Formal Analysis of a Privacy-Preserving Billing Protocol

Alessandro Armando[1,2], Roberto Carbone[2], and Alessio Merlo[1,3]

[1] DIBRIS, Università degli Studi di Genova, Italy
{alessandro.armando,alessio.merlo}@unige.it
[2] Security & Trust Unit, FBK-irst, Trento, Italy
{armando,carbone}@fbk.eu
[3] Università e-Campus, Italy
alessio.merlo@uniecampus.it

Abstract. We provide a formal model and a security analysis of the Private Billing Protocol. This formal analysis allowed us to spell out precisely the details of the protocol, the security assumptions as well as the expected security goals. For the formal analysis we used SATMC, a model checker for security protocol analysis that supports the specification of security assumptions and goals as LTL formulae. Further analysis that we conducted manually revealed that the protocol allows for implementations that fail to meet the expected privacy goal. We describe the implications of our findings and discuss how the problem can be avoided.

Keywords: Privacy, Smart Meters, Billing Protocol, Formal Analysis.

1 Introduction

Smart Metering solutions are receiving growing attention as they support the automatic collection of fine-grained consumption profiles of utilities (e.g. electricity) thereby paving the way to sophisticated pricing schemes based, e.g., on time or load. However, the adoption of Smart Metering solutions is to date controversial due to privacy concerns related to the release of consumption profiles to suppliers [1]. In order to address these concerns a number of security protocols for smart metering have been put forward [2–4]. A Private Billing Protocol (PBP) has been recently proposed [5]. By using a form of homomorphic encryption, the protocol allows the Smart Meter to deliver the price associated with the consumption profile without revealing the latter to the supplier.

In this paper we provide a formal model and a security analysis of PBP. The formal analysis of the protocol allowed us to spell out precisely the details of the protocol, the assumptions that are necessary for their proper functioning (namely the trust assumptions and the assumptions on the communications channels used by the protocol to transport the messages), and the security goals that the protocol is expected to meet. For the formal analysis of the protocol we used SATMC [6, 7], a model checker for security protocol analysis that supports the specification of security assumptions and goals as LTL formulae. Further analysis

J. Cuellar (Ed.): SmartGridSec 2012, LNCS 7823, pp. 108–119, 2013.

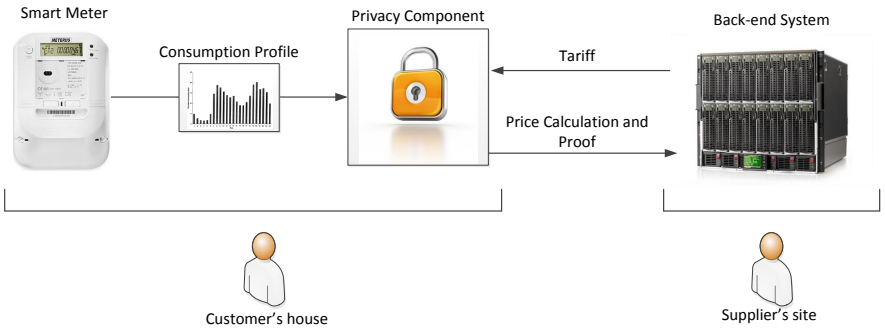


Fig. 1. Overview of the Private Billing Protocol

of the protocol that we conducted manually revealed that the protocol allows for implementations that fail to meet the expected privacy goal. We describe the implications of our findings and discuss how the problem can be avoided.

