes in the resulting instances. It has a wider scope than previous works, and also allows experimentation with new process calculi with a relatively low effort. Pwb is parametric so that it provides functionality for bisimulation equivalence checking and symbolic simulation (or execution) of processes in any psi instance, and a base library for implementing new psi-calculi instances. Pwb thus has two types of users: the user analysing systems in an existing instance of the framework, and the instance implementer.

The following is a quick recapitulation of a simplified version of the psi-calculi framework. The full framework features dynamic environments, meaning that the validity of logical formulas may change during the execution of a process; this feature will not be used in the present paper and is hence not presented. For an in-depth introduction with motivations and examples we refer the reader to [10].

A psi-calculus is defined by instantiating two nominal data types and two equivariant operators. Before explaining the parameters of a psi-calculus, it
might be helpful to remind about nominal data types; as names are at the
core of a psi-calculus.

Names $\mathcal{N}$ are atomic elements of a countably infinite set that is ranged
over by $a, b, \ldots, z$. Intuitively, names are the symbols that can be scoped
and be subject to substitution. A nominal set [35, 22] is a set equipped with
a formal notion of what it means to swap names in an element; this leads
to a notion of when a name $a$ occurs in an element $X$, written $a \in n(X)$
(pronounced “$a$ is in the support of $X$”). We write $a \# X$, pronounced “$a$
is fresh for $X$”, for $a \not\in n(X)$, and if $A$ is a set of names we write $A \# X$
to mean $\forall a \in A. a \# X$. In the following $\tilde{a}$ is a finite sequence of names. The empty
sequence is written $\varepsilon$ and the concatenation of $\tilde{a}$ and $\tilde{b}$ is written $\tilde{a}\tilde{b}$. We say
that a function symbol is equivariant if all name swappings distribute over
it.

A nominal data type is a nominal set together with a set of functions
on it. In particular we shall consider substitution functions that substitute
elements for names. If $X$ is an element of a datatype, $\tilde{a}$ is a sequence of
names without duplicates and $\tilde{Y}$ is an equally long sequence of elements of
possibly another datatype, the substitution $X[\tilde{a} := \tilde{Y}]$ is an element of the
same datatype as $X$. The substitution function can be chosen freely, but
must satisfy certain natural laws regarding the treatment of names; it must
be equivariant, and the names $\tilde{a}$ in $X[\tilde{a} := \tilde{Y}]$ must be alpha-convertible as
if they were binding in $X$. See [10] for details.

Definition 3 (Psi-calculus parameters). A psi-calculus requires the two (not
necessarily disjoint) nominal data types: the (data) terms $T$, ranged over by
$M, N$, the conditions $C$, ranged over by $\varphi$, and the two operators:

\[
\leftrightarrow \in T \times T \rightarrow C \quad \text{Channel Equivalence}
\]
\[
\vdash \in C \rightarrow \{\text{true, false}\} \quad \text{Entailment}
\]

Channel equivalence will be written in infix. Thus, $M \leftrightarrow N$ is a con-
dition, pronounced “$M$ and $N$ are channel equivalent”. We write $\vdash \varphi$ for
$\vdash (\varphi) = \text{true}.$

We impose certain requisites on the sets and operators: channel equiv-
alence must be symmetric and transitive, and substitution $M[\tilde{a} := \tilde{T}]$ on
terms must be such that if the names $\tilde{a}$ are in the support of $M$, the support
of $\tilde{T}$ must be in the support of $M[\tilde{a} := \tilde{T}]$.

We assume a given set of process identifiers, ranged over by $A$, and
assume that each process identifier is associated with a clause $A(\tilde{x}) \leftarrow P$,
where $\tilde{x}$ binds into $P$ and $n(P) = \tilde{x}$.

Definition 4 (Psi-calculus agents). Given a psi-calculus with parameters
as in Definition 3, the agents, ranged over by $P, Q, \ldots$, are of the following
forms.

88
\[0\]
\[\overline{MN}.P\]
\[\overline{M(\lambda\bar{x})N}.P\]
\[M \rhd N\]
\[M \vdash N\]
\[\text{case } \varphi_1 : P_1 \ldots \varphi_n : P_n\]
\[(\nu a)P\]
\[P|Q\]
\[\text{run } A(\overline{M})\]

Output
Input
Broadcast Out
Broadcast In
Case
Restriction
Parallel
Invocation

Restriction \((\nu a)P\) binds a in P and input \(\overline{M(\lambda\bar{x})N}\). P binds \(\bar{x}\) in both N and P. An occurrence of a subterm in an agent is guarded if it is a proper sub-term of an input or output term. An agent is well-formed if in all \(\overline{M(\lambda\bar{x})N}\). P it holds that \(\bar{x} \subseteq \text{u}(N)\) is a sequence without duplicates, and in all invocations \(\text{run } A(\overline{M})\mid \overline{M}\) matches the arity of the clause associated with A.

The agent \(\text{case } \varphi_1 : P_1 \ldots \varphi_n : P_n\) is sometimes abbreviated as \(\text{case } \bar{\varphi} : P\). We sometimes write \(\overline{M(x)P}\) for \(\overline{M(\lambda x)xP}\). From this point on, we only consider well-formed agents.

The actions ranged over by \(\alpha, \beta\) are of the following three kinds: Output \(\overline{M(\nu\bar{a})N}\), input \(\overline{MN}\), and silent \(\tau\). Here we refer to \(M\) as the subject and \(N\) as the object. We define \(\text{bn}(\overline{M(\nu\bar{a})N}) = \bar{a}\), and \(\text{bn}(\alpha) = \emptyset\) if \(\alpha\) is an input or \(\tau\). As in the pi-calculus, the output \(\overline{M(\nu\bar{a})N}\) represents an action sending \(N\) along \(M\) and opening the scopes of the names \(\bar{a}\).

\textbf{Definition 5} (Transitions). A transition is written \(P \xrightarrow{\alpha} P'\), meaning that \(P\) can do \(\alpha\) to become \(P'\). The transitions are defined inductively in Tables 1 and 2.

