A Secure Relay Protocol for Door Access Control*

Erik Ramsgaard Wognsen, Henrik Søndberg Karlsen, Marcus Calverley
Mikkel Normann Follin, Bent Thomsen, Hans Hüttel

[1] Department of Computer Science, Aalborg University
Aalborg, Denmark

{hkarls07,mcalve07,mfolli07}@student.aau.dk
{erw,bt,hans}@cs.aau.dk

Abstract. *Physical keys are easy to use but difficult to manage securely for large institutions. Digital replacements have been created, but dedicated hardware such as smartcards or RFID tags can have the same problems as physical keys. Several commercial products try to solve this by using the users' Bluetooth-enabled mobile devices as keys, but the built-in security of the Bluetooth standard is insufficient. Furthermore, to manage a varying set of users, such systems may require the door locks to be connected to the Internet which may require expensive infrastructure.*
We present a cryptographic protocol and a prototype implementation that solves these problems by letting door locks communicate with a central server using the Internet connections of the users' mobile devices. The protocol is specified formally in the applied π-calculus and security through secrecy and authenticity is verified using the cryptographic protocol verifier ProVerif. A prototype of the system is implemented for Android smartphones.

1. Introduction

Keys have for centuries been a central part of managing access rights to buildings. Physical keys are easy to use, but difficult to manage, especially for large institutions with a large and varying number of users and even worse when access rights to a building change over time. Digital replacements for physical key systems can solve this problem and they have been around for several decades, but until recently required dedicated hardware to be issued to the user, e.g. smartcards or RFID tags, which to some extent are as cumbersome to manage as their physical key counter parts. With the proliferation of commodity smartphones the idea of using a smartphone as a wireless key has been commercialized by several vendors such as MVC-Data [1], Bluelon ApS [2], Flexipanel Ltd. [3], Steab AB [4], ECKey [5], and SOREX [6]. Several of these commercial systems are based on establishing a Bluetooth connection between door and smartphone and rely on the Bluetooth stack for security. Although this provides "good enough" security between the door and the smartphone for many practical uses, it is possible to compromise the security with some effort and modified consumer hardware [7]. Furthermore, to manage a varying set of users, such systems either require the door locks to be connected to the Internet, or manual entry of each user in each door lock.

In this paper we present a system, developed in collaboration with MVC-Data, that can solve both problems mentioned above. The system is designed so that each door lock

*This work is partially funded by the European Regional Development Fund and Centre for Embedded Software Systems (CISS)

requires no knowledge of the individual users, as well as no direct Internet connection. Instead, the door locks use the Internet connection provided by the user's smartphone to set up a secure connection with a central server which holds the information about each user, their access rights and audit logs. For communication between the door lock and the smartphone, the system uses an insecure Bluetooth connection without pairing. Figure 1 illustrates the architecture. Informally, the protocol works as follows: The smartphone app runs a service in the background that keeps a connection to the server and scans for door units. When the smartphone gets within a predefined range of a door unit, it connects to the door unit and asks it to unlock. To find out if it should open, the door unit consults the server through the Internet connection of the smartphone.

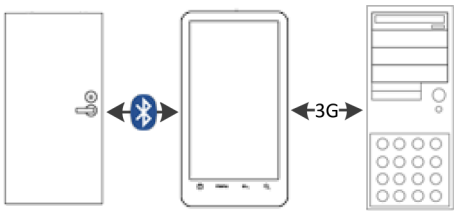

Figure 1. The architecture: a smartphone creates a link between a door module and a server through Bluetooth and an Internet connection.

This paper has two main contributions: a formal protocol and verification of relevant security properties of the protocol. The protocol uses a public key infrastructure where a central authentication server is able to authenticate users for individual door locks, based on their private key which they save on their smartphones. It is designed to allow users to securely unlock doors through an insecure Bluetooth connection and authenticate them through an insecure Internet connection. The protocol provides a secure layer using hybrid encryption such that asymmetric encryption is used for authentication and to establish a faster, symmetric encryption for larger message exchanges. The protocol is formalized in the applied π-calculus [8, 9]. Furthermore, the protocol is verified using ProVerif [10] to be secure against eavesdropping (by strong secrecy), man-in-the-middle attacks, and replay attacks.

To the best of our knowledge, ours is the first formally verified wireless protocol using Bluetooth and 3G communication.

ProVerif has been used to verify the security of several protocols including an electronic voting protocol [11] and implementations of TLS [12]. Additionally, in [13], ProVerif was used to analyze the authentication process of Bluetooth device pairing, where it was able to determine an attack strategy that confirms a previously known vulnerability. A list of these and other publications using ProVerif can be found in [14].

The rest of this paper is organized as follows: in Section 2, an informal description of the protocol is given followed by a classic protocol narration in the form of message sequence charts. In Section 3 the applied π-calculus is introduced and in Section 4 it is used to formalize the protocol. The applied π-calculus is used as an intermediate step which helps in the description and verification of properties using ProVerif in Section 5. Section 6 describes the implementation and illustrates how the ProVerif model relates to the prototype implementation. Section 7 concludes.

2. The Protocol Narration

The protocol narration describes each step in the protocol in a manageable way, but it leaves out many details such as checks that must be done at each step to know when to abort communication because of an attack. These details are instead included in the applied π-calculus model. Table 1 describes the notation we use in the protocol narration.

Table 1. Notation for the protocol narration.

M, S, D	Identities of the mobile unit, server and door unit
K_X, K_X^{-1}	The public and private key of principal X
$\{x\}_K, \{x\}_{K^{-1}}$	Public and private encryption (signature) of message x
$[x]$	A cryptographic hash of message x
N_x	A nonce named x
EK_{XY}, AK_{XY}	Encryption and authentication keys for symmetrically encrypted communication from X to Y
K_{XY}	Key material for deriving EK_{XY}, AK_{XY}, EK_{YX}, and AK_{YX}
$[x]^h_{AK}$	MAC of message x using authentication key AK
$\{x\}^s_{EK}$	Symmetric encryption of message x with encryption key EK
x	An arbitrary value
"foo"	A string literal
$m_x = \ldots$	A subexpression introduced for clarity

Before interacting with any doors, the mobile unit M connects to the server S and is authenticated by the server using asymmetric encryption. Figure 2 shows the messages exchanged during this first phase. The data that is encrypted asymmetrically is key material that will be used to encrypt all following messages symmetrically. Nonces are sent in every step to avoid replay attacks and each nonce is returned to prove the receiver was able to decrypt the nonce. After the symmetric keys are exchanged and the principals trust each other, the channel is ready to send arbitrary messages as shown by the message x and the ellipsis. The mobile unit can now look for a door unit to connect to.