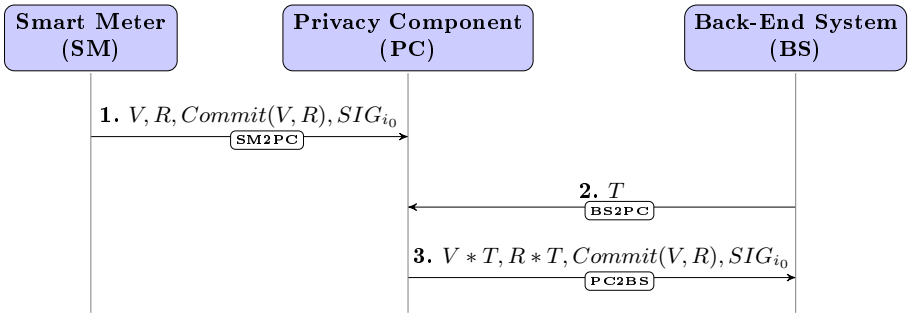
Structure of the paper. In the next section we introduce PBP. In Section 3 we present the formal model of the protocol and in Sections 4 and 5 we discuss the results of the security analysis. In Section 6 we provide an overview of the related work and we conclude in Section 7 with some final remarks.

2 The Private Billing Protocol

The Private Billing Protocol (hereafter PBP) has been proposed by Jawurek et al. [5] as a protocol for billing customers in a privacy-preserving way. The aim of PBP is to allow the supplier to charge customers for energy consumption keeping the fine-grained consumption profile of the customer private (i.e. not disclosing it to the supplier or other external stakeholders).

The PBP (see Fig. 1) involves three entities: the Smart Meter (SM), a Privacy Component (PC) and the Back-End System (BS). SM is installed at the customer's house and registers the *consumption profile*, i.e. a set of energy consumption values registered at short-length intervals (typically from 10 to 20 minutes). PC is a plug-in component placed in between SM and BS; it intercepts the consumption profiles produced by SM and the tariffs from BS and calculates the billing without disclosing the consumption profile to BS. BS provides tariffs to PC and receives the billing calculated by PC. The protocol allows BS to verify the billing produced by PC without knowing the consumption profile of the customer. This is achieved through the homomorphic commitment scheme proposed in [8]. Given two values x and r, a commitment for x and r, in symbols $\text{Commit}(x, r)$, is a pseudo-random, one-way function enjoying the following properties:

1. Given $\text{Commit}(x, r)$, it is hard to compute x (*Secrecy*);



Legend:
A $\xrightarrow{m}_{ch}$ B: A sends the message m to B on the communication channel ch.

Fig. 2. Message sequence chart of the Private Billing Protocol

2. Given $\text{Commit}(x,r)$, x and r, it is hard to compute $x' \neq x$ such that $\text{Commit}(x',r) = \text{Commit}(x,r)$ (*Binding*);
3. $\text{Commit}(x,r) * \text{Commit}(y,s) = \text{Commit}(x+y, r+s)$; and
4. $\text{Commit}(x,r)^y = \text{Commit}(x*y, r*y)$.

The PBP consists of the three steps shown in Fig. 2. In the first step of the protocol SM sends PC:

- the consumption profile $V = \langle v_{i_0}, \ldots, v_{i_n} \rangle$, where v_{i_k} is the consumption in the interval i_k for $k = 0, \ldots, n$,
- a vector $R = \langle r_{i_0}, \ldots, r_{i_n} \rangle$ of random numbers,
- a vector of commitments $Commit(V,R) = \langle \text{Commit}(v_{i_0}, r_{i_0}), \ldots, \text{Commit}(v_{i_n}, r_{i_n}) \rangle$, and
- a signature SIG_{i_0} over $\langle i_0, Commit(V,R) \rangle$, where i_0 is the interval of the first measurement in V. The signature is calculated using the private key of SM.

In step 2, BS sends PC the tariff vector T. Finally, in step 3 PC computes the price $V * T = \sum_{k=0}^{n} v_{i_k} * t_{i_k}$ associated with the consumption vector V, the value $R * T = \sum_{k=0}^{n} r_{i_k} * t_{i_k}$, and then sends $V * T$, $R * T$, $Commit(V,R)$ and SIG_{i_0} to BS. At this point BS verifies the validity of the signature and whether the commitments in $Commit(V,R)$ correspond to a commitment of the price computed by PC by checking whether

$$\text{Commit}(v_{i_0}, r_{i_0})^{t_{i_0}} * \cdots * \text{Commit}(v_{i_n}, r_{i_n})^{t_{i_n}} = \text{Commit}(V * T, R * T) \quad (1)$$

This completes the protocol.