We identify alpha-equivalent agents and transitions. In a transition the names in \(\text{bn}(\alpha)\) bind into both the action object and the derivative, therefore \(\text{bn}(\alpha)\) is in the support of \(\alpha\) but not in the support of the transition.
In \[ \vdash K \leftrightarrow M \]
\[ M(\lambda \tilde{y})N . P \xrightarrow{K \{ \tilde{y} := \tilde{L} \}} P[\tilde{y} := \tilde{L}] \]

Out \[ \vdash M \leftrightarrow K \]
\[ M N . P \xrightarrow{K N} P \]

Case \[ P_i \xrightarrow{\alpha} P' \]
\[ \text{case } \tilde{\varphi}_i : \tilde{P} \xrightarrow{\alpha} P' \]

Par \[ P \xrightarrow{\alpha} P' \]
\[ P \mid Q \xrightarrow{\alpha} P' \mid Q \]

Com \[ \vdash M \leftrightarrow K \]
\[ P \xrightarrow{M(\nu \tilde{a})N} P' \quad Q \xrightarrow{K N} Q' \]
\[ P \mid Q \xrightarrow{\tau} (\nu \tilde{a})(P' \mid Q') \]

Inv \[ P[\tilde{x} := \tilde{M}] \xrightarrow{\alpha} P' \quad \text{run } A(\tilde{M}) \xrightarrow{\alpha} P' \]

Scope \[ P \xrightarrow{\alpha} P' \]
\[ (\nu b)P \xrightarrow{\alpha} (\nu b)P' \]

Open \[ P \xrightarrow{M(\nu \tilde{a})N} P' \]
\[ (\nu b)P \xrightarrow{M(\nu \tilde{a} \cup \{b\})N} P' \]
\[ b \# \tilde{a}, M \quad b \in \text{n}(N) \]

BrOut \[ \Psi \vdash M \not\vdash K \]
\[ \Psi \not\vdash M N . P \xrightarrow{K N} P \]

Table 1: Structural operational semantics where \( K, L, M \) and \( N \) are terms; \( P, P', Q \) and \( Q' \) are agents. Symmetric versions of Com and Par are elided. In Open the expression \( \tilde{a} \cup \{b\} \) means the sequence \( \tilde{a} \) with \( b \) inserted anywhere. A symmetric version of BrCom is elided. In rules BrCom and BrMerge we assume that \( F(P) = (\nu b_P)\Psi_P \) and \( F(Q) = (\nu b_Q)\Psi_Q \) where \( \tilde{b}_P \) is fresh for \( P, \tilde{b}_Q, Q, K \) and \( \Psi \), and that \( \tilde{b}_Q \) is fresh for \( Q, \tilde{b}_P, P, K \) and \( \Psi \).
Table 2: Structural operational semantics where $K$, $L$, $M$ and $N$ are terms; $P$, $P'$, $Q$ and $Q'$ are agents. Symmetric versions of Com and Par are elided. In Open the expression $\tilde{a} \cup \{b\}$ means the sequence $\tilde{a}$ with $b$ inserted anywhere. A symmetric version of BrCom is elided. In rules BrCom and BrMerge we assume that $F(P) = (\nu bP)\Psi_P$ and $F(Q) = (\nu bQ)\Psi_Q$ where $\tilde{b}_P$ is fresh for $P, \tilde{b}_Q, Q, K$ and $\Psi$, and that $\tilde{b}_Q$ is fresh for $Q, \tilde{b}_P, P, K$ and $\Psi$.

3.2 Algebra of Communicating Processes and mCRL2

The mCRL2 language [17] is a process description language and a toolset for formal modelling. The language is based on the Algebra of Communicating Processes (ACP) [6] extended with higher-order datatypes and equation systems on data. The following presentation is based on [23].

The observable action in mCRL2 is that of finite set of multi-actions of the form $\{a_1 \mid \ldots \mid a_n\}$ where $a_i$ is are arbitrary action defined by the user. We write $\alpha$ for multi-actions. The intuition here that multiple actions are observable simultaneously. Then, we can define the mCRL2 language.
Definition 6. The mCRL2 processes are defined as follows.

\[
P, Q ::= P \parallel Q \quad \text{parallel composition/merge} \\
P | Q \quad \text{synchronisation} \\
P + Q \quad \text{choice} \\
P \cdot Q \quad \text{sequence} \\
\Sigma_{x \in D} P \quad \text{summation} \\
\alpha \quad \text{multi action} \\
\delta \quad \text{deadlock/inaction} \\
c \to P \circ Q \quad \text{if/then/else} \\
X(x_1, \ldots, x_n) \quad \text{process reference} \\
\text{comm}_{C}P \quad \text{communication operator} \\
\text{hide}_{H}P \quad \text{hiding operator} \\
\text{allow}_{A}P \quad \text{restriction operator/allow} \\
\text{block}_{B}P \quad \text{block operator}
\]

where \( H, A \) and \( B \) are finite sets of multi-actions and \( C \) is multi-action to multi-action transformation rule given as a finite set of the form \( C = \{a_{11} \mid \ldots \mid a_{1m_1} \to a_{1}', \ldots, a_{n1} \mid \ldots \mid a_{nm_n} \to a_n' \} \).

The mCRL2 language is slightly richer: we here give syntax for the language fragment without time as we had not used this fragment in this report.

The semantics for mCRL2 process are defined using equations on the processes that are reminiscent that of algebraic theories, e.g., that of \( \lambda \)-calculus. The semantics would not surprise the reader familiar with basic process algebra such as CCS.

Most of the semantics of the operators are quite standard and self-explanatory. The parallel composition is defined in terms of synchronisation and choice, e.g., the process \( a \parallel b \) is defined as \( a \cdot b + b \cdot a + a \mid b \) where \( a \) and \( b \) are actions; so that the process reveals either \( a \), \( b \), or \( a \mid b \) multi-actions. The synchronisation operator is defined to merge multi-actions into multi-actions recursively in a process. The choice operator + is the standard non-deterministic choice. The infinite choice \( \Sigma_{x \in D} P \) parameterises the process \( P \) over the domain \( D \); this operator is useful for modelling message reception where \( D \) represents all the possible input values.

The specific language features of mCRL2 are the \text{comm}, \text{hide}, \text{allow}, and \text{block}. The \text{comm}_{C}P operators fuses multi-actions found in \( P \) according to the rules in \( C \). The \text{hide}_{H}P operator as the name suggests hides the action found in \( H \) by emitting the silent action \( \tau \) instead. The \text{block}_{B}P operator simply sends the process into the deadlock process if one of the multi-actions are emitted by \( P \) are in \( B \). Finally, the allow operator only allows the specified multi-actions to be observed.
3.3 The Applied pi calculus and ProVerif

The applied pi calculus extends the standard pi-calculus so that data structures rather than names can be transmitted on channels. The data structures may be equipped with equations, and pattern matched against. ProVerif is a tool for verification of cryptographic protocols that supports a subset of applied $\pi$ as input language. Details on ProVerif can be found in [11]; applied $\pi$ was first introduced in [2].

The supported subset of applied pi is the following:

**Definition 7.** The terms are as follows:

$$M, N ::= a, b, c, k, m, n, s \ 	ext{name}$$
$$x, y, z \ 	ext{variables}$$
$$(M_1, \ldots, M_n) \ 	ext{tuple}$$
$$h(M_1, \ldots, M_k) \ 	ext{constructor/destructor application}$$
$$M = N \ 	ext{equality test}$$
$$M \langle N \ 	ext{inequality test}$$
$$M \& N \ 	ext{conjunction}$$
$$M || N \ 	ext{disjunction}$$
$$\text{not}(M) \ 	ext{negation}$$

The processes are as follows:

$$P, Q ::= P | Q \ 	ext{parallel}$$
$$! P \ 	ext{replication}$$
$$\text{new} n : t; P \ 	ext{restriction}$$
$$\text{in}(M, x : t); P \ 	ext{input}$$
$$\text{out}(M, N); P \ 	ext{output}$$
$$\text{if} M \ 	ext{then} P \ 	ext{else} Q \ 	ext{conditional}$$
$$\text{let} x = M \ 	ext{in} P \ 	ext{else} Q \ 	ext{evaluation}$$
$$R(M_1, \ldots, M_n) \ 	ext{macro}$$
$$0 \ 	ext{nil}$$

where $t$ ranges over types.