When the connection to the server S is established and the mobile unit M wants to connect to a door unit D, it queries S for the public key of D. It then uses asymmetric encryption to share key material with D for the symmetrically encrypted communication with nonces being sent and returned. Figure 3 shows this process. As before, the mobile unit verifies the identity of its communication partner, but since the server will later decide if the mobile unit is allowed access, the door does not need to verify the identity of the mobile unit.

Figure 4 shows what happens when the user wants to unlock the door after having established connections to the server and the door. It is the most complicated figure since communication is relayed through the mobile unit which takes the output from one secure channel and sends it over the other while the messages themselves are the encrypted

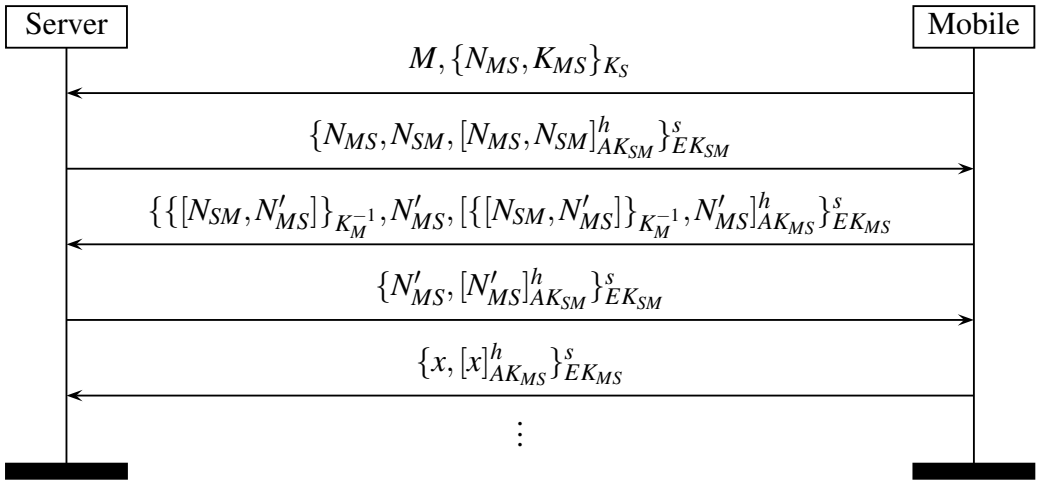

Figure 2. The mobile unit connects to the server.

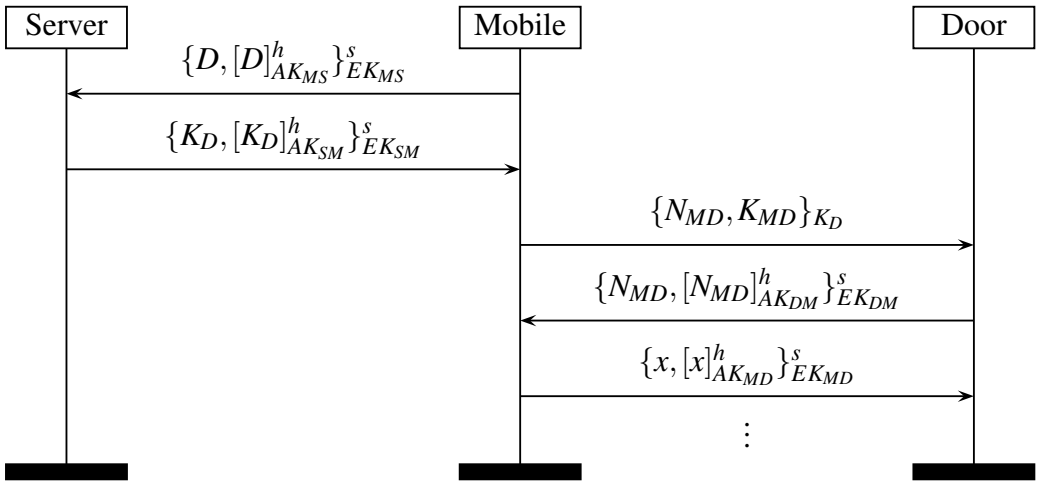

Figure 3. The mobile unit connects to a door after having established a connection with the server.

challenges between door and server. When the door is authenticated the server looks up whether the owner of the mobile unit has access to the door at that time and sends "1" or "0" in m_4 depending on whether the user is allowed access or not.

3. The Applied π-calculus

The applied π-calculus was proposed by Abadi and Fournet [15] as a generalization of the many extensions of the π-calculus [16] in existence. Since then, Blanchet has used a version of the applied π-calculus as the basis of the ProVerif tool.

While the protocol narration describes the messages exchanged, the applied π-calculus model formalizes exactly what properties must be checked by each principal at each step.