Security Assumptions
SM and BS must agree on some signature scheme. This allows SM to sign the commitments and BS to check the integrity and authenticity of the signed messages. Moreover, SM is assumed to be trustworthy. SM is equipped with a Trusted Platform Module to ensure that the reported consumption profiles are trustworthy and reliable. The communication channels used to transport the messages exchanged among the involved parties must offer some level of security. However, the necessary security assumptions are not explicitly stated in [5]. In Section 4 we consider different assumptions, analyzing the consequences of the different choices from a security perspective.

Security Requirements
The purpose of the protocol is to fulfill the security requirements of both the customer and the supplier.

- *Supplier Requirements.* BS must be sure that the price provided by PC is the correct one, i.e. that it has been calculated using the proper consumption values and tariffs.
- *Customer Requirements.* PC must be sure to compute the proper price, i.e. the consumption vector V is the one provided by SM and the tariff is the one provided by BS. Moreover, the customer has some privacy concerns: V must be kept secret between SM and PC.

In this paper, we focus on the latter.

3 Formal Modeling of the Private Billing Protocol

We focus on the problem of determining whether the concurrent execution of a finite number of sessions of the protocol enjoys the expected security properties in spite of the interference of a malicious intruder. In particular, we specified PBP using ASLan [9], one of the specification languages developed in the context of the AVANTSSAR Project [10] (`www.avantssar.eu`). For the sake of brevity in this paper we present a simplified version of ASLan, featuring only the aspects of the language that are relevant for this work. ASLan supports the specification of model checking problems of the form $M \models \phi$, where M is a labeled transition system modeling the behaviors of the honest principals and of the Dolev-Yao intruder (DY)[1][11] and their initial state, and ϕ is a Linear Temporal Logic (LTL) formula stating that the expected security properties hold provided that the exchange of messages enjoys the level of protection guaranteed by the transport protocols (See [12] for the details).

The states of M are sets of ground (i.e. variable-free) *facts*, i.e. atomic formulae of the form given in the left column of Table 1 and whose informal meaning is explained in the right column. The facts $\texttt{authentic}(c, p)$ and $\texttt{confidential}(c, p)$

[1] A Dolev-Yao intruder has complete control over the network and can generate new messages both from its initial knowledge and the messages exchanged over the network.

Table 1. Facts and their informal meaning

Fact	Meaning
$\texttt{state}_r(j, a, es, s)$	Principal a, playing role r, is ready to execute step j in session s of the protocol, and es is a list of expressions representing the internal state of a.
$\texttt{ik}(m)$	The intruder knows message m.
$\texttt{sent}(rs, b, a, m, c)$	Principal rs has sent message m on channel c to principal a pretending to be principal b.
$\texttt{rcvd}(a, b, m, c)$	Message m (supposedly sent by principal b) has been received on channel c by principal a.
$\texttt{authentic}(c, p)$	Channel c provides authenticity to p, i.e. its input is exclusively producible by a specified sender p.
$\texttt{confidential}(c, p)$	Channel c provides confidentiality to p, i.e. its output is exclusively accessible to a given receiver p.

trigger LTL formulae stating the property of the channels. More details can be found in [7, 12].

The initial state of the system defines the initial knowledge of the intruder (usually including its cryptographic material, various agent identifiers, and their public keys) and the initial state of all the honest principals involved in the considered protocol sessions.

