A Secure and Private Billing Protocol for Smart Metering

Tom Eccles and Basel Halak
Electronics and Computer Science Department, Southampton University
Email: {tde1g14, bh9}@ecs.soton.ac.uk

***Abstract*—Traditional utility metering is to be replaced by smart metering. Smart metering enables very fine grained utility consumption measurements. These fine grained measurements raise privacy concerns due to the lifestyle information which can be inferred from the precise time at which utilities were consumed. This paper outlines two privacy respecting time of use billing protocols for smart metering. These protocols protect the privacy of customers by never transmitting the fine grained utility readings outside of the customer's home network. One protocol favours complexity on the trusted smart meter hardware while the other uses homorphic commitments to offload computation to a third device. Both protocols are designed to operate on top of existing cryptographic secure channel protocols in place on smart meters. Proof of concept software implementations of these protocols have been written and their suitability for real world application to low performance smart meter hardware is discussed. These protocols may also have application to other privacy conscious aggregation systems such as electronic voting.**

I. Introduction

A. Motivation

Ever more objects are designed to be connected to each other and to the Internet as part of the "Internet of Things". Predictions indicate that 50 billion of these devices will be online by 2025 [1]. As part of this trend utility supply is in the process of upgrading to use network connected meters which allow frequent consumption measurements to be taken automatically. The network of these meters is called the "Smart Grid".

As discussed in a survey of smart metering [2], one advantage of these regular measurements is to improve load monitoring and forecasting so that utility re-sellers can make more profitable contracts with utility producers and so that utility providers can make more informed decisions about how to maintain their infrastructure. Another use is to detect network leakage and fraud. The regular consumption measurements can be used automatically to bill customers depending on complex tariffs such as varying the cost of a utility depending upon the time at which it was used. For these reasons smart meter roll out has been mandated by governments in the EU, UK and United States [2].

However, the transmission of these fine grained consumption measurements has raised significant opposition from both privacy advocates and ordinary consumers [3]. For example, fine grained household power consumption data make it easy to spot religious activities such as fasting and nighttime praying. As another example, a burglar could use this data to ensure that a target property is not occupied at the time of a break-in. For similar reasons, law enforcement agencies are interested in utility consumption data. Utility consumption measurements were first used by law enforcement in West Germany in the 1970s where police used energy consumption records to locate a Red Army safe house [2]. Very high frequency electricity consumption readings have been used to identify the TV program being watched on several TV models from different manufacturers [4].

In general there have been two responses to privacy concerns: the most widespread of these has been regulatory solutions. For example, in the UK Smart Energy Code [5], parties to the energy grid are required only to access the information they are authorised to use and this authorisation is only assigned to those parties who most need it. However, regulation is a weak guarantee because compromised or malicious parties may not abide by the regulation. A stronger privacy guarantee is given by technical solutions such as the cryptographic protocols put forward in this paper and others [3][6][7][8][9]. One downside of the technical solutions is that they often require smart meters to carry out complicated cryptographic calculations which necessitate more expensive smart meter hardware. Furthermore, controlling distribution of meter readings reduces the flexibility of the smart metering system. For example, under private billing systems, the energy provider loses the ability to sell the meter readings to data brokers and advertising companies.

B. This Paper

This paper outlines two cryptographic protocols. These protocols were designed to provide private and secure time of use billing while minimising the additional smart meter cost. Each of these protocols addresses different tradeoffs between technical complexity and smart meter cost. These billing protocols are designed to run over an authenticated and confidential communication channel such as TLS. Both protocols were primarily designed and written as software libraries which may be included into other projects (such as a vendor's specific implementation of smart metering for their product).

C. Paper Outline

Section II describes the smart meter billing security specification and the two protocols designed to solve this challenge.

Section III explains how the protocols meet these requirements. Section IV discusses the existing implementation and Section V evaluates the implementation's suitability for use in a real world smart meter deployment.

II. Smart Meter Billing

A. Security Specification

These protocols are intended to provide time of use pricing without incurring the privacy loss inherent in giving utility companies fine grained utility consumption information.

1) Requirements:

R1 Do not allow the utility provider (or any active or passive attacker outside the customer's LAN) to learn utility consumption measurements with any more temporal granularity then they learn in the traditional utility billing system.

R2 Do not leak the bill amount to any passive or active attacker.

R3 Utility providers must be able to verify that they receive the correct bill amount (according to the trust assumptions).

These requirements are met by calculating the bill on a device on the same network as the smart meter and then only reporting that aggregate bill to the utility company. This solution was previously proposed in [8], [3] and probably others. This paper proposes two protocols which differ in their method to verify that the bill was computed honestly from the meter readings: providing a different trade-off between computational complexity in different parts of the system.