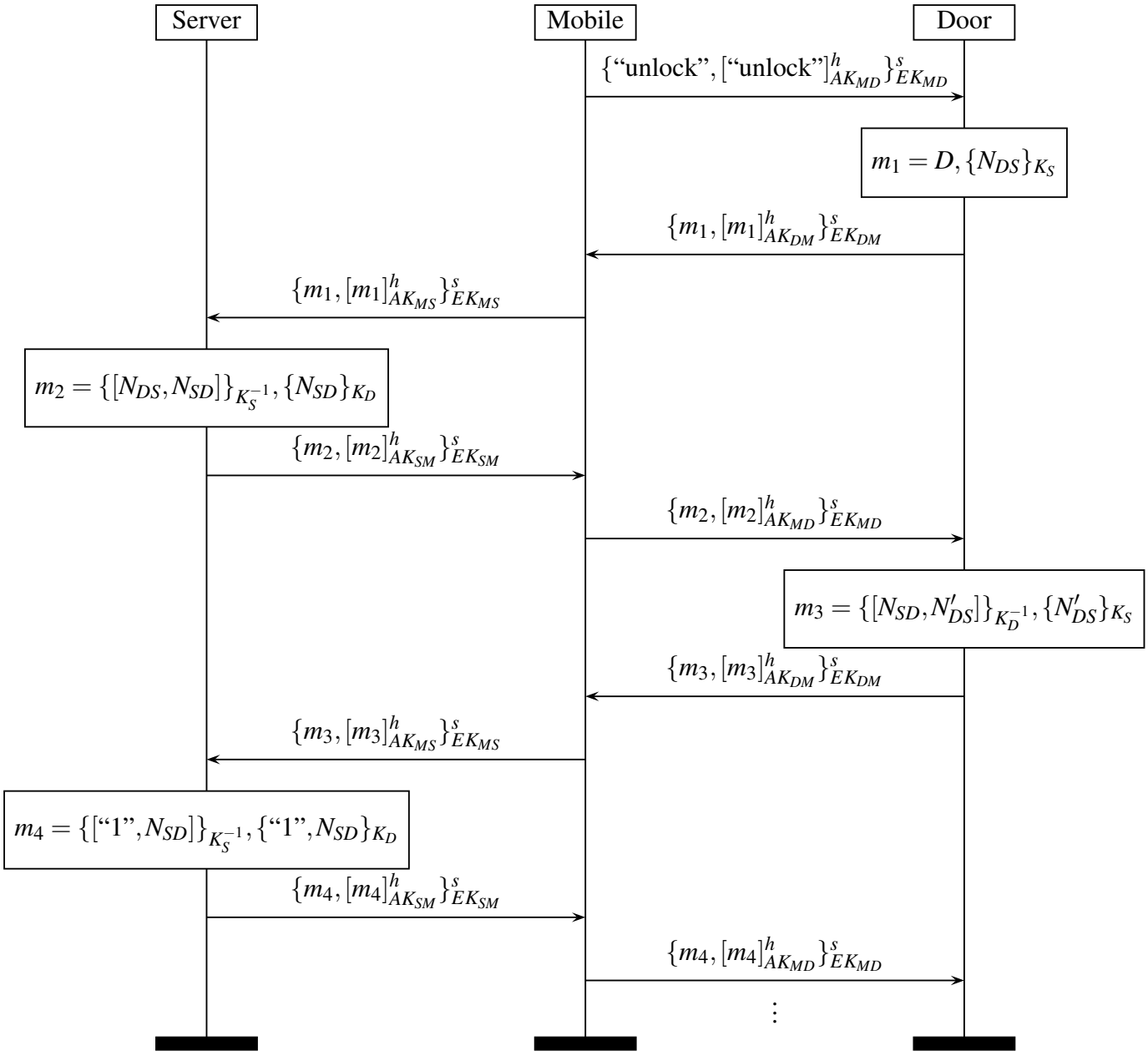

Figure 4. The door is unlocked after having the mobile unit establish connections to server and door so that the door and server can communicate through the mobile unit.

The syntax of applied π-calculus processes presupposes a set of terms built from a finite signature and a countable set of names, ranged over by $x, y, z \ldots$. The syntax of processes is then given by the following formation rules where terms are here ranged over by M and N.

$$P ::= x(y).P_1 \mid \overline{x}\langle N\rangle.P_1 \mid (P_1|P_2) \mid (\nu x)P_1 \mid !P_1 \mid \{M/y\} \mid [M = N]P_1$$

Here, $x(y).P_1$ denotes an input process that on the channel named x can receive a name, instantiating y within the continuation P_1. The output process $\overline{x}\langle N\rangle.P_1$ sends out the term

N on channel x. $P_1|P_2$ denotes the parallel composition of processes P_1 and P_2. $(\nu x)P_1$ denotes that the name x is private to and bound within P_1. $!P_1$ denotes an unbounded supply of parallel components of the form P_1. An active substitution, $\{M/y\}$, denotes that the name y is bound to M. Finally, $[M = N]P_1$ is a match process that can only proceed as P_1 if the message terms M and N are provably equal.

In our presentation, we sometimes define processes by means of parameterized definitions of the form

$$A(\vec{x}) \stackrel{\text{def}}{=} P$$

where A is an agent identifier that may also occur in the right-hand-side process P.

The semantics of the applied π-calculus uses reduction semantics (see Abadi and Fournet for the definition) with reductions of the form

$$P \to P'$$

We write $P \to^* P'$, if P becomes P' within zero or more reduction steps.

4. The Protocol in the Applied π-calculus

The protocol is represented by the following process, where the pk function is defined in Table 2:

$$\begin{aligned}\mathcal{P}(K_S^{-1}, K_M^{-1}, K_D^{-1}, c_i, c_b, c_m, l_M, f_D) \stackrel{\text{def}}{=} \\ !\,\mathcal{P}_M(K_M^{-1}, \mathsf{pk}(K_S^{-1}), c_i, c_b) \mid \\ !\,\mathcal{P}_S(K_S^{-1}, c_i, c_m, l_M, f_D) \mid \\ !\,\mathcal{P}_D(K_D^{-1}, \mathsf{pk}(K_S^{-1}), c_b, c_m)\end{aligned}$$

The process is parameterized by a number of variables: the private keys of the three parties, the public key of the server, three public channels, a set of mobile units l_M representing the server database of identities of registered mobile units, and a function f_D mapping the Bluetooth addresses of the registered door units to their identities. Channels c_i and c_b represent communication over the Internet and over Bluetooth, respectively. When the communication link between the door and server is set up using the mobile unit as a relay, the public channel c_m is used to model the mobile unit as an untrusted party.

Public channels may be monitored by any attacker and manipulated. Messages in transit may be dropped, altered, replayed, replaced, etc. This is typically how cryptographic adversaries are modeled.