Transitions are represented by *rewrite rules* of the form $(L \xrightarrow{rn(var_1,\ldots,var_n)} R)$, where L and R are finite sets of facts, rn is a *rule name*, i.e. a function symbol uniquely associated with the rule, and $var_1, \ldots, var_n$ are the variables occurring in L. Here and in the sequel we use typewriter font to denote states and rewrite rules with the additional convention that variables are capitalized (e.g. PC, T0), while constants and function symbols begin with a lower-case letter (e.g. pc, t0). The constant i is used to denote the intruder.

The behavior of honest participants is specified by rules of the form:

$$\texttt{sent}(\texttt{RS}, \texttt{B}, a, \texttt{M}, \texttt{Ch}) \xrightarrow{\texttt{receive(B,RS,M,Ch)}} \texttt{rcvd}(a, \texttt{B}, \texttt{M}, \texttt{Ch}) \tag{2}$$

$$\begin{aligned} \texttt{rcvd}(a, b, m, ch)\texttt{.state}_r(j, a, es, \texttt{S}) &\xrightarrow{\texttt{send}_j\texttt{(S,...)}} \\ \texttt{sent}(a, a, b', m', ch')\texttt{.state}_r(l, a, es', \texttt{S}) & \end{aligned} \tag{3}$$

for all honest principals a and suitable terms b, b', ch, ch', es, es', m, and m'. Rule (2) models the reception of a message by an honest agent, whereas rule (3) models the processing of a previously received message and the sending of the next protocol message. More in detail, rule (3) states that if principal a is at step j in session S of the protocol and she has received message m on channel ch (supposedly) by b, then she can send message m' to b' on channel ch' and change her internal state accordingly preparing for step l.

To simplify the presentation in the sequel we use the infix notation for the arithmetic operators instead of the prefix one as required in ASLan. We define

them, together with the commit binary operator $\mathtt{c}$, as non invertible public functions. Moreover $\langle m_1, \ldots, m_k \rangle$ stands for $\mathtt{pair}(m_1, \mathtt{pair}(\ldots, m_k))$. To illustrate we consider the protocol steps 2 and 3, in which PC receives the tariff values from BS and it sends back the last message to BS. This transition is modeled by the following rule, for any positive integer n:

$$\mathtt{rcvd}(\mathtt{PC}, \mathtt{BS}, \langle \mathtt{T0}, \ldots, \mathtt{T}n \rangle, \mathtt{BS2PC}) \,.\, \mathtt{state}_{\mathtt{PC}}(2, \mathtt{PC}, [\mathtt{BS}, \mathtt{BS2PC}, \mathtt{PC2BS} \ldots], \mathtt{S})$$
$$\xrightarrow{\mathtt{send}_2(\mathtt{PC},\ldots,\mathtt{S})}$$
$$\mathtt{sent}(\mathtt{PC}, \mathtt{PC}, \mathtt{BS}, \langle (\mathtt{V0} * \mathtt{T0}) + \ldots + (\mathtt{V}n * \mathtt{T}n), (\mathtt{R0} * \mathtt{T0}) + \ldots + (\mathtt{R}n * \mathtt{T}n),$$
$$\mathtt{c}(\mathtt{V0}, \mathtt{R0}), \ldots, \mathtt{c}(\mathtt{V}n, \mathtt{R}n), \mathtt{sign}(\mathtt{inv}(\mathtt{pk}(\mathtt{SM})), \langle \mathtt{I}, \mathtt{c}(\mathtt{V0}, \mathtt{R0}), \ldots, \mathtt{c}(\mathtt{V}n, \mathtt{R}n) \rangle) \rangle, \mathtt{PC2BS}) \,.$$
$$\mathtt{state}_{\mathtt{PC}}(3, \mathtt{PC}, [\mathtt{BS}, \mathtt{BS2PC}, \mathtt{PC2BS}, \mathtt{T0}, \ldots, \mathtt{T}n, \ldots], \mathtt{S})$$

where $\mathtt{sign}(\mathtt{inv}(\mathtt{pk}(\mathtt{SM})),m)$ is the digital signature of the message m with the private key of $\mathtt{SM}$, and $\mathtt{BS2PC}$ and $\mathtt{PC2BS}$ are the channels used by $\mathtt{BS}$ and $\mathtt{PC}$ to communicate with each other. (Channels in our model are unidirectional: if x and y are principals, then $x2y$ denotes a channel used by x to send messages to y and $y2x$ denotes a channel used by y to send messages to x.)