Note that unlike psi-calculus, communication and pattern matching are separate. Since communication is monadic, we must use the **let** construct to obtain the arguments of composite terms using destructors. A typical example is a process that expects a message encrypted with key $k$, and upon receipt tries to decrypt it using the **dec** destructor:

$$\text{in}(M, x : t); \ 	ext{let} y = \text{dec}(x, k) \ 	ext{in} P \ 	ext{else} Q$$

When some term is received on $M$, the process will proceed as $P[y := N]$ in the event of successful decryption, where $N$ is the cleartext of the message.
Otherwise it proceeds as $Q$; this happens if $x$ is not an encrypted message or if $k$ is the wrong decryption key. Note that the setting is parameterised on arbitrary constructors and destructors, so as users we are free to define our own.

The semantics of the other operators would be unlikely to surprise the reader.

4 Modelling and Verifying SHIA

In this section, we present how the SHIA model from Section 2.1 is actually written in Pwb, mCRL2 and ProVerif. We go through data and process models, explain divergences from the protocol model, and share results that are obtained from each tool.

4.1 Psi-Calculi Workbench

4.1.1 Model

We model the SHIA protocol (Section 2.1) in Pwb in the following ways.

Data Model. We provide the full data model for SHIA in Listing 1. The data model consists of basic data types (Sorts and CSorts), more complex data types (Symbols) that are based on the basic ones and the relations between these data types (Axioms).

Process Model. We provide the process model for SHIA in Listing 2. There are three kinds of processes: Querier, InternalVertex and LeafVertex.

Querier starts the secure aggregation by sending a nonce to all other processes via authenticated broadcast. It receives the label of the root of the tree and starts the result checking by sending the label back via authenticated broadcast. It receives one MAC for all of the leaf vertices, checks that the MAC indeed represents all of the leaf vertices and, depending on the outcome of the check, it either accepts the aggregation or rejects it (lines 1-7).

InternalVertex receives a nonce from Querier via authenticated broadcast that indicates the start of a round of secure aggregation. It receives labels from its children and sends the aggregation of those labels to its parent. It receives the label of the root of the tree via authenticated broadcast that indicates the start of result checking. It swaps labels of children and sends as off-path labels. It receives MACs from its children and sends the XOR of them to its parent (lines 9-19).
LeafVertex receives a nonce from Querier via authenticated broadcast that indicates the start of a round of secure aggregation. It creates a label based on the sensing and sends it to its parent. It receives the label of the root of the tree via authenticated broadcast. It also receives its off-path label from its parent. If it can re-build the label of the root of the tree using its own label and off-path label, then it sends a MAC, which is based on the nonce of the round and the key that is shared with the querier, to its parent (lines 21-27).

System represents an example scenario in which there is a querier and
two sensor nodes, and all of them are honest participants in SHIA protocol. Two sensor nodes are modelled as two leaf vertices and one internal vertex as in the commitment tree; therefore, the leaf vertices do sensing and send labels whereas the internal vertex does aggregation and carries messages between leaf vertices and the querier. All of the processes in the scenario are initialised with appropriate parameters. For example; the Querier process must know the channels, where it should send and receive its messages, the nonce of the round and the final MAC value, which should be produced by collaboration of all the vertices in the tree and also based on the nonce (lines 29-36).

Listing 2: Process model of SHIA in Pwb language for 2 node network.

```pwb
Querier(chQuerier, chBS, nonceQ, macXorAllAuthCode) <=
  "Query(chQuerier)"!<nonceQ>.
  "Query(chBS)"(lblRoot).
  "Verify(chQuerier)"!<lblRoot>.
  "Verify(chBS)"(macAuthCode).
  case "macAuthCode = macXorAllAuthCode" : "Qsuccess"<0>
  [] "not(macAuthCode = macXorAllAuthCode)" : 0;

InternalVertex(chQuerier, chParent, chLeft, chRight) <=
  "Query(chQuerier)"?(nonceQ).
  "Query(chLeft)"(lblLeft).
  "Query(chRight)"(lblRight).
  "Verify(chQuerier)"?(lblRoot).
  "Offpath(chLeft)"<lblRight>.
  "Offpath(chRight)"<lblLeft>.
  "Verify(chLeft)"(macLeft).
  "Verify(chRight)"(macRight).
  "Verify(chParent)"<"XOR(macLeft, macRight)";

LeafVertex(chQuerier, chParent, keyK, iSensedValue) <=
  "Query(chQuerier)"?(nonceQ).
  "Query(chParent)"<"Label(iSensedValue)".
  "Verify(chQuerier)"?(lblRoot).
  "Offpath(chParent)"(lblOffpath).
  case "AggrLabel(Label(iSensedValue), lblOffpath) = lblRoot" :
    "Verify(chParent)"<"MAC(nonceQ, keyK)";

System(chQuerier) <= (new chBS, chLeft, chRight, keyLeft, keyRight)
  (new nonceQ) (Querier<chQuerier, chBS, nonceQ, "XOR(MAC(nonceQ, keyLeft), MAC(nonceQ, keyRight))") |
  InternalVertex<chQuerier, keyK, MAC(nonceQ, keyRight)>
  LeafVertex<chQuerier, chLeft, keyLeft, 20> |
  LeafVertex<chQuerier, chRight, keyRight, 22>
);```

96
4.1.2 Verification

SHIA protocol is verified both manually and automatically by using Pwb. Manual verification is step-wise execution of the protocol model, where each step is an observable transition that takes place in the state space of the protocol. This verification can also be done automatically, hence we are able to draw state diagrams of the protocol for different scenarios in an automated way. An example state diagram is shown in Figure 2. The diagram belongs to a system where there are two nodes and an attacker plays man-in-the-middle between the nodes and the querier. After running the process model in the Pwb, the diagram is generated for taking transitions only for a depth of ten. The text in the figure is not visible as the diagram does not fit in this page format. The ellipses represent states and the arrows represent the transitions. The text in an ellipse or next to a transition defines the constraints that need to be fulfilled to reach the state or to take the transition.

Figure 2: A state diagram from a network of two nodes with an attacker
4.1.3 Results

SHIA protocol runs correctly as modelled. The results for a small scenario with a few sensor nodes can be obtained in terms of seconds. When the network size increases, the time necessary to evaluate the model increases exponentially, i.e. from seconds to hours. Here are some example timings in Table 3.

<table>
<thead>
<tr>
<th>Network Size</th>
<th>Evaluation Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3.67</td>
</tr>
<tr>
<td>4</td>
<td>10.56</td>
</tr>
<tr>
<td>8</td>
<td>9266.33</td>
</tr>
</tbody>
</table>

Table 3: Pwb timing results for evaluating the SHIA protocol model
4.2 mCRL2

4.2.1 Model

Listing 3: Data model of SHIA in mCRL2 language