Table 2. One-way functions

$\mathsf{pk}(K^{-1})$	Derive public key from private key
$\mathsf{hash}(m)$	Cryptographic hash of message m
$\mathsf{mac}(K,m)$	MAC of message m using key K
$\mathsf{ekxy}(K_{XY})$	Derive encryption key for symmetric communication $X \to Y$
$\mathsf{ekyx}(K_{XY})$	Derive encryption key for symmetric communication $Y \to X$
$\mathsf{akxy}(K_{XY})$	Derive authentication key for symmetric communication $X \to Y$
$\mathsf{akyx}(K_{XY})$	Derive authentication key for symmetric communication $Y \to X$

Table 3. Functions with reductions

$\mathsf{publicEncrypt}(\mathsf{pk}(K^{-1}),x)$	—Abbreviated as $\{x\}_{\mathsf{pk}(K^{-1})}$
$\mathsf{privateDecrypt}(K^{-1},\mathsf{publicEncrypt}(\mathsf{pk}(K^{-1}),x)) = x$	
$\mathsf{sign}(K^{-1},x)$	
$\mathsf{stripSign}(\mathsf{pk}(K^{-1}),\mathsf{sign}(K^{-1},x)) = x$	
$\mathsf{symEncrypt}(K,IV,x)$	—Abbreviated as $\{x\}_K^{IV}$
$\mathsf{symDecrypt}(K,IV,\mathsf{symEncrypt}(K,IV,x)) = x$	
$\mathsf{m_i}(x_0,x_1,\ldots,x_n)$	—Packing components of message i
$\mathsf{m_i.j}(\mathsf{m_i}(x_0,x_1,\ldots,x_n)) = x_j \quad \text{for } j \in 0,1,\ldots,n$	—Unpacking
$\mathsf{w}(x_0,x_1,x_2)$	—Wrapping message with meta data
$\mathsf{w.j}(\mathsf{w}(x_0,x_1,x_2)) = x_j \quad \text{for } j \in 0,1,2$	—Unwrapping

Table 2 shows the one-way functions used in the model while Table 3 shows the function reductions. Most functions represent standard cryptographic operations in idealized forms. The packing/wrapping functions have no effect on the protocol other than providing a familiar structure for the processes. Packing is used for grouping arbitrary components into a single message while wrapping is used for grouping a message with an initialization vector (IV) and a message authentication code (MAC) for the symmetric encryption.

For convenience, notation for decryption with pattern matching is introduced as in [8] such that the process

$$\text{if } ct_{XY}^n = \{\mathsf{w}(\nu m_{XY}^n, \nu IV_{XY}^n, \nu H_{XY}^n)\}_{\mathsf{ekyx}(K_{XY})}^{IV_{XY}^{n-1}} \text{ using } \mathsf{ekyx}(K_{XY}) \text{ then } Q$$

is equivalent to the following which includes checking that the computed MAC matches the received MAC:

$$\nu m_{XY}^n.\nu IV_{XY}^n.\nu H_{XY}^n.\nu HC_{XY}^n.$$
$$\left(\begin{array}{l} \{\ m_{XY}^n = \mathsf{w.0}(\mathsf{symDecrypt}(\mathsf{ekyx}(K_{XY}), IV_{XY}^{n-1}, ct_{XY}^n))\} \mid \\ \{\ IV_{XY}^n = \mathsf{w.1}(\mathsf{symDecrypt}(\mathsf{ekyx}(K_{XY}), IV_{XY}^{n-1}, ct_{XY}^n))\} \mid \\ \{\ H_{XY}^n = \mathsf{w.2}(\mathsf{symDecrypt}(\mathsf{ekyx}(K_{XY}), IV_{XY}^{n-1}, ct_{XY}^n))\} \mid \\ \{\ HC_{XY}^n = \mathsf{mac}(\mathsf{akyx}(K_{XY}), (m_{XY}^n, IV_{XY}^n))\} \mid \\ \text{if } ct_{XY}^n = \{\mathsf{w}(m_{XY}^n, IV_{XY}^n, H_{XY}^n)\}_{\mathsf{ekyx}(K_{XY})}^{IV_{XY}^{n-1}} \text{ and } H_{XY}^n = HC_{XY}^n \text{ then } Q \end{array}\right)$$

The first steps of $\mathcal{P}_M$ and $\mathcal{P}_S$ are shown here in order to explain the meaning of the processes and their interaction. Instead of simply establishing a shared secret in a few steps as many protocols do, this protocol consists of several parts with variable numbers of steps each. One specific, minimal run of the full protocol is modeled. Since the protocol also includes more low-level details (including hybrid encryption) than many other protocols, the applied π-calculus formalization is relatively long. The full processes are found in Figures 5, 6, and 7.

First part of $\mathcal{P}_M$ derives the corresponding public key from its private key and then creates a nonce, an IV and key material for the symmetric communication. The public key is sent over the Internet channel along with the nonce, IV, and key material encrypted

with the public key of the server:

$$\text{let } K_M = \mathsf{pk}(K_M^{-1}) \text{ in } \nu N_{MS}.\nu IV_{MS}^1.\nu K_{MS}.$$
$$\overline{c_i}\langle K_M, \{\mathsf{m}_{\mathsf{MS}}^1(N_{MS}, IV_{MS}^1, K_{MS})\}_{K_S}\rangle$$

The server receives the messages sent by the mobile unit and checks that it is registered before continuing. The encrypted message is decrypted and the components extracted. Then, a new nonce is created a sent back with the one from the mobile unit using symmetric encryption with the key material and IV sent by the mobile unit. A new IV is also included in the payload for use with the next message in the communication:

$$c_i(M, ct_{MS}^1).\text{if } M \in l_M \text{ then}$$
$$\text{if } ct_{MS}^1 = \{\mathsf{m}_{\mathsf{MS}}^1(\nu N_{MS}, \nu IV_{MS}^1, \nu K_{MS})\}_{K_S} \text{ using } K_S^{-1}$$
$$\text{then } \nu N_{SM}.\nu IV_{MS}^2.\text{let } m_{MS}^2 = \mathsf{m}_{\mathsf{MS}}^2(N_{MS}, N_{SM}) \text{ in}$$
$$\overline{c_i}\langle\{\mathsf{w}(m_{MS}^2, IV_{MS}^2, \mathsf{mac}(\mathsf{akyx}(K_{MS}), (m_{MS}^2, IV_{MS}^2)))\}_{\mathsf{ekyx}(K_{MS})}^{IV_{MS}^1}\rangle$$