2) Assumptions:

A1 The smart meter is tamper resistant and is trusted by all parties.

A2 The utility provider trusts its own servers to operate correctly.

A3 Unless otherwise specified, all communications in these protocols are to be performed over a confidential and authenticated channel.

A4 (Specific to the three party protocol) The customer trusts the "customer" device.

B. Simple Protocol

1) The smart meter records readings into its own memory.
2) If a price change is necessary, the provider sends a signed copy of the new price information to the smart meter.
3) On receiving new pricing information, the smart meter verifies the signature. If the signature is incorrect the smart meter discards the message. If the signature is correct the new pricing information is stored.
4) At the end of a billing period the smart meter calculates the bill and deletes the readings.
5) The smart meter signs the bill and sends it to the provider.
6) The provider verifies the signature on the bill. If it is valid the bill is charged to the customer by the provider. If the signature was invalid, the message is discarded.
7) If the provider receives no valid bill for a billing period then this is reported to human operators so that appropriate action may be taken.

One disadvantage of this simple protocol is that all of the protocol computation and bill calculation has to be calculated on trusted hardware. [8] suggests that smart meters would have to use expensive tamper-proof hardware and so it would be better if some of this computation could be offloaded to another device as in the three party protocol.

C. Three Party Protocol

This protocol depends upon the usage of additively and multiplicly homorphic commitments. For this purpose Pedersen Commitments over the set of integers modulo a prime have been used. The next section briefly reviews Pedersen commitments. For more information on Pedersen commitments see [10] and [11].

1) Pedersen Commitments Commitments are often used to claim knowledge of some data without revealing the data. For example, in a puzzle competition, a participant may publish a commitment to the solution of a puzzle so that if challenged at a later time, the participant can prove that they knew the solution of the puzzle at the time at which they published the commitment. Doing so will not reveal the solution to the puzzle to other participants until the time at which the commitment is opened.
   To protect against situations in which the data being committed to has low entropy[1], commitments are usually derived from both this data and some random material. Opening a commitment is performed by revealing the data committed to and any other information (such as the random salt) which is required to compute the commitment. Any observer may then compute a commitment using this data to check that the result is the same as the commitment previously published.
   The functional requirements for a commitment system lead to two important properties for commitments (for formal definitions see [10]):
   a) A commitment scheme is said to be computationally[2] binding if nobody can find more than one valid opening to the same commitment without performing excessive (impossible on modern hardware) computation.
   b) A commitment scheme is said to be computationally concealing if observers only learn a negligible amount about the data which was committed to using non-excessive computation on the commitment.

   In the usage of commitments in this protocol, two further properties are required (where $\mathrm{commit}(x)$ is a commitment on x):

[1] One attack against such a system could learn which data was committed to by trying to compute a commitment to all possible values for the data and checking which one had the same value as the one which was published. The addition of a sufficientlly long random salt makes this attack computationally infeasable because there would be too many possible inputs to exhaustively test.

[2] Some commitment schemes are information theoretically hiding or binding. This is a stronger guarantee than computational infeasibility. See [10].

a) The commitment is additively homorphic: an efficient operation $\oplus$ exists such that $\text{commit}(x) \oplus \text{commit}(y) = \text{commit}(x + y)$
b) The commitment is multiplically homorphic: an efficient operation $\otimes$ exists such that $\text{commit}(x) \otimes y = \text{commit}(x \cdot y)$

The proof of concept implementation uses Pedersen commitments[11] defined as follows:

- q is a 256-bit random prime
- p is a large (2048-4096 bit) prime of the form $n * q + 1$ such that q generates a subgroup of p.
- g and h are random members of the subgroup of $\mathbb{F}_p^*$ of order q
- x is the data to be committed to (an integer modulo q)
- r is the random salt (an integer modulo p [3]).

$\text{commit}_r(x) := h^x g^r \pmod p$
This scheme is computationally binding (so long as the commiter does not know the discrete logarithm of h to the base g) and information theoretically concealing [10].
Additive homorphism can be achieved by multiplying commitments:

$$\begin{aligned}\text{commit}_{r_1}(x_1) \oplus \text{commit}_{r_2}(x_2) :=& \\ \text{commit}_{r_1}(x_1) \cdot \text{commit}_{r_2}(x_2)& \\ = (h^{x_1} g^{r_1}) \cdot (h^{x_2} g^{r_2}) \pmod p& \\ = h^{(x_1+x_2)} g^{(r_1+r_2)} \pmod p& \\ = \text{commit}_{(r_1+r_2)}(x_1 + x_2)&\end{aligned}$$