```plaintext
sort Nonce = struct nonce(n: Int) ;
sort PubPrivKeyPair, PubKey, PrivKey;
map genKeyPair : Int -> PubPrivKeyPair;
map pubkey : PubPrivKeyPair -> PubKey;
map privkey : PubPrivKeyPair -> PrivKey;
sort Signature = struct sg1 | sg2 | sg3 | sg4 | sg5;
map signNonce : Nonce # PrivKey -> Signature;
map signLabel : Label # PrivKey -> Signature;
map verifNonceSig : Nonce # Signature # PubKey -> Bool;
var n1, n2 : Nonce, pk1, pk2 : PubPrivKeyPair;
eqn (n1 == n2 && pk1 == pk2)
    -> verifNonceSig(n1, signNonce(n2, privkey(pk1)),
        pubkey(pk2)) = true;
    (!n1 || !n2 || !pk1 || !pk2)
    -> verifNonceSig(n1, signNonce(n2, privkey(pk1)),
        pubkey(pk2)) = false;
map verifLabelSig : Label # Signature # PubKey -> Bool;
var n1, n2 : Label, pk1, pk2 : PubPrivKeyPair;
eqn (n1 == n2 && pk1 == pk2)
    -> verifLabelSig(n1, signLabel(n2, privkey(pk1)),
        pubkey(pk2)) = true;
    (!n1 || !n2 || !pk1 || !pk2)
    -> verifLabelSig(n1, signLabel(n2, privkey(pk1)),
        pubkey(pk2)) = false;
sort Key;
map key : Int -> Key;
sort MACode;
map maCode : Nonce # Key -> MACode;
sort Digest = Bag(MACode);
map XOR : Digest # Digest -> Digest;
var a, b : Digest;
eqn XOR(a,b) = a + b;
map MAC : Nonce # Key -> Digest;
var n, k : Key;
eqn MAC(n,k) = (maCode(n,k) : 1);
sort SensedValues = Int;
sort Label = Bag(SensedValues);
map AggrLabel : Label # Label -> Label;
var left, right : Label;
eqn AggrLabel(left, right) = left + right;
sort ValueLabel : Int -> Label;
var value : Int;
eqn ValueLabel(value) = {value : 1};
sort Chan = struct c(channelId: Int) ;
act sendQuery, recvQuery, tauQuery : Chan # Nonce # Signature;
act sendQueryLabel, recvQueryLabel, tauQueryLabel : Chan # Label;
act sendVerify, recvVerify, tauVerify : Chan # Label # Signature;
act sendVerifyCode, recvVerifyCode, tauVerifyCode : Chan # Digest;
act sendOffpath, recvOffpath, tauOffpath : Chan # Label;
act Q_SUCCESS, Q_FAIL, NODE_FAIL;
```

99
We model the SHIA protocol (Section 2.1) in *mCRL2* in the following ways.

**Data Model.** We provide the full data model for SHIA in Listing 3.

The messages originating from the base station in SHIA need to be authenticated. We model authenticity of a message by using verification of digital signatures with asymmetric key cryptography (lines 3-25). This is needed for both nonces and the final label that the base station receives. Since the *mCRL2* maps are monomorphic, we need to provide two separate maps and equations for nonce and label signature verification. In the process model, the nodes would have the public key of the base station. The digital signature verification is easily expressed equationally in *mCRL2* (lines 14-18, and 20-25). The reason why we use digital signatures and not message authentication codes to obtain authenticity of nonces and labels is that these messages are part of a command that is broadcasted to a network (lines 55 and 57). That is, if we used MACs with symmetric keys, it would be sufficient for an attacker to compromise a single node and obtain the shared key to be able to forge authenticated broadcasts for sending request to a network.

Message Authentication Codes are modelled abstractly as an indeterminate map from nonces and keys to types for MACs (lines 27-31). *mCRL2* allows for equational specification on data; however, the tools treat those specification as term rewriting systems (where the left hand side of equations may be rewritten to the right hand side). We cannot directly model the commutativity of XOR operation, i.e. \( \text{XOR}(a, b) = \text{XOR}(b, a) \); however, we are able to express the commutativity of XOR by modelling it as an operation on finite multisets\(^1\) (called bags in *mCRL2*) of message authentication codes. This is the reason why we introduce the type \( \text{Digest} \) and the construction of a MAC is an injection to \( \text{Digest} \) (lines 32-40). We use the same technique to model label aggregation (lines 42-51).

We model channels as a separate sort with a single injective operation of sort integer (line 53). We define distinct actions for each phase of the protocol: broadcast of query, aggregation of labels from nodes, broadcast of verification of a label, and distribution of off-path labels. For each of these phases, we define three actions for sending and receiving ends of a channel, and an action to indicate communication (lines 55-59).

---

\(^1\)XOR over MACs and multiset union over MACs are the same kind of algebraic structures that are associative, commutative, and closed.
Listing 4: Process model of SHIA in mCRL2 language for 2 node network.

```plaintext
proc Querier(bs : Chan, non: Nonce, 
    pk: PubPrivKeyPair, authCode: Digest) = 
    sendQuery(bs, non, signNonce(non, privkey(pk))) . 
    sum rootLabel : Label . recvQueryLabel(bs, rootLabel) . 
    sendVerify(bs, rootLabel, signLabel(rootLabel, privkey(pk))) . 
    sum code: Digest. recvVerifyCode(bs, code) . 
    ((authCode == code) -> Q_SUCCESS 
    <> Q_FAIL )
); 

proc ForwardNonce(n: Nonce, s: Signature, 
    p: Chan, l: Chan, r: Chan) = 
    recvQuery(p, n, s). 
    sendQuery(l, n, s). 
    sendQuery(r, n, s); 

proc ForwardRootLabel(lbl : Label, s : Signature, 
    p: Chan, l: Chan, r: Chan) = 
    recvVerify(p, lbl, s). 
    sendVerify(l, lbl, s). 
    sendVerify(r, lbl, s); 

proc InternalVertex(parent : Chan, left : Chan, right : Chan) = 
    sum n : Nonce, s : Signature. 
    ForwardNonce(n, s, parent, left, right). 
    sum leftLabel : Label . recvQueryLabel(left, leftLabel) . 
    sum rightLabel : Label . recvQueryLabel(right, rightLabel) . 
    sendQueryLabel(parent, AggrLabel(leftLabel, rightLabel)) . 
    sum rootLabel : Label , ss : Signature. 
    ForwardRootLabel(rootLabel, ss, parent, left, right). 
    sendOffpath(left, rightLabel). 
    sendOffpath(right, leftLabel). 
    sum leftCode : Digest. recvVerifyCode(left, leftCode). 
    sum rightCode : Digest. recvVerifyCode(right, rightCode). 
    sendVerifyCode(parent, XOR(leftCode, rightCode)); 