The mobile unit then receives, decrypts, and extracts the reply from the server. If the nonce previously sent by the mobile unit is returned correctly, it continues to create a new message containing a third nonce and this nonce hashed with the one generated by the server. This is then sent to the server:

$$c_i(ct_{MS}^2).\text{if } ct_{MS}^2 = \{\mathsf{w}(\nu m_{MS}^2, \nu IV_{MS}^2, \nu H_{MS}^2)\}_{\mathsf{ekyx}(K_{MS})}^{IV_{MS}^1}$$
$$\text{using } \mathsf{ekyx}(K_{MS}) \text{ and } \mathsf{m}_{\mathsf{MS}}^2.0(m_{MS}^2) = N_{MS} \text{ then}$$
$$\nu N'_{MS}.\nu IV_{MS}^3.\text{let } m_{MS}^3 = \mathsf{m}_{\mathsf{MS}}^3(\mathsf{sign}(K_M^{-1}, \mathsf{hash}(\mathsf{w}.1(m_{MS}^2), N'_{MS})), N'_{MS}) \text{ in}$$
$$\overline{c_i}\langle\{\mathsf{w}(m_{MS}^3, IV_{MS}^3, \mathsf{mac}(\mathsf{akxy}(K_{MS}), (m_{MS}^3, IV_{MS}^3)))\}_{\mathsf{ekxy}(K_{MS})}^{IV_{MS}^2}\rangle$$

5. Verification of the Protocol using ProVerif

ProVerif is a tool for verifying cryptographic protocols that supports verification of protocols using common cryptographic primitives including symmetric and asymmetric encryption. By creating a model of a protocol in the applied π-calculus, it is possible to write queries that ProVerif can answer to prove certain properties, e.g., whether or not a message sent between two trusted entities can be obtained by an attacker. The complete ProVerif model of the protocol can be downloaded at [17].

We will use ProVerif to verify these three security properties:

Secrecy An attacker is not able to read secret data.
Strong secrecy An attacker is not able to discern when secret data changes.
Authenticity An attacker is not able to impersonate a legitimate party of the system.

Our ProVerif model is directly based on the applied π-calculus description of Section 4. The remainder of this section shows examples of the use of ProVerif queries to verify the security of the protocol.

$$
\begin{aligned}
&\mathcal{P}_M(K_M^{-1},K_S,c_i,c_b) \stackrel{\text{def}}{=} \text{ let } K_M = \mathsf{pk}(K_M^{-1}) \text{ in } \nu N_{MS}.\nu IV_{MS}^1.\nu K_{MS}.\\
&\quad \overline{c_i}\langle K_M, \{\mathsf{m}^1_{\mathsf{MS}}(N_{MS},IV_{MS}^1,K_{MS})\}_{K_S}\rangle.c_i(ct_{MS}^2).\\
&\quad \text{if } ct_{MS}^2 = \{\mathsf{w}(\nu m_{MS}^2,\nu IV_{MS}^2,\nu H_{MS}^2)\}^{IV_{MS}^1}_{\mathsf{ekyx}(K_{MS})} \text{ using } \mathsf{ekyx}(K_{MS})\\
&\qquad \text{and } \mathsf{m}^2_{\mathsf{MS}}.0(m_{MS}^2) = N_{MS} \text{ then}\\
&\quad \nu N'_{MS}.\nu IV_{MS}^3.\text{let } m_{MS}^3 = \mathsf{m}^3_{\mathsf{MS}}(\mathsf{sign}(K_M^{-1},\mathsf{hash}(\mathsf{w}.1(m_{MS}^2),N'_{MS})),N'_{MS}) \text{ in}\\
&\quad \overline{c_i}\langle\{\mathsf{w}(m_{MS}^3,IV_{MS}^3,\mathsf{mac}(\mathsf{akxy}(K_{MS}),(m_{MS}^3,IV_{MS}^3)))\}^{IV_{MS}^2}_{\mathsf{ekxy}(K_{MS})}\rangle.c_i(ct_{MS}^4).\\
&\quad \text{if } ct_{MS}^4 = \{\mathsf{w}(\nu m_{MS}^4,\nu IV_{MS}^4,\nu H_{MS}^4)\}^{IV_{MS}^3}_{\mathsf{ekyx}(K_{MS})} \text{ using } \mathsf{ekyx}(K_{MS})\\
&\qquad \text{and } \mathsf{m}^4_{\mathsf{MS}}.0(m_{MS}^4) = N'_{MS} \text{ then}\\
&\quad c_b(b).\nu IV_{MS}^5.\text{let } m_{MS}^5 = \mathsf{m}^5_{\mathsf{MS}}(b) \text{ in}\\
&\quad \overline{c_i}\langle\{\mathsf{w}(m_{MS}^5,IV_{MS}^5,\mathsf{mac}(\mathsf{akxy}(K_{MS}),(m_{MS}^5,IV_{MS}^5)))\}^{IV_{MS}^4}_{\mathsf{ekxy}(K_{MS})}\rangle.c_i(ct_{MS}^6).\\
&\quad \text{if } ct_{MS}^6 = \{\mathsf{w}(\nu m_{MS}^6,\nu IV_{MS}^6,\nu H_{MS}^6)\}^{IV_{MS}^5}_{\mathsf{ekyx}(K_{MS})} \text{ using } \mathsf{ekyx}(K_{MS}) \text{ then}\\
&\quad \text{let } K_D = \mathsf{m}^6_{\mathsf{MS}}(m_{MS}^6) \text{ in}\\
&\quad \nu N_{MD}.\nu IV_{MD}^1.\nu K_{MD}.\text{let } m_{MD}^1 = \mathsf{m}^1_{\mathsf{MD}}(N_{MD},IV_{MD}^1,K_{MD}) \text{ in}\\
&\quad \overline{c_b}\langle\{m_{MD}^1\}_{K_D}\rangle.c_b(ct_{MD}^2).\\
&\quad \text{if } ct_{MD}^2 = \{\mathsf{w}(\nu m_{MD}^2,\nu IV_{MD}^2,\nu H_{MD}^2)\}^{IV_{MD}^1}_{\mathsf{ekyx}(K_{MD})} \text{ using } \mathsf{ekyx}(K_{MD})\\
&\qquad \text{and } \mathsf{m}^2_{\mathsf{MD}}.0(m_{MD}^2) = N_{MD} \text{ then}\\
&\quad \nu IV_{MD}^3.\text{let } m_{MD}^3 = \text{“unlock” in}\\
&\quad \overline{c_b}\langle\{\mathsf{w}(m_{MD}^3,IV_{MD}^3,\mathsf{mac}(\mathsf{akxy}(K_{MD}),(m_{MD}^3,IV_{MD}^3)))\}^{IV_{MD}^2}_{\mathsf{ekxy}(K_{MD})}\rangle
\end{aligned}
$$

Figure 5. The process describing the mobile unit.