Since SATMC does not support reasoning about homomorphic encryption we replaced the check of equation (1) with pattern matching. Namely, in the left hand side of the last rule of BS, we explicitly state which pattern must have the message received in order to be accepted by BS. For each admissible pattern, we add a similar rule leading to the same final state. As we will see, this level of abstraction is enough to detect the issues reported in Section 4. Moreover, in Section 5 we point out which kind of attacks can be lost.

For the sake of brevity, we omit here the rules modeling the abilities of the DY intruder. He can overhear and divert messages, and by using the knowledge gleaned from the observed traffic he forges and sends fraudulent messages to the honest participants.

3.1 Security Requirements

The use of LTL allows for the specification of the security goals of the protocol. The language of LTL we consider here uses facts as atomic formulae, the usual propositional connectives (namely, $\vee$, $\wedge$, $\Rightarrow$), and the temporal operators **G** (globally) and **O** (once).

Let us focus on the customer's requirements that the PBP is expected to enjoy (cf. end of Section 2). They can be expressed in terms of authentication and secrecy properties, defined as follows:

Authentication. To define the authentication goals of the protocol we rely on the definition given in [13]: whenever principal b (playing role r_2) completes a run of the protocol apparently with principal a (playing role r_1), then *(i)* a has previously been running the protocol apparently with b, and *(ii)* the two agents

agree on m. We then say that b *authenticates* a *on* m *in session* s if and only if the following formula holds:

$$\begin{aligned} &\mathit{authentication}(b, a, m, s) := \\ &\qquad \mathbf{G}\,\forall(\mathtt{state}_{r_2}(j_2, b, [a, \ldots, m, \ldots], s) \Rightarrow \\ &\qquad\qquad \exists\,\mathbf{O}\,\mathtt{state}_{r_1}(j_1, a, [b, \ldots, m, \ldots], s)) \end{aligned}$$

where j_1 is the protocol step in which m is sent by an agent playing role r_1 and j_2 is the last protocol step of an agent playing role r_2.

Secrecy. The secrecy of a message m is guaranteed if, and only if, the intruder does not know m. This is formalised by the following formula:

$$\mathit{secret}(m) := \mathbf{G}\,\neg\,\mathtt{ik}(m)$$

Thus, for each protocol session s, this amounts to including the following formulae in ϕ:

$$\begin{aligned} \mathit{authentication}(pc, sm, v, s) \\ \mathit{authentication}(pc, bs, t, s) \\ \mathit{secret}(v) \text{ provided that } \mathtt{i} \notin \{sm, pc\} \end{aligned}$$

where sm, pc, and bs are the agents playing in session s the roles SM, PC, and BS respectively, while v and t are the data values of V and T as exchanged by the agents in s. (Clearly, the intruder is entitled to access v if, and only if, it is playing in s one of those agents that is legitimated to do so i.e., the intruder is either sm or pc.)

4 Formal Security Analysis

We have mechanically analyzed the formal model of PBP presented in Section 3 using SATMC, a state-of-the-art model checker for security protocols. At the core of SATMC lies a procedure that automatically generates a propositional formula whose satisfying assignments (if any) correspond to counterexamples (i.e. execution traces of M that falsify ϕ) of length bounded by some integer k. Finding attacks (of length k) on the protocol therefore boils down to solving propositional satisfiability problems. SATMC relies on state-of-the-art SAT solvers for this task which can handle propositional satisfiability problems with hundreds of thousands of variables and clauses or more. SATMC can be instructed to perform an iterative deepening on k. More details on SATMC can be found in [14, 7].