proc LeafVertex(parent : Chan, k: Key, pk: PubKey, sensedValue : Int) = 
    sum n : Nonce, s : Signature . recvQuery(parent, n, s). 
    !verifNonceSig(n, s, pk) -> NODE_FAIL <> 
    sendQueryLabel(parent, ValueLabel(sensedValue)) . 
    sum rootLabel : Label, ls: 
        Signature . recvVerify(parent, rootLabel, ls). 
    !verifLabelSig(rootLabel, ls, pk) -> NODE_FAIL <> 
    sum offpathLabel: Label. recvOffpath(parent, offpathLabel). 
    ((AggrLabel(ValueLabel(sensedValue), offpathLabel) == rootLabel ) 
    -> sendVerifyCode(parent, MAC(n, k)) 
    <> NODE_FAIL ); 
```

Process Model. We provide a mCRL2 process model for a SHIA network in Listing 4. For the sake of simplicity, it is for a two node network. It is straightforward to extend it to bigger networks (the complete model can be obtained at [1].

101
Each vertex in the commitment tree is modelled as a separate process — even the internal ones.

The querier works as expected: it first issues a query command with signed nonce, and then waits for the full label to arrive from the network. It requests verification of the label from the network by signing the label. Ultimately, if the received verification code from the network does not match the expected it signals a failure, otherwise success (lines 1-9).

SHIA relies on the messages broadcasted from the base station to be authenticated. Since \texttt{mCRL2} language does not have primitives for broadcast communication, we instead model message broadcast as point-to-point forwarding of messages by the intermediate vertices to their left and right child vertex in a commitment tree (lines 24, 29, and process definitions are on lines 11 and 17).

The leaf processes check for the authenticity of messages that originate from the base station and signal a failure (lines 39 and 42) whenever the check fails. The vertices also signal failure with the same action whenever they detect that their values were not accounted in the final label.

The keys are pre-distributed and this is done by simply initialising the parameters of the processes. The keys and nonces need to be concretely constructed as terms. We also provide the expected verification result. \texttt{mCRL2} does not have primitives for generation of arbitrary values as creation of new names in psi-calculi. The following is an example of two node network initialisation.

```plaintext
proc System =
  Querier(c(0), nonce(0), genKeyPair(0),
    XOR(MAC(nonce(0), key(0)), MAC(nonce(0), key(1)))) ||
  InternalVertex(c(0), c(1), c(2)) ||
  LeafVertex(c(1), key(0), pubkey(genKeyPair(0)), 20) ||
  LeafVertex(c(2), key(1), pubkey(genKeyPair(0)), 22) ;
```

The following completes the process model of SHIA for the two node network. We only allow communication actions, queries success and failures, and node failure actions. The \texttt{comm} renames the multi-actions with different polarities to our communication actions.
4.2.2 Divergence from the Protocol Model

In some ways the model in mCRL2 is more concrete than the abstract model (Section 2.1). The abstract model does not specify how the messages are authenticated. Chan et al. [14] do not commit to a particular protocol and state that SHIA is independent of this choice, while we needed to commit to a particular method to ensure authenticity. We used digital signatures based on asymmetric key cryptography. While the asymmetric key cryptography is straightforward to express equationally, it may be unrealistic as asymmetric cryptography is computationally expensive and protocols like $\mu$Tesla might be used instead. Furthermore, we have used point-to-point communication to model broadcast. This choice makes the internal vertices exhibit more behaviour than in the abstract model, while in abstract model only the leaf vertices (viz., the physical nodes) would exhibit broadcast communication.

We could not state XOR and label aggregation operations equationally (abstractly), although mCRL2 language has sufficient built-in data structures that allows us to model them in this case.

4.2.3 Verification

Unfortunately, the mCRL2 toolset uses only explicit state enumeration to generate state transitions and, what is more, it is not possible to associate equations on the traces of actions. Thus, we are forced to model the attacker explicitly, that is, as a process. This is a very weak attacker model that relies on the ingenuity of the model writer to cover all of the cases.

We then use the mCRL2 model checker on specified properties in a very powerful modal logic, modal $\mu$-calculus (the mCRL2 parametric boolean equation system). We check that in the attacker’s presence, the querier eventually rejects the aggregate value. Similarly, without the attacker the system eventually accepts the aggregate value.

For example, in Listing 5 an internal vertex that has been captured ignores the labels received from the children and injects its own sensed value label (line 6). If we use the captured internal vertex instead of internal one in the network, then the following property is attested by the mCRL2 model.
checker (\texttt{pbes2bool}), that is, a node eventually fails since it detects that its value was not contributed to the final label.

\[
\langle \texttt{true}\rangle \langle \texttt{NODE\_FAIL}\rangle \texttt{true}
\]

In the network absent of malicious nodes, the following is true, that the querier always eventually accepts the final aggregate, that is, from any state there is a path ending with \texttt{Q\_SUCCESS} action.

\[
[\texttt{true}] \langle \texttt{true}\rangle \langle \texttt{Q\_SUCCESS}\rangle \texttt{true}
\]

Listing 5: Captured node in two node network

\begin{verbatim}
1 proc CapturedInternalVertex1(parent : Chan, left : Chan, right : Chan) =
2    sum n : Nonce, s : Signature.
3       ForwardNonce(n, s, parent, left, right).
4    sum leftLabel : Label . recvQueryLabel(left, leftLabel).
5    sum rightLabel : Label . recvQueryLabel(right, rightLabel).
6       sendQueryLabel(parent, ValueLabel(999999)).
7    sum rootLabel : Label, ss : Signature.
8       ForwardRootLabel(rootLabel, ss, parent, left, right).
9       sendOffpath(left, rightLabel).
10      sendOffpath(right, leftLabel).
11    sum leftCode : Digest. recvVerifyCode(left, leftCode).
12    sum rightCode : Digest. recvVerifyCode(right, rightCode).
13       sendVerifyCode(parent, XOR(leftCode, rightCode));
\end{verbatim}

4.2.4 Usability and Experience

The \texttt{mCRL2} toolset is mature and well documented. It is fairly straightforward to model with efficient tools for model checking and simulation. The major shortcoming is that the transition model is overly concrete, thus it is hard to capture more abstract features as discussed in the previous sections. This also has a practical implication: the simulator of \texttt{mCRL2} processes works on the intermediate language, the linearised process, that is syntactically far removed from the original and is essentially unreadable. This makes debugging even fairly small models such as this one a very daunting task.

4.2.5 Results

The largest network that we considered in \texttt{mCRL2} is 8 node network. The linearised process generation is very fast, and the model checking takes around 30 min per formula on a system with dual core at 1.7 GHz clock speed and 4 GB of RAM.

4.3 ProVerif

As always, the question of how to model authenticated broadcast from the base station presents itself. The problem is twofold: we must model broadcast, and we must model authentication.
Unlike Pwb there is the lack of a broadcast communication primitive; hence we must model broadcast communication using only unicast communication. It is impossible to obtain a “good” encoding in the general case when the number of nodes is unknown [21]; fortunately we are modelling a scenario in which the topology of the network is known, so we can come up with a protocol-specific encoding.

As for authentication, the original presentation of SHIA abstracts away from the particular mechanism used to do this; several techniques are out there [42, 25, 34] and it is always nice to offer the implementer some freedom and not bog down the presentation.

We offer two different models of SHIA in ProVerif, corresponding to different ways of modelling authentication. The first model attempts to stay true to the philosophy of abstracting away from SHIA, while the second model deviates from it by committing to a public/private-key signature scheme.

4.3.1 Model 1

The first model uses a sequence of unicast messages on a private channel br\text{Auth} to model authenticated broadcasts (it is declared \texttt{private} in line 9, meaning that the attacker cannot partake). Other than that, the deviations from the Pwb model are mostly syntactic; the sequentialisation in particular follows the same style.

Security guarantees are formulated as relations between events. Events may be explicitly signalled when certain states of a process have been reached, and are useful for using ProVerif to reason about reachability and safety. Lines 16-18 define the three events we use: Success, which is signalled when the querier believes the aggregation result is valid; ActualSuccess, which is signalled when regardless of the querier’s beliefs, the aggregation result is actually valid; and Node\text{OK}(\text{mkey}), meaning that the node with private key Node\text{OK}(\text{mkey}) accepts the aggregation result.

Node\text{OK} events are triggered by the respective nodes when they send out MACs (lines 88 and 99). The purpose of also having the event is to be able to distinguish situations where an attacker was somehow able to spoof the MAC without any interaction with the node in question. Success and ActualSuccess are triggered in the querier process in lines 60-70. First, if the received label is equal to a label constructed from the actual sensed values of the nodes, we signal ActualSuccess. Then, if the querier is able to verify the MAC, we signal Success. Note that in reality, the querier of course cannot construct a label from the actual sensed values of the nodes since it has no knowledge of them. This unrealism is not a problem since our process has the form:
if unrealistic test then event $E; P$ else $P$

In other words, the continuation is the same regardless of the outcome of the unrealistic tests, modulo raising the event $E$. However, the event does not influence the subsequent behaviour of $P$ and is not observable by any other process; it is only used in the verification stage.

We use events to construct seven queries to ask ProVerif (lines 24-47). The first four queries are sanity checks, to verify that states where the various sets are satisfied are indeed reachable. The remaining three queries correspond to the correct operation of SHIA. They test that if the querier is able to verify the MAC, then all nodes are happy with the label; and most importantly, that whenever the querier is able to verify the MAC, it must be the case that the label has not been tampered with.

Listing 6: ProVerif model with private channel