5.1. Secrecy and Strong Secrecy

Certain variables have been marked as secret in the ProVerif model. This has been done to be able to subsequently test if an attacker is able to read encrypted data sent with the protocol. The following excerpt from the ProVerif model exemplifies this:

```
1 free access: bool [private].
2 query attacker(access).
3 noninterf access.
```

The variable `access` defined as secret (or private) in line 1 is sent from the server to the door unit and decides if the door unit will unlock the door. The query in line 2 determines whether secrecy is maintained, and the query in line 3 whether strong secrecy is maintained. Using the ProVerif tool we have verified that these properties indeed hold; this means that an attacker is not able to read whether someone was granted or denied access to a door, and not even whether someone who was previously granted access has now been denied access.

$$
\begin{aligned}
&\mathcal{P}_S(K_S^{-1}, c_i, c_m, l_M, f_D) \stackrel{\text{def}}{=} c_i(M, ct^1_{MS}).\text{if } M \in l_M \text{ then}\\
&\quad \text{if } ct^1_{MS} = \{\mathsf{m}^1_{\mathsf{MS}}(\nu N_{MS}, \nu IV^1_{MS}, \nu K_{MS})\}_{K_S} \text{ using } K_S^{-1} \text{ then}\\
&\quad \nu N_{SM}.\nu IV^2_{MS}.\text{let } m^2_{MS} = \mathsf{m}^2_{\mathsf{MS}}(N_{MS}, N_{SM}) \text{ in}\\
&\quad \overline{c_i}\langle\{\mathsf{w}(m^2_{MS}, IV^2_{MS}, \mathsf{mac}(\mathsf{akyx}(K_{MS}), (m^2_{MS}, IV^2_{MS})))\}^{IV^1_{MS}}_{\mathsf{ekyx}(K_{MS})}\rangle.c_i(ct^3_{MS}).\\
&\quad \text{if } ct^3_{MS} = \{\mathsf{w}(\nu m^3_{MS}, \nu IV^3_{MS}, \nu H^3_{MS})\}^{IV^2_{MS}}_{\mathsf{ekxy}(K_{MS})} \text{ using } \mathsf{ekxy}(K_{MS})\\
&\qquad \text{and } \mathsf{stripSign}(K_M, \mathsf{m}^3_{\mathsf{MS}}.0(m^3_{MS})) = \mathsf{hash}(N_{SM}, \mathsf{m}^3_{\mathsf{MS}}.1(m^3_{MS})) \text{ then}\\
&\quad \nu IV^4_{MS}.\text{let } m^4_{MS} = \mathsf{m}^4_{\mathsf{MS}}(\mathsf{m}^3_{\mathsf{MS}}.1(m^3_{MS})) \text{ in}\\
&\quad \overline{c_i}\langle\{\mathsf{w}(m^4_{MS}, IV^4_{MS}, \mathsf{mac}(\mathsf{akyx}(K_{MS}), (m^4_{MS}, IV^4_{MS})))\}^{IV^3_{MS}}_{\mathsf{ekyx}(K_{MS})}\rangle.c_i(ct^5_{MS}).\\
&\quad \text{if } ct^5_{MS} = \{\mathsf{w}(\nu m^5_{MS}, \nu IV^5_{MS}, \nu H^5_{MS})\}^{IV^4_{MS}}_{\mathsf{ekxy}(K_{MS})} \text{ using } \mathsf{ekxy}(K_{MS})\\
&\qquad \text{and } \mathsf{m}^5_{\mathsf{MS}}.0(m^5_{MS}) \in \mathrm{dom}(f_D) \text{ then}\\
&\quad \text{let } K_D = f_D(\mathsf{m}^5_{\mathsf{MS}}.0(m^5_{MS})) \text{ in } \nu IV^6_{MS}.\text{let } m^6_{MS} = \mathsf{m}^6_{\mathsf{MS}}(K_D) \text{ in}\\
&\quad \overline{c_i}\langle\{\mathsf{w}(m^6_{MS}, IV^6_{MS}, \mathsf{mac}(\mathsf{akyx}(K_{MS}), (m^6_{MS}, IV^6_{MS})))\}^{IV^5_{MS}}_{\mathsf{ekyx}(K_{MS})}\rangle.c_m(D, ct^1_{DS}).\\
&\quad \text{if } D \in \mathrm{codom}(f_D) \text{ and } ct^1_{DS} = \{\nu N_{DS}\}_{K_S} \text{ using } K_S^{-1} \text{ then}\\
&\quad \nu N_{SD}.\text{let } s^2_{DS} = \mathsf{sign}(K_S^{-1}, \mathsf{hash}(N_{DS}, N_{SD})) \text{ and } m^2_{DS} = \mathsf{m}^2_{\mathsf{DS}}(N_{SD}) \text{ in}\\
&\quad \overline{c_m}\langle s^2_{DS}, \{m^2_{DS}\}_{K_D}\rangle.c_m(s^3_{DS}, ct^3_{DS}).\\
&\quad \text{if } ct^3_{DS} = \{\nu N'_{DS}\}_{K_S} \text{ using } K_S^{-1}\\
&\qquad \text{and } \mathsf{stripSign}(K_D, s^3_{DS}) = \mathsf{hash}(N_{SD}, N'_{DS}) \text{ then}\\
&\quad \nu access.\text{let } s^4_{DS} = \mathsf{sign}(K_S^{-1}, \mathsf{hash}(access, N_{SD}))\\
&\qquad \text{and } m^4_{DS} = \mathsf{m}^4_{\mathsf{DS}}(access, N_{SD}) \text{ in}\\
&\quad \overline{c_m}\langle s^4_{DS}, \{m^4_{DS}\}_{K_D}\rangle
\end{aligned}
$$

Figure 6. The process describing the server.

5.2. Authenticity

Listing 1 shows the correspondence query used to determine if a message received by the door unit about the user's access rights has been sent by the server.

```
query K_D: pubKey, challenge: bool; inj-event(doorUnlockMessageReceived(access)
    ) ==> inj-event(serverSendingFinalUnlock(access)).
```

Listing 1. Proving authenticity of messages from the server.