In our analysis, we set the number n of time intervals to 2. This value is sufficient to spot issues, thereby maintaining the problem concise. We focused on the customer's requirements and considered a variety of protocol scenarios. Among them, we selected two significant scenarios. In both of them SM and PC

are assumed to be trustworthy. In the first scenario, BS is trustworthy as well, and he is only vulnerable to passive attacks. The rationale is that active attacks are much more difficult to perform (e.g. for a malicious employee), and they would be easy to detect, if they have a certain dimension, because customers would be paying strange bills with unexpected amounts. In the second scenario, we assume that BS is not trustworthy and can misbehave. Even considering this scenario, the customer's requirements should be guaranteed.

In PBP the assumptions on the properties of the communication channels between the agents are not explicitly stated, thus we checked different channel configurations. Moreover, we modeled the customer's requirements by using the secrecy and authentication properties described in Section 3.1. We discuss them in turn.

secret(v)

We considered an initial scenario comprising two sessions in which the roles of the protocol are played by trustworthy agents (namely `sm`, `pc`, and `bs`) and in which all messages are exchanged over insecure channels. As expected, SATMC detected an attack to the secrecy property *secret*(v). Indeed, it is easy to see that in this scenario the intruder can overhear the message sent by `sm` to `pc` on channel `sm2pc`, thereby learning the consumption value `v`. This indicates that the communication channel `sm2pc` used to transport the messages from SM to PC must ensure confidentiality. Thus, we added the fact `confidential(sm2pc,pc)` to the initial state of the protocol and fed the resulting specification to the model checker. SATMC did not find any attack when applied to the resulting model. It must be noted that since BS is not necessarily trustworthy, the secrecy property must hold even when the intruder plays the role of BS. We therefore changed the model of the protocol accordingly. Also in this case SATMC did not find any violation on the secrecy property.

authentication(`pc,sm,v,s`)

Assuming that PC is able to verify the validity of the signature of SM, SATMC did not report any attack on this property. Indeed, in the first step of the protocol, the vector of commitments is explicitly signed by SM, as well as the interval i_0, thus allowing PC to check the authenticity of the values.

authentication(`pc,bs,t,s`)

The protocol is clearly violated if BS is malicious. We therefore focused only on scenarios in which all the protocol participants are trustworthy. SATMC reported the following attack: a malicious agent `i`, pretending to be `bs`, can send to `pc` a fake value for the tariff vector `t`. The consequence is that `pc` can be induced to produce a wrong bill. This indicates that the communication channel used to transport the messages from BS to PC must ensure authenticity of the messages. By adding the fact `authentic(bs2pc,bs)` on the initial state, SATMC did not find any violation on the authentication property.

5 Further Security Considerations

As shown above, model-checkers are helpful for detecting security weaknesses in security protocols. However the results of analysis must be interpreted carefully by considering the limitations of the tools. SATMC does not currently support reasoning about the algebraic properties of the cryptographic operators and therefore it may fail in detecting attacks that rely strictly on them.

By manual inspection of the protocol we realized that, in case BS is not trustworthy, he can acquire the whole consumption profile of the customer. Let n be the number of time intervals for each consumption profile. By sending a sequence of n fake tariff vectors $T_1, \ldots, T_n$ to PC, BS receives the corresponding prices $p_1, \ldots, p_n$. Given that the prices are computed by PC using the same consumption profile V, then $V * T_i = p_i$ for $i \in [1..n]$, and BS can determine V, by solving a system of n equations.

This issue could be prevented by changing the protocol as follows. The message 2 in Fig. 2 should be enriched including the interval i_0 of the first measurement in V. When receiving that message, PC must check whether i_0 is the one received from SM in step 1. Moreover, if PC obtains several requests containing the same i_0, he must only repeat his previous answer. Notice that, BS can still forge and send PC a fake tariff vector $T'_k = \langle t'_1, \ldots, t'_n \rangle$ such that $t'_k = 1$ and $t'_i = 0$ for all $i \neq k$, for $k \in [1..n]$. It is easy to see that in this case the price computed by PC and sent to BS is equal to v_k. Nevertheless, using the solution proposed, it is possible to control the amount of information leakage: BS can obtain at most a single consumption value v_k for each time frame, while it is prevented to acquire the whole consumption profile.