```plaintext
1 type nonce.
2 type mkey.
3
4 free mynonce : nonce. (* note: nonce is not [private] *)
5 free chBS, chLeft, chRight : channel.
6 free leftvalue : bitstring.
7 free rightvalue : bitstring.
8
9 free brAuth : channel [private].
10
11 (* Cryptographic primitives for symmetric key cryptography *)
12 fun mac(nonce, mkey) : bitstring.
13 fun AggrLabel(bitstring, bitstring) : bitstring.
14 fun XOR(bitstring, bitstring) : bitstring.
15
16 event Success. (* the querier receives a correct MAC *)
17 event ActualSuccess. (* the querier receives a correct label *)
18 event NodeOK(mkey).
19 (* the node with private key accepts the aggregation *)
20
21 (* Security assumptions: *)
22 (* the attacker cannot learn the private keys of X and Y. *)
23 not attacker(new leftkey).
24 not attacker(new rightkey).
25
26 (* Ask ProVerif if... *)
27
28 (* 1. Reachability of successful state. *)
29 query event(ActualSuccess).
30 query event(Success).
31
32 (* 2. Reachability of node acceptance. *)
33 query event(NodeOK(new leftkey)).
34 query event(NodeOK(new rightkey)).
35
36 (* 3. Whenever the querier accepts the MAC, the nodes
37 have previously accepted the label
38 *)
```
query event(Success) =⇒ event(NodeOK(new leftkey)).

query event(Success) =⇒ event(NodeOK(new rightkey)).

(* 4. If the querier accepts the MAC, the querier received a correct label. *)

query event(Success) =⇒ event(ActualSuccess).

(* Definitions of protocol participants. *)

let Querier(leftkey : mkey, rightkey : mkey) =
  out(brAuth, mynonce);
  out(brAuth, mynonce);
  in(chBS, lblRoot : bitstring);
  out(brAuth, lblRoot);
  out(brAuth, lblRoot);
  in(chBS, macAuthCode : bitstring);
  if lblRoot = AggrLabel(leftvalue, rightvalue) then
    event ActualSuccess;
    if macAuthCode = XOR(mac(mynonce, leftkey), mac(mynonce, rightkey))
    then
      event Success
    else
      0
  else
    if macAuthCode = XOR(mac(mynonce, leftkey), mac(mynonce, rightkey))
    then
      event Success
    else
      0.

let NodeInternal() =
  in(chLeft, lblLeft : bitstring);
  in(chRight, lblRight : bitstring);
  out(chBS, AggrLabel(lblLeft, lblRight));
  out(chLeft, lblRight);
  out(chRight, lblLeft);
  in(chLeft, macLeft : bitstring);
  in(chRight, macRight : bitstring);
  out(chBS, XOR(macLeft, macRight)).

let NodeLeafLeft(leftkey : mkey, leftvalue : bitstring) =
  in(brAuth, nnonce : nonce);
  out(chLeft, leftvalue);
  in(brAuth, lblRoot : bitstring);
  in(chLeft, lblOffpath : bitstring);
  if AggrLabel(leftvalue, lblOffpath) = lblRoot then
    event NodeOK(leftkey);
    out(chLeft, mac(nnonce, leftkey))
  else
    0.

let NodeLeafRight(rightkey : mkey, rightvalue : bitstring) =
  in(brAuth, nnonce : nonce);
  out(chRight, rightvalue);
  in(brAuth, lblRoot : bitstring);
4.3.2 Model 2

The second model eschews the private channel approach and instead uses a public/private-key signature scheme to obtain authentication. Removing private channels from consideration yields a possibly stronger attacker model (who can now do Dolev-Yao skulduggery also on nonce propagation transitions), at the price of cluttering the model with encryption and decryption operations. Note that because of ProVerif’s type system it is necessary to have two distinct signing and checking functions for messages and nonces, respectively.

Listing 7: ProVerif model with explicit public/private keys

```plaintext
free chBS, chLeft, chRight, chAttacker : channel.