Similarly, multiplicative homorphism can be achieved using exponentiation:

$$\begin{aligned}\text{commit}_r(x) \otimes y &:= (\text{commit}_r(x))^y \\ &= (h^x g^r)^y \pmod p \\ &= h^{xy} g^{ry} \pmod p \\ &= \text{commit}_{ry}(xy)\end{aligned}$$

2) Protocol Description
The three parties in the protocol are the tamper resistant smart meter, a proxy device hereby called the "customer hardware" and the utility provider's server. The customer hardware may be a hardware device purchased and operated by the customer such as a smart phone as suggested in [8] or a utility communication hub similar to that used in the UK smart utility system [5]; it may be any device which the customer is willing to trust with their utility readings and with a reliable communication channel with both the smart meter and the server. The customer hardware does not have to be trusted by the utility provider but is trusted by the customer *(A4)*.

[3] r is usually picked from the set of integers modulo q so that computation can remain within this subgroup, however, in the course of the protocol values for r become too large for this and so the more expensive computations across the whole of the group generated by p are allowed.

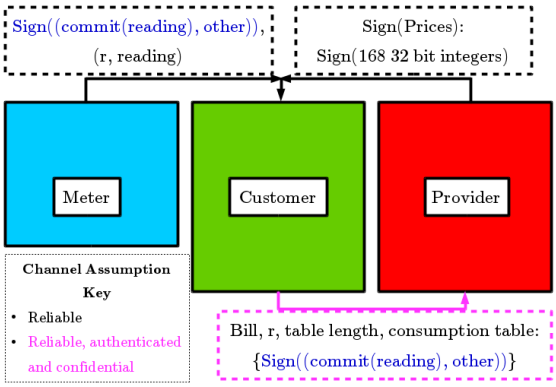

Fig. 1. Three Party Protocol Communication Channels

1) When the smart meter makes a reading it first calculates a commitment to the reading. This commitment is then concatenated to the "other" pricing information (for time of use billing: the time of the measurement[4]). The commitment, other tuple is signed by the smart meter. Finally, this signed tuple is sent to the customer hardware along with the salt used for the commitment and the clear-text reading.
2) On receiving this message from the smart meter, the customer hardware verifies the signature in the message. If the signature is valid the message is stored by the customer hardware. If the signature is invalid, the message is discarded.
3) If a price change is necessary, the provider sends a signed copy of the new price information to the customer hardware.
4) On receiving new pricing information, the customer hardware verifies the signature. If the signature is incorrect it discards the message. If the signature is correct the new pricing information is stored.
5) At the end of the billing period the customer hardware calculates the bill using its stored readings and prices. The salts of each reading are combined (in exactly the same way as the readings are combined when calculating the bill) to calculate the salt $\bar{r}$ for the bill verification. The bill, $\bar{r}$ and all of the signed tuples from the meter are sent to the utility provider.
6) On receiving all of this billing information, the provider's server must verify that the bill was calculated honestly by the customer hardware. First, the server should verify all of the signatures from the smart meter; so as to ensure that none of the meter readings or times were forged by the customer. The server should also check that all of the expected time measurements (and no others) are present in the billing information. This check prevents the

[4] This time must be unique within the life of the signing key otherwise a malicious customer could use lower readings from a previous billing cycle.

customer from replaying measurements from earlier billing cycles and from omitting some readings. Next, the server should calculate a commitment to the received bill, using the salt which it was sent. Finally, the utility provider can ensure that the bill was calculated honestly from the meter readings by using the homorphic commitments from the smart meter to calculate its own commitment to the final bill and then ensuring that this matches the expected commitment which it had already calculated. The homorphic properties are used as follows:

$$\begin{aligned}&(\text{commit}_{r_1}(x_1) \otimes p_1) \oplus (\text{commit}_{r_2}(x_2) \otimes p_2) \\ &\quad \oplus (\text{commit}_{r_3}(x_3) \otimes p_3) \oplus \dots \\ &\quad \oplus (\text{commit}_{r_N}(x_N) \otimes p_N) \\ &\quad = \text{commit}_{\bar{r}}\left(\sum_{i=1}^{N} x_i p_i\right) = \text{commit}_{\bar{r}}(\text{bill})\end{aligned}$$

III. Security Analysis

A. Simple Protocol

1) Customer Privacy: The privacy of meter readings *(R1)* is preserved by never transmitting the individual readings outside of the customer's home network. Only the aggregate bill is transmitted from the smart meter to the utility provider; and so for a sufficiently long billing cycle there is no change in the granularity of the disclosed readings between this protocol and the traditional utility billing system.