6 Related Work

Smart Metering is a privacy-sensitive scenario, due to the involvement of (potentially untrusted) third-party aggregators which gather and analyze streams of data [15]. In general, the goal with smart metering is to allow aggregators to infer less information as possible on the behavior of customer while retaining the ability to deliver the expected functionalities (e.g. billing). To this aim, recent literature provides different cryptographic schemes aimed at preserving the customer privacy from untrusted aggregators and under different scenario assumptions.

One of the first privacy-preserving scheme for smart metering has been proposed by Rial and Denezis [16] and has been enhanced and applied to different contexts. In this paper, we formally have analyzed the privacy-preserving billing protocol described in [5], based on Pedersen Commitments and Zero-Knowledge Proof as a basis to provide verification. Also in [2] Zero-Knowledge Proof is used in a billing protocol specifically optimized for low consumption. In [17] authors propose a scheme able to allow aggregators to get data from different

smart meters in a privacy-preserving way: each customer may challenge the aggregator to prove that his privacy has been preserved. Moreover, the same scheme is robust against failures of the involved smart meters. Furthermore, in [4] the authors propose a distributed scheme for differential privacy preservation that does not rely on trusted third parties. Schemes for detecting attempts to violate the consumer privacy have also been defined. For instance, in [18] a scheme for the detection of leakage and fraud attempts is presented. Finally, in [3] schemes for allowing spatial (i.e. data coming from different smart meters at the same time) and temporal (i.e. data from different periods in the same smart meter) privacy-preserving aggregation are discussed.

The efficacy of all previous schemes are proved against an adversarial model aimed at breaking the cryptographic scheme. However, to the best of our knowledge, no systematic and formal analysis considering the interleaving of multiple protocol sessions among protocol participants and the intruder have ever been applied.

Concerning formal analysis, besides SATMC, the AVANTSSAR Platform features two other model-checkers for security protocols, namely CL-AtSe [19], and OFMC [20]. Both CL-AtSe and OFMC support reasoning about the algebraic properties of cryptographic operators. This is very helpful to detect flaws in a variety of protocols, but unfortunately homomorphic encryption is not yet supported by the tools. It must be noted that this is very active research area. For instance, the decidability of unification modulo homomorphic encryption has been proved only recently [21]. On the contrary, SATMC supports model checking of LTL formulae, allowing for the specification of assumptions on principals and communication channels as well as of complex security properties that are normally not handled by state-of-the-art security protocol analysers.

7 Conclusions

In this paper we provided a preliminary formal analysis of a smart metering protocol (the Private Billing Protocol), also discussing security issues related to different assumptions that may hold in actual implementations. We showed that—under certain assumptions—the security of the protocol can be violated.

Acknowledgments. We are grateful to Jorge Cuéllar for stimulating discussions on these topics and the feedback he provided. This work has partially been supported by the activity "TSES 12178 SESSec-EU - Networked Smart Energy Systems Security in Europe" of the action line TSES Smart Energy Systems of the EIT ICT Labs, by the FP7-ICT Project SPaCIoS (no. 257876), and by the project SIAM funded in the context of the FP7 EU "Team 2009 - Incoming" COFUND action.