(* Cryptographic primitives for public-key signatures *)
type sskey .
type spkey .
fun spk ( sskey ) : spkey .
fun sign ( bitstring , sskey ) : bitstring .
reduc forall m: bitstring , k : sskey ; getmess( sign(m,k)) = m.
reduc forall m: bitstring , k: sskey ; checksign( sign(m,k),spk(k)) = m.
fun signn ( nonce , sskey ) : nonce .
reduc forall m: nonce , k : sskey ; getmessn ( signn (m, k )) = m.
reduc forall m: nonce, k: sskey; checksignn( signn(m,k),spk(k)) = m.
free qskey : sskey [private].

let Querier(leftkey : mkey, rightkey : mkey) =
  out(chBS, signn(mynonce, qskey));
in(chBS, lblRoot : bitstring);
out(chBS, sign(lblRoot, qskey));
```

let NodeInternal() =
  in(chBS, nonce : bitstring);
  out(chLeft, nonce);
  out(chRight, nonce);
  in(chLeft, lblLeft : bitstring);
  in(chRight, lblRight : bitstring);
  out(chBS, AggrLabel(lblLeft, lblRight));
  in(chBS, lblRoot : bitstring);
  out(chLeft, lblRoot);
  out(chRight, lblRoot);
  out(chLeft, lblRight);
  out(chRight, lblLeft);
  in(chLeft, macLeft : bitstring);
  in(chRight, macRight : bitstring);
  out(chBS, XOR(macLeft, macRight)).

let NodeLeafLeft(leftkey : mkey, leftvalue : bitstring) =
  in(chLeft, nonce : nonce);
  out(chLeft, leftvalue);
  in(chLeft, lblLeft : bitstring);
  in(chLeft, lblRoot : bitstring);
  if AggrLabel(leftvalue, lblOffpath) = checksign(lblRoot, spk(qskey))
  then
    event NodeOK(leftkey);
    out(chLeft, mac(checksignn(nonce, spk(qskey)), leftkey))
  else
    0.

let NodeLeafRight(rightkey : mkey, rightvalue : bitstring) =
  in(chRight, nonce : nonce);
  out(chRight, rightvalue);
  in(chRight, lblRight : bitstring);
  in(chRight, lblOffpath : bitstring);
  if AggrLabel(lblOffpath, rightvalue) = checksign(lblRoot, spk(qskey))
  then
    event NodeOK(rightkey);
    out(chRight, mac(checksignn(nonce, spk(qskey)), rightkey))
  else
    0.

process new leftkey : mkey;
  new rightkey : mkey;
  out(chAttacker, spk(qskey));
  Querier(leftkey, rightkey) |
### 4.3.3 Result

Unfortunately, ProVerif doesn’t produce any conclusive answers with either model. The output produced is identical in both cases:

RESULT `event(Success)==>event(ActualSuccess)` cannot be proved.
RESULT `event(Success)==>event(NodeOK(rightkey[]))` is true.
RESULT `event(Success)==>event(NodeOK(leftkey[]))` is true.
RESULT `not event(NodeOK(rightkey[]))` is false.
RESULT `not event(NodeOK(leftkey[]))` is false.
RESULT `not event(Success)` cannot be proved.
RESULT `not event(ActualSuccess)` is false.

“Cannot be proved” in ProVerif lingo means “I don’t know”. In other words, in both models ProVerif fails to verify whether `success` is reachable or not; it also fails to verify whether `ActualSuccess` always precedes `Success`. All other queries produce the expected result.

It is no secret that ProVerif is an incomplete (but sound) solver. It is unfortunate that we seem to have struck incompleteness in this particular case, and it would be interesting to discover why. The ProVerif manual discusses various sources of incompleteness [11, pp. 92-93], and it should be investigated which (of any) applies.

On a more positive note, the inconclusive answer arrives really fast (a matter of milliseconds at most). Since no satisfactory answers were procured on a two-node network, we have not investigated how this scales to larger networks.

### 5 Pwb Development

During the course of this project, we have extended Pwb in significant ways and developed a new advanced module (instance) for Pwb. We strived throughout this enterprise to make the work as generic as possible, so that it may be applicable to other similar formalisation attempts.

Our work on Pwb consists of extending the main codebase of the tool and developing a module which allows Pwb to use first order algebraic datatypes with equational logic on the data. We have implemented the needed interface with Pwb to parse and represent such algebraic specifications and also we implemented a translation to SMT-LIB language as an interface to external
SMT solvers to do the heavy lifting of solving these equations. Thus, it was a conscious design decision to keep the algebraic specification simply sorted first-order language close to that of SMT solvers. An example of the specification is found earlier in this report in Listing 1.

We have implemented support for algebraic specification (signature) consisting of finite number of sorts with distinguished sort for booleans, a list of first order and sorted (function symbols), and a list of equations over the terms formed over the signature. We introduced a distinction between sorts (akin to variables) and constant sorts (akin to fixed names) which is not found in SMT-LIB. Their treatment when solving transition constraints differ as follows: we accept solutions where two values with the same sort are equated, but do not accept such solutions for constant sorts. This helps rule out many nonsensical scenarios, such as transitions that are available under the constraint that the private keys of two different nodes are equated. In our implementation, labels, MACs, node names and nonces are sorts; channels and keys are constant sorts.

Here we list a detailed list of enhancements to Pwb on top of the basis described above.

**Concurrent SMT solvers.** We implemented support for checking constraints on multiple SMT solvers concurrently. Implemented with the Poly/ML Threads library [29]. Queries are run simultaneously on CVC4 [8] and Z3 [20]; this turns out to be practically useful since many queries that time out on Z3 are solved instantly by CVC4, and vice versa. The implementation is such that it would be trivial to add more solvers. However, even though there are many SMT solvers on the market, our current encoding of transition constraints into SMT queries requires an SMT-LIB logic with some rather advanced features (at least UF NIA). Very few solvers support these features. In fact, the only solvers even entered for this category for the 2015 SMT-COMP were CVC4, Z3, and CVC3. Looking beyond SMT solvers, a promising candidate for another back-end solver to include would be VAMPIRE [36].

**Caching of SMT solver queries.** We implemented caching of SMT solver queries. The result of all queries are stored in a hash table, that is indexed by a string representation of the query, but with all identifier names replaced by de Bruijn-indices. Using the de Bruijn representation allows us to increase the number of cache hits, since hits are up-to alpha rather than only for syntactically equal queries. An interesting direction for future work might be hash tables indexed by canonical representations of constraints up-to more sophisticated equivalences, such as associativity and
commutativity (AC) congruence of formulas, and reordering of constraints.

Query caching is practically indispensable in the context of symbolic execution [31], since transition constraints are often very similar to each other. In our work we have often experienced drastic improvements in efficiency and reliability of our tool by simply introducing caching, or increasing the maximum cache size when we hit brick walls.

**Reliable broadcast semantics.** We implemented reliable broadcast semantics [4]. By using broadcast communication on a private channel, we can model authenticated broadcast without having to introduce an explicit protocol to disseminate messages through the network, and without having to introduce a model of the associated cryptography. By using reliable as opposed to unreliable broadcast communication, we obtain smaller state spaces by removing many uninteresting cases from consideration, namely those where the protocol cannot run to completion because not every participant knows what’s going on. Since we disregard denial of service attacks such scenarios are not interesting.

The implementation of reliable broadcast is based on a simple observation. We consider the same transition candidates as in the unreliable case (namely for each unguarded output prefix, there is one candidate transition to be evaluated for every subset of the available input prefixes). For each transition candidate, we keep track of which rules are applied in its derivation. First, the constraints corresponding to all candidate transitions are solved. Of the ones that are solvable, we inspect their derivations and count how many times the rules \( \text{Com} \) and \( \text{Merge} \) are applied in the derivations; call this the **reliability value** of the transition. We then discard all solvable candidate transitions whose reliability value is less than some other solvable candidate transition.