The privacy of the bill *(R2)* in transit is preserved by the confidential channel along which it is transmitted *(A3)*.

2) Utility Provider Bill Verification: The smart meter is assumed to be tamper-resistant *(A1)* and so the utility provider can trust its output. As the trusted smart meter has signed the bill, the utility provider (after verifying the signature) can be sure of the bill's integrity and origin. Therefore *R3* is met.

B. Three Party Protocol

1) Customer Privacy: Only the aggregate bill and commitments to the meter readings are transmitted to the utility provider. For a sufficiently long billing cycle there is no change in reading granularity between the bill in this system and the bill in the traditional utility billing system. The commitments do not reveal any information about the meter readings because Pedersen Commitments are information theoretically concealing [10]. Therefore *R1* is met.

The privacy of the bill *(R2)* in transit is preserved by the confidential channel along which it is transmitted *(A3)*.

2) Utility Provider Bill Verification: Firstly, the utility provider can be sure that it has received a commitment for every measurement which it is expecting because the commitments are signed alongside the timing information by the trusted tamper-resistant smart meter *(A1)*. The signature's integrity prevents the timing information from being forged (for example sending a valid commitment to a low reading for every time period). The signature also guarantees that the source of this pair was the smart meter.

Next the provider can use these homorphic commitments to calculate a commitment to a correctly calculated final bill. The provider knows that this commitment is to the correctly calculated bill because it performs the calculations itself *(A2)*. The provider may then calculate a commitment to the bill it was sent using the appropriate salt. The two commitments to the bill will be the same if and only if the two bills are the same because Pedersen Commitments are computationally binding [10] and because the Pedersen Commitment algorithm is deterministic for a given salt. Therefore *R3* is met.

IV. Implementation and Evaluation

This section discusses the existing proof of concept implementation of these protocols and evaluates its suitability for real world deployment.

A. Existing Implementation

Implementations of the smart meter billing protocols may be found at https://github.com/tblah/project-billing.

The Secure Channel protocol used to provide confidential and authenticated communication may be found at https://github.com/tblah/project-net.

Some cryptographic primitives used in these protocols are implemented at https://github.com/tblah/project-crypto but most used the widely reviewed implementations in Libsodium through the SodiumOxide language bindings [13]. The integer exponentiation implementation in GMP was used to implement the commitment scheme.

Standard practice in the development of software for embedded systems is to write in C. Contrary to this, the protocol implementations were written in Rust. Rust [14] is a modern systems programming language which uses compile-time checks to ensure that code is memory safe and so incurs no run-time overhead to prevent memory safety issues. This feature is very important for secure software because a large proportion of software security vulnerabilities, particularly in C programs, occur because of memory safety issues such as buffer overflow attacks [15]. Also, when Rust software is multithreaded, these same compile-time checks help to avoid data race conditions.

The protocols were implemented as software libraries so as to ease their integration into other projects.

B. Barriers To Integration In A Real Product

Many Smart Meters are designed with cheap microcontrollers [16]. These microcontrollers have limited data processing capability and cannot run general purpose operating systems such as GNU+Linux.

The first problem is the performance of the three party protocol. The computation of a Pedersen commitment requires two modular exponentiation operations and the generation of a large pseudorandom number. This may take too long on some microcontrollers to allow billing to reach the frequency specified by utility providers.

The second problem is that the proof of concept implementations are written for a hosted environment. Depending upon

the microcontroller used and software licensing requirements, adapting the proof of concept implementations to work on the unhosted smart meter systems could require substantial work.

Finally, the proof of concept implementations were written for demonstration purposes and so the implementation of the provider's server in the three party protocol lacks the tests to ensure that consumption measurements are not being replayed or removed, which were not included because forcing a particular number of commitments to particular hours would have made the demonstration impractical. These tests would not be difficult to add to the three party protocol implementation.

V. Conclusions and Future Work

A. Conclusion

As shown in Section III, both protocols meet the requirements in Section II-A. The simple signing protocol meets these requirements with acceptable performance when implemented using ed25519 signatures. The three party protocol does not achieve sufficient performance to be useful in a real world application but less performance sensitive applications exist for similar blind addition and multiplication proofs; for example electronic voting.

Similar work in [3] concluded more positively about the performance of Pedersen Commitments on smart meters. Firstly, Jawurek *et al* used smaller primes. It is the opinion of the author that this is not appropriate for smart meters because they are very costly to replace after they have been deployed and so the useful lifetime of their cryptography should be maximized. Secondly, performance measurements were made on an Intel(R) Core(TM) i5-M540 CPU: operating at 2.53GHz, which is a significantly more powerful processor than those included in modern smart meters.