Correspondence queries prove authenticity using two events: begin and end events. Authenticity is proved if the end event cannot be reached without the begin event being triggered [18]. In Listing 1, these two events are: `serverSendingFinalUnlock` which is the begin event and is triggered when the server is about to send the `access` variable to the door, and the end event `doorUnlockMessageReceived` is triggered when the door has received a message at the time it expected to receive the `access` variable. The query thus verifies that

$$
\begin{aligned}
\mathcal{P}_D(K_D^{-1}, K_S, c_b, c_m) \stackrel{\text{def}}{=}\ & \text{let } K_D = \mathsf{pk}(K_D^{-1}) \text{ in } \nu b. \\
& \overline{c_b}\langle b\rangle.c_b(ct^1_{MD}). \\
& \text{if } ct^1_{MD} = \{\mathsf{m}^1_{\mathsf{MD}}(\nu N_{MD}, \nu IV^1_{MD}, \nu K_{MD})\}_{K_D} \text{ using } K_D^{-1} \text{ then} \\
& \nu IV^2_{MD}.\text{let } m^2_{MD} = N_{MD} \text{ in} \\
& \overline{c_b}\langle\{\mathsf{w}(m^2_{MD}, IV^2_{MD}, \mathsf{mac}(\mathsf{akyx}(K_{MD}), (m^2_{MD}, IV^2_{MD})))\}^{IV^1_{MD}}_{\mathsf{ekyx}(K_{MD})}\rangle.c_b(ct^3_{MD}). \\
& \text{if } ct^3_{MD} = \{\mathsf{w}(\nu m^3_{MD}, \nu IV^3_{MD}, \nu H^3_{MD})\}^{IV^2_{MD}}_{\mathsf{ekxy}(K_{MD})} \text{ using } \mathsf{ekxy}(K_{MD}) \\
& \qquad \text{and } m^3_{MD} = \text{“unlock” then} \\
& \nu N_{DS}.\text{let } m^1_{DS} = \mathsf{m}^1_{\mathsf{DS}}(N_{DS}) \text{ in} \\
& \overline{c_m}\langle K_D, \{m^1_{DS}\}_{K_S}\rangle.c_m(s^2_{DS}, ct^2_{DS}). \\
& \text{if } ct^2_{DS} = \{\nu N_{SD}\}_{K_D} \text{ using } K_D^{-1} \\
& \qquad \text{and } \mathsf{stripSign}(K_S, s^2_{DS}) = \mathsf{hash}(N_{DS}, N_{SD}) \text{ then} \\
& \nu N'_{DS}.\text{let } s^3_{DS} = \mathsf{sign}(K_D^{-1}, \mathsf{hash}(N_{SD}, N'_{DS})) \text{ and } m^3_{DS} = \mathsf{m}^3_{\mathsf{DS}}(N'_{DS}) \text{ in} \\
& \overline{c_m}\langle s^3_{DS}, \{m^3_{DS}\}_{K_S}\rangle.c_m(s^4_{DS}, ct^4_{DS}). \\
& \text{if } ct^4_{DS} = \{\nu access, N_{SD}\}_{K_D} \text{ using } K_D^{-1} \\
& \qquad \text{and } \mathsf{stripSign}(K_S, s^4_{DS}) = \mathsf{hash}(access, N_{SD}) \text{ then} \\
& \nu c_a.\overline{c_a}\langle access\rangle
\end{aligned}
$$

Figure 7. The process describing the door unit. The private channel c_a models the door unit controlling the door lock hardware.

a man-in-the-middle attack is not possible. The fact that the query is injective (`inj-event`) means that the door will never receive more messages from the server than it actually sent, thus preventing replay attacks.

5.3. Discovering Security Vulnerabilities

During the development of the protocol, ProVerif was used to detect vulnerabilities. One example of a vulnerability discovered and fixed during the development was found in proving authenticity of the server on the door access control unit with the query in Listing 1. The following lines are used to prove the authenticity of the message received on the door from the server:

```
1 in(mobile, (s4_DS: signature, ct4_DS: asymEncrypted));
2 let message = privDecrypt(Kpriv_D, ct4_DS) in
3   if verifySign(K_S, hash(message), s4_DS) = true then
4     let (access_received: bool, N_SD_returned: bitstring) = message in
5       if N_SD_returned = N_SD then
6         event doorUnlockMessageReceived(access_received);
```

In line 1, the message is received from the server via the mobile unit. It contains two parts: `s4_DS`, the signature of the message, and `ct4_DS`, the asymmetrically encrypted message. In line 2, the message is decrypted using the private key used by the door,


```
1 event serverSendingMobileChallenge(K_M, N_MS);
2 out(internet, symEncryptWithMAC(ekyx(K_MS), IV1_MS, IV2_MS, akyx(K_MS), (N_MS,
      N_SM)));
3
4 in(internet, ct3_MS: symEncrypted);
5 let (Hc3_MS: mac, H3: mac, IV3_MS: iv, (returnedChallengeSignature: signature,
      Nprime_MS_received: bitstring)) = symDecryptWithMAC(ekxy(K_MS), IV2_MS,
      akxy(K_MS), ct3_MS) in
6 if Hc3_MS = H3 then
7 if verifySign(K_M, hash((N_SM, Nprime_MS_received)), returnedChallengeSignature
      ) = true then
8 event serverAcceptsMobile(K_M, K_MS);
9
10 (* Send all ok to mobile *)
11 out(internet, symEncryptWithMAC(ekyx(K_MS), IV3_MS, IV4_MS, akyx(K_MS),
      Nprime_MS_received));
```

Listing 2. Excerpt of ProVerif model of the server.

```
1 N_SM = Rand.rand_bytes(32)
2 message = N_MS + N_SM
3 self.aes_send(message)
4
5 #Verifying returned challenge from phone
6 dec = self.aes_recv()
7 if len(dec) != 256 + 32:
8   raise Error('Message does not have correct length, aborting')
9 sig, Nprime_MS_received = dec[:256], dec[256:]
10 if not phone_rsa.verify(sha256(N_SM + Nprime_MS_received).digest(), sig, '
      sha256'):
11   raise Error('Unable to verify signature of server challenge, aborting')
12 #Challenge passed, connection ready for use
13 self.aes_send(Nprime_MS_received)
```

Listing 3. Excerpt of Python implementation of the server.