Intuitively, the reason this algorithm works is that a lower reliability value means that at some point in the derivation tree, \( \text{Par} \) was used when \( \text{Merge} \) was applicable, which is prohibited by the semantics of reliable broadcast. It should be noted that this algorithm is incorrect in the presence of free choice. For an example, consider the process

\[
P \triangleq \overline{M} | (M | M) + M.Q
\]

With reliable broadcast semantics this process has two transitions (up-to struct): (1) \( P \xrightarrow{M} 0 \) and (2) \( P \xrightarrow{M} Q \). However, transition (1) has reliability value 2 and transition (2) has reliability 1, so the algorithm will discard transition (2) and thus conclude that only (1) is available. Fortunately, this does not come into play in the SHIA model since it does not use choice at all (all its case statements are unary). An interesting direction for future work
is to find a symbolic semantics for reliable broadcast in the presence of free choice, and an efficient implementation thereof.

For every unguarded broadcast output we must generate one transition constraint for every subset of the unguarded broadcast inputs. For an example, the process

\[ P \triangleq \overline{M} \mid \Pi_{1 \leq i \leq 16} M_i \]

generates \(2^{16} = 65536\) transition constraints (c.f. point-to-point semantics, where \(P\) has only 16 transition constraints)! This does not scale at all well to larger systems; merely generating (not even solving) transition constraints for larger networks (say, 8 nodes or more) can take an annoyingly long time, and represents a performance brick wall waiting to happen at even larger network sizes.

However, for the process \(P\), the human observer instantly realises that 65535 of these constraints are in fact logically equivalent! This suggests that transition constraint generation for broadcast can probably be done in a more clever way that avoids this explosion.

The modelling of the SHIA protocol was a significant driver for improving the Pwb tool. The result of this is a solid first-order algebraic specification for the term language of the processes with battle-tested improvements on the interface with SMT solvers, pragmatic trade-offs on improving the constraint generation and solving run-times, and bringing the implementation of communication primitives closer to what is required by the protocol.

6 Related Work

SHIA [14] is a secure hierarchical in-network aggregation protocol for WSNs. SHIA is well known for introducing a hierarchical structure and the concept of optimal security to secure in-network aggregation in WSNs. It has been cited over 300 times in Google Scholar and considered in terms of efficiency in network congestion so far due to the high energy cost of data transmission in WSNs.

Chan et al.[14] claims that SHIA is a provably secure protocol, yet, as Chatterjee et al.[15] states the security proofs are provided informally. Our aim is to verify SHIA using a formal methodology and the most related works in this direction are mentioned here. Manulis and Schwenk [28] formalise optimal security using the popular sequence of games approach [37] and proves that this property is satisfied for secure aggregation protocols; however, they do not address a hierarchical scenario as in SHIA. In addition to optimal security, there is \(\mu\)TESLA [34] which has an essential role in
SHIA by providing authenticated broadcast communication with symmetric cryptographic primitives. Ballardin and Merro [7] have formalised $\mu$TESLA using a timed broadcasting calculus for wireless systems. Ballardin and Merro have proven that $\mu$TESLA’s time dependent authentication property that takes place in the broadcast holds. $\mu$TESLA has also been modelled by Tobarra et al. [40] with the UPPAAL tool [9, 26] that uses timed automata to model real-time systems.

There are also other works with formal methods in order to verify WSN security protocols. Macedonio and Merro [27] later extended their work of modelling $\mu$TESLA with formalisation of LEAP+ [44] and LiSP [32], which are among well known key management protocols for WSNs. Tobarra et al. [38, 39] has also used AVISPA tool [5], which allows to specify security protocols with their properties via a high-level formal language, to model SNEP [34] and, then, TinySec [25], LEAP [43] and TinyPK [42]. SNEP (Secure Network Encryption Protocol) provides data confidentiality, authentication, integrity and freshness to a WSN. TinySec brings access control, message integrity and message confidentiality to link layer communication in a WSN. TinyPK allows the use of public-key based authentication and key agreement between WSNs.

There are efforts in comparing formal verification tools for modelling security protocols. One example is [18], which compares use of state spaces in AVISPA, ProVerif, Scyther and Casper/FDR. Another work [24] compares Hermes and AVISPA with respect to their complexity, front-end languages, verifiable security services, intrusion models and back-end analytical tools. Another work [16] compares Casper/FDR, STA, $S^3A$, and OFMC with respect to their features and ability to detect bugs under the same experimental conditions. Perhaps, the most related work among these works is [19], which implements six popular cryptographic protocols in ProVerif and Scyther in order to outline different characteristics of these tools.

7 Conclusions and Future Work

In this study, we have modelled a secure aggregation protocol for WSN (SHIA) in three modelling languages and tools ($Pwb$, $mCRL2$ and ProVerif). We needed to make some concessions on using both ProVerif and $mCRL2$: reliable authenticated broadcast was modelled as forwarding of point-to-point messages; authenticity was modelled with public key cryptography.

We have made significant improvements to $Pwb$ that allowed us to model reliable authenticated broadcast, query multiple SMT solvers concurrently and cache queries for performance gain in symbolic execution. With the latest version of $Pwb$, we are able to represent our abstract protocol model.
of SHIA, which is extracted from the protocol description in [14]. At the abstraction level and size of the network that we modelled, we did not find any problems with the SHIA protocol.

As pointed out in Section 5, efficient implementation of a symbolic semantics for reliable broadcast in the presence of free choice can be considered for future work in order to achieve significant gain in the performance of constraint solving.

References


Recent licentiate theses from the Department of Information Technology

2016-006  Anton Axelsson: Context: The Abstract Term for the Concrete

2016-005  Ida Bodin: Cognitive Work Analysis in Practice: Adaptation to Project Scope and Industrial Context

2016-004  Kasun Hewage: Towards a Secure Synchronous Communication Architecture for Low-power Wireless Networks


2016-002  Rubén Cubo: Mathematical Modeling for Optimization of Deep Brain Stimulation

2016-001  Victor Shcherbakov: Radial Basis Function Methods for Pricing Multi-Asset Options

2015-006  Hanna Holmgren: Towards Accurate Modeling of Moving Contact Lines

2015-005  Siyang Wang: Analysis of Boundary and Interface Closures for Finite Difference Methods for the Wave Equation

2015-004  Pavol Bauer: Parallelism and Efficiency in Discrete-Event Simulation

2015-003  Fredrik Hellman: Multiscale and Multilevel Methods for Porous Media Flow Problems

2015-002  Ali Dorostkar: Developments in preconditioned iterative methods with application to glacial isostatic adjustment models

2015-001  Karl Ljungkvist: Techniques for Finite Element Methods on Modern Processors

UPPSALA UNIVERSITY

Department of Information Technology, Uppsala University, Sweden