B. Future Work

Specific to the problems addressed in this paper, more mathematical work would be useful to search for alternative commitment systems for the three party protocol to replace Pedersen Commitments with an alternative with better performance on microcontrollers.

More generally, billing is only one proposed usage for smart meters. Another reason for their deployment is the improvements to utility supply forecasting which their fine grained measurements could facilitate. Protocols (such as those outlined in this paper) which prevent clear text readings from leaving the home networks of customers will prevent this functionality: if utility providers only learn monthly consumption totals they will have no more ability to forecast demand than they do currently.

References

[1] Cisco. (Accessed 2016). The internet of things: It's not about things, it's about service, [Online]. Available: https://www.jasper.com/infographics/internet-things-its-not-about-things-its-about-service.

[2] M. Jawurek, F. Kerschbaum, and G. Danezis, "Privacy technologies for smart grids - a survey of options," Nov. 2012. [Online]. Available: https://www.microsoft.com/en-us/research/publication/privacy-technologies-for-smart-grids-a-survey-of-options/.

[3] M. Jawurek, M. Johns, and F. Kerschbaum, "Plug-in privacy for smart metering billing," in *Privacy Enhancing Technologies: 11th International Symposium, PETS 2011, Waterloo, ON, Canada, July 27-29, 2011. Proceedings*, S. Fischer-Hübner and N. Hopper, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 192–210, ISBN: 978-3-642-22263-4. DOI: 10.1007/978-3-642-22263-4_11. [Online]. Available: http://www.fkerschbaum.org/pets11.pdf.

[4] M. Enev, S. Gupta, T. Kohno, and S. N. Patel, "Televisions, video privacy, and powerline electromagnetic interference," in *In Proceedings of the 18th ACM conference on Computer and communications security, CCS '11*, ACM, 2011, pp. 537–550.

[5] "Privacy assessments," 2017. [Online]. Available: https://www.smartenergycodecompany.co.uk/sec/privacy-assessments.

[6] A. R. George Danezis Markulf Kohlweiss, "Differentially private billing with rebates," Feb. 2011. [Online]. Available: https://www.microsoft.com/en-us/research/publication/differentially-private-billing-with-rebates/.

[7] M. Backes and S. Meiser, "Differentially private smart metering with battery recharging," in *Data Privacy Management and Autonomous Spontaneous Security*, ser. Data Privacy Management and Autonomous Spontaneous Security. Springer Science + Business Media, 2014, pp. 194–212. DOI: 10.1007/978-3-642-54568-9_13. [Online]. Available: https://eprint.iacr.org/2012/183.pdf.

[8] G. Danezis, "Privacy-preserving smart metering," Nov. 2010. [Online]. Available: https://www.microsoft.com/en-us/research/publication/privacy-preserving-smart-metering/.

[9] K. Kursawe, G. Danezis, and M. Kohlweiss, "Privacy-friendly aggregation for the smart-grid," 2011. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/privacy%5C_in%5C_metering-mainfinal.pdf.

[10] N. Smart, *Cryptography : an introduction*. McGraw-Hill, 2003, pp. 363–367, ISBN: 0077099877. [Online]. Available: http://www.worldcat.org/oclc/248930429?referer=xid.

[11] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology - CRYPTO '91*, ser. Advances in Cryptology - CRYPTO '91. Springer Nature, 1991, pp. 129–140. DOI: 10.1007/3-540-46766-1_9. [Online]. Available: https://doi.org/10.1007/3-540-46766-1_9.

[12] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, and P. Zimmermann, "Imperfect forward secrecy: How diffie-hellman fails in practice,"

in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15, Denver, Colorado, USA: ACM, 2015, pp. 5–17, ISBN: 978-1-4503-3832-5. DOI: 10.1145/2810103.2813707. [Online]. Available: http://doi.acm.org/10.1145/2810103.2813707.

[13] dnaq. (Accessed 2016). Sodium oxide: Fast cryptographic library for rust (bindings to libsodium), [Online]. Available: https://github.com/dnaq/sodiumoxide.

[14] Rust. (Accessed 2017). The rust programming language, [Online]. Available: https://www.rust-lang.org/en-US/.

[15] M. Kerrisk, *The Linux programming interface*. No Starch, 2010, ISBN: 978-1-59327-220-3. [Online]. Available: http://www.worldcat.org/oclc/460060172?referer=xid.

[16] A. Molina-Markham, G. Danezis, K. Fu, P. Shenoy, and D. Irwin, "Designing privacy-preserving smart meters with low-cost microcontrollers," University of Massachusetts Amherst, Microsoft Research Cambridge, 2011.