References

1. Heck, W.: Smart energy meter will not be compulsory. NRC Handelsblad (April 2009)
2. Molina-Markham, A., Danezis, G., Fu, K., Shenoy, P., Irwin, D.: Designing privacy-preserving smart meters with low-cost microcontrollers. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 239–253. Springer, Heidelberg (2012)
3. Erkin, Z., Tsudik, G.: Private computation of spatial and temporal power consumption with smart meters. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 561–577. Springer, Heidelberg (2012)
4. Ács, G., Castelluccia, C.: I have a dream!: differentially private smart metering. In: Filler, T., Pevný, T., Craver, S., Ker, A. (eds.) IH 2011. LNCS, vol. 6958, pp. 118–132. Springer, Heidelberg (2011)
5. Jawurek, M., Johns, M., Kerschbaum, F.: Plug-in privacy for smart metering billing. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 192–210. Springer, Heidelberg (2011)
6. Armando, A., Compagna, L.: SATMC: a SAT-based model checker for security protocols. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 730–733. Springer, Heidelberg (2004)
7. Armando, A., Carbone, R., Compagna, L.: LTL Model Checking for Security Protocols. In: Journal of Applied Non-Classical Logics, Special Issue on Logic and Information Security, Hermes Lavoisier, pp. 403–429 (2009)
8. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
9. AVANTSSAR. Deliverable 2.1: Requirements for modelling and ASLan v.1 (2008), `http://www.avantssar.eu`
10. Armando, A., Arsac, W., Avanesov, T., Barletta, M., Calvi, A., Cappai, A., Carbone, R., Chevalier, Y., Compagna, L., Cuéllar, J., Erzse, G., Frau, S., Minea, M., Mödersheim, S., von Oheimb, D., Pellegrino, G., Ponta, S.E., Rocchetto, M., Rusinowitch, M., Torabi Dashti, M., Turuani, M., Viganò, L.: The AVANTSSAR Platform for the Automated Validation of Trust and Security of Service-Oriented Architectures. In: Flanagan, C., König, B. (eds.) TACAS 2012. LNCS, vol. 7214, pp. 267–282. Springer, Heidelberg (2012)
11. Dolev, D., Yao, A.: On the Security of Public-Key Protocols. IEEE Transactions on Information Theory 2(29) (1983)
12. Armando, A., Carbone, R., Compagna, L., Cuellar, J., Abad, L.T.: Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps. In: Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008). ACM Press, Hilton Alexandria Mark Center (2008)
13. Lowe, G.: A Hierarchy of Authentication Specifications. In: Proceedings of the 10th IEEE Computer Security Foundations Workshop (CSFW 1997), pp. 31–43. IEEE Computer Society Press (1997)
14. Armando, A., Compagna, L.: SAT-based Model-Checking for Security Protocols Analysis. International Journal of Information Security 7(1), 3–32 (2008)
15. Shi, E., Chan, H., Rieffel, E., Chow, R., Song, D.: Privacy-preserving aggregation of time-series data. In: Proc. of the 18th Annual Network and Distributed System Security Symposium (NDS 2011). National Science Foundation Expeditions in Computing (2011)

16. Rial, A., Danezis, G.: Privacy-preserving smart metering. In: Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society (WPES 2011), pp. 49–60. ACM, New York (2011)
17. Chan, T.-H.H., Shi, E., Song, D.: Privacy-preserving stream aggregation with fault tolerance. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 200–214. Springer, Heidelberg (2012)
18. Kursawe, K., Danezis, G., Kohlweiss, M.: Privacy-friendly aggregation for the smart-grid. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 175–191. Springer, Heidelberg (2011)
19. Turuani, M.: The CL-Atse Protocol Analyser. In: Pfenning, F. (ed.) RTA 2006. LNCS, vol. 4098, pp. 277–286. Springer, Heidelberg (2006)
20. Basin, D., Mödersheim, S., Viganò, L.: OFMC: A symbolic model checker for security protocols. International Journal of Information Security 4(3), 181–208 (2005)
21. Anantharaman, S., Lin, H., Lynch, C., Narendran, P., Rusinowitch, M.: Unification modulo homomorphic encryption. In: Ghilardi, S., Sebastiani, R. (eds.) FroCoS 2009. LNCS, vol. 5749, pp. 100–116. Springer, Heidelberg (2009)