`Kpriv_D`, and in line 3, the signature of the message is verified using the public key of the server `K_S`. If the signature matches, the message components are extracted in line 4: the `access_received` boolean value that tells the door whether or not the user has been granted access to the door by the server, and `N_SD_returned` that is a nonce included to prevent replay attacks by the mobile unit. In line 5, the nonce is compared with the expected value, and only if it matches is the event `doorUnlockMessageReceived` triggered in line 6.

In an earlier version of the protocol, the check in line 5 was not present. This caused the protocol to be vulnerable to replay attacks allowing the user access to doors where his access had been revoked. However, by verifying the query in Listing 1, ProVerif was able to find this mistake allowing for its correction.

6. Implementation of the Protocol

Based on the ProVerif model, a working prototype implementation was created to test the viability for use in an actual end-user product. The server and door units were programmed in Python and the mobile unit was an Android smartphone with a custom application written in Java and C. The level of detail in the protocol specification allowed for an implementation that has a close correspondence with the ProVerif model.

Listing 2 shows a code excerpt from the ProVerif model from the part of the model where the server entity communicates with the mobile device in order to verify the mobile

device authenticity. Listing 3 shows the corresponding part of the server implementation in Python, where AES is used for symmetric encryption and RSA is used for public key encryption. In the Python implementation the IVs and keys are hidden within the `aes_send` and `aes_recv` methods, but are otherwise the same as sending and receiving messages with the `symEncryptWithMAC` and `symDecryptWithMAC` functions through the Internet channel in the ProVerif model. Since the type system in ProVerif is different from that of Python, the messages in Python are merely strings, and therefore the length of each message is also verified.

7. Conclusion

In this paper, a protocol was presented that allows secure communication over insecure Bluetooth and Internet connections between door locks and a central server. By using only the Internet connection of a smartphone, the cost of deploying an electronic door lock solution can be minimized, especially for large buildings and institutions. Use cases include access control at companies, universities, and for home care.

The protocol is designed with implementation in mind and for real-world use. It is explained as a protocol narration and formalized using applied π-calculus. The detailed formalization of the protocol allowed us to verify a set of properties using the cryptographic protocol verifier ProVerif. The protocol allows companies such as MVC-Data to improve security for their door lock units, by using the formalization to implement the protocol in a consumer product. By integrating the protocol in their current product range MVC-Data will also be able to address what should happen in (the rare) case where a connection via 3G or Wi-Fi is flaky. In that case the protocol could default to an improved variant of their current protocol where (a limited number of) authorized users credentials are cached in the door unit. The caching of user credentials could even piggy-bag on communication in the protocol proposed in this paper when 3G or Wi-Fi is working correctly. Future work clearly includes a thorough analysis of such a combined protocol.

The current implementations of the protocol assume that both Bluetooth and 3G/Wifi is constantly turned on. To save energy on the mobile device the application could ask the user to turn the relevant radios on and off. However, that would defeat some of the purpose of the application as it currently does not require interaction from the user. The user can keep the mobile device in a pocket and as soon as the user approaches a door, for which the user has authorized access, the system will unlock the door. To preserve this property and save energy, it would make sense to extend the protocol to use location information to make decision on when to turn on and off the relevant radios, i.e. only when the user approaches a door will the relevant radios be turned on. It is a case for future studies to investigate the feasibility and energy efficiency of this extension.

References

[1] MVC-Data ApS. (2012, January 11th) Wireless Bluetooth access control. [Online]. Available: http://www.mvc-data.com/Home.html

[2] BlueLon. (2011, May 24th) BlueAccess BAL-100-BL. [Online]. Available: http://www.bluelon.com/index.php?id=248

[3] FlexiPanel. (2007, March 1st) BlueLock - Access control triggered by Bluetooth on your mobile phone. [Online]. Available: http://www.flexipanel.com/Docs/BlueLock%20DS377%20Cover.pdf

[4] Steab AB. (2012, January 11th) Blue Step. [Online]. Available: http://steab.se/2bluestep.html

[5] ECKey. (2011, May 24th) ECKey - turn your phone into a key! [Online]. Available: http://www.eckey.com

[6] SOREX - Wireless Solutions GmbH. (2012, January 11th) SOREX wireless products. [Online]. Available: http://www.sorex-austria.com/overview_wireless.html

[7] E. R. Wognsen, H. S. Karlsen, M. Calverley, and M. N. Follin. (2011, May 27th) A proposal for a secure relay protocol for door access control. Student report at Aalborg University. [Online]. Available: http://sw8.lmz.dk/report.pdf

[8] M. Abadi, "Private authentication," in *Proceedings of the 2nd international conference on Privacy enhancing technologies*, ser. PET'02. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 27–40.

[9] M. Abadi and A. D. Gordon, "A calculus for cryptographic protocols," *Inf. Comput.*, vol. 148, pp. 1–70, January 1999.

[10] B. Blanchet. (2011, December 11th) ProVerif. [Online]. Available: http://www.proverif.ens.fr

[11] S. Kremer and M. Ryan, "Analysis of an electronic voting protocol in the applied pi calculus," in *In Proc. 14th European Symposium On Programming (ESOP'05), volume 3444 of LNCS*. Springer, 2005, pp. 186–200.

[12] K. Bhargavan, C. Fournet, R. Corin, and E. Zalinescu, "Cryptographically verified implementations for TLS," in *Proceedings of the 15th ACM conference on Computer and communications security*, ser. CCS '08. New York, NY, USA: ACM, 2008, pp. 459–468. [Online]. Available: http://doi.acm.org/10.1145/1455770.1455828

[13] R. Chang and V. Shmatikov, "Formal analysis of authentication in bluetooth device pairing," 2007.

[14] B. Blanchet. (2011, December 11th) ProVerif users. [Online]. Available: http://www.proverif.ens.fr/proverif-users.html

[15] M. Abadi and C. Fournet, "Mobile values, new names, and secure communication," in *Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, ser. POPL '01. New York, NY, USA: ACM, 2001, pp. 104–115. [Online]. Available: http://doi.acm.org/10.1145/360204.360213

[16] R. Milner, *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, 1999.

[17] E. R. Wognsen, H. S. Karlsen, M. Calverley, and M. N. Follin. (2012, Feb) Protocol proverif model. [Online]. Available: http://sw8.lmz.dk/protocol.pv

[18] Woo and S. Lam, "A semantic Model for Authentication Protocols," in *The 1993 Symposium on Research in Security and Privacy*. IEEE Computer Society Press, May 1993, pp. 178–194.

