Privacy-Preserving Energy-Reading for Smart Meter

Gianpiero Costantino(✉) and Fabio Martinelli

Istituto di Informatica e Telematica, CNR, Pisa, Italy
{gianpiero.costantino,fabio.martinelli}@iit.cnr.it

Abstract. Smart Meters belong to the Advanced Metering Infrastructure (AMI) and allow customers to monitor locally and remotely the current usage of energy. Providers query Smart Meters for billing purpose or to establish the amount of energy needed by houses. However, reading details sent from smart meters to the energy provider can be used to violate customers' privacy. In this paper, our contribution is twofold: first, we present an architecture to turn traditional energy meters into Smart Meters, and then we illustrate a privacy-preserving solution, which uses Secure Two-party Computation, to preserve customers' privacy during energy-readings. In particular, we deployed a Smart Meter built upon an existing energy meter available in Italy. Then, we collected and analysed an energy trace of two months, and we tag customers hourly/daily/monthly habits by observing their consumes. Finally, we provide the feasibility of our solution to protect customers' privacy.

Keywords: Smart meter · Privacy · Secure Two-party computation · Energy trace · Raspberry pi

1 Introduction

The current energy infrastructure provides electric meters that run basic operations, e.g. showing the energy used in an embedded display. Over last ten years, Italy has maintained the electric meter leadership with ENEL and its "Telegestore", which is able to communicate with the energy provider through narrowband Power Line Communication (PLC).

The interest in developing and deploying new meters, called Smart Meters, is one of the next years goals. In 2007, the US Congress decided to modernise its electricity transmission distribution network via the Energy Independence and Security. In 2009, the European Union passed a directive asking all Member States to conduct an economic assessment of smart metering[1]. In *Energy: Commission paves the way for massive roll-out of smart metering systems*[2] published in 2012, the European Commission asserts that only 10% of houses have some

[1] European Parliament and Council, 'Directive 2009/72/EC concerning common rules for the internal market in electricity and repealing Directive 2003/54/EC

[2] http://tiny.cc/sfyzvx

A. Geissbühler et al. (Eds.): ICOST 2015, LNCS 9102, pp. 165–177, 2015.
DOI: 10.1007/978-3-319-19312-0_14

sort of smart meter installed, and that 80% of all electricity meters in the EU will be replaced by smart meters by 2020. As negative fact, in November 2014, the UK's Department of Energy and Climate Change announced that the UK's smart metering deployment is delayed for other twelve months[3] without giving a clear reason for that choice.

From the above situations, the installation of Smart Meters into every house appears to be a long process. Thus, in this paper we propose an architecture to turn current energy meters into Smart Meters to bridge the gap between the Smart Meters roll-out and the existing energy meters. In particular, we developed a meter in which customers are able to know in real time their energy consumption both locally and remotely by using any device connected to the Internet, plus the opportunity to record the energy consumptions for analytics.

Then, we show a privacy study of our energy trace[4] that illustrates how to identify some human behaviours analysing the energy consumption. In fact, the adoption of "smart" devices brings new privacy issues that attackers may exploit for different purposes. For instance, attackers could observe the energy consumes of a house to learn the daily activities of the landlord in order to discover the right period to burgle his house.

Not only privacy issues hit Smart Meter, but also security problems can create damages to Smart Meters and their owners. As an example, Spanish researchers[5] proved how to send to the energy provider fake values of energy reads to get a "lighter" bill. In this attack, the researchers were able to discover the keys adopted by the smart meters and the provider nodes to spoof exchanged messages.

Finally, we propose the use of Secure-Two-party Computation (STC) to preserve customers' privacy. STC is part of the cryptographic field, and it can be used for privacy-preserving computation in which the goal of the two parties is to jointly compute the outcome of a generic function $g(x, y)$ without disclosing to the other party the own input. Our solution allows customers to keep protected their values of energy, but at the same time providers have indication on how much energy customers need.

The structure of this paper is the following: in Section 2 we provide our architecture to turn traditional energy meters into Smart Meters. In Section 3, we plot energy consumes and we illustrate how it is possible to rebuild customer hourly/daily/monthly pattern by observing his energy consumption. In Section 4, we provide our solution to preserve customers' privacy using STC. Section 5 shows some work related to ours. Finally, Section 6 concludes the paper.

2 Smart Meter Architecture

In this section we show the architecture that we use to turn an energy meter into a smart one. Figure 1 shows the building blocks of our architecture. We use the

[3] http://tiny.cc/7gyzvx

[4] We recorded two months of energy consumption with our Smart Meter.

[5] http://www.bbc.com/news/technology-29643276

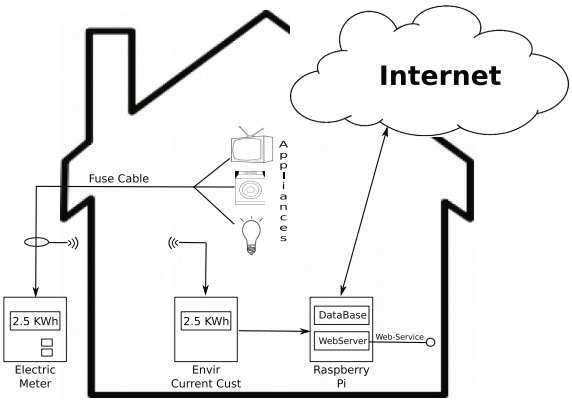

Fig. 1. Smart Meter Architecture

Envir Current Cost[6] to estimate the energy required by all capabilities attached at the house energy net, and the *Raspberry Pi*, which is programmed to store all energy consumes and to expose a web service able to communicate the current energy usage to remote entities connected to the Internet, such as Smartphones, Tablets and so on.

The Envir device is composed by two main components: the *transmitter* and the *receiver*. The transmitter is attached to the energy meter through a sensor jaw, instead the receiver is located into the house and displays the current energy usage plus other details, such as the indoor temperature, time and so on.

The Raspberry Pi is a tiny and low cost computer with 700 Mhz ARM processor, 512 Mbyte of RAM that runs a Unix Operating System. In our architecture the Raspberry Pi hosts a MySQL database plus an Apache TomCat WebServer, and the Raspberry Pi is connected to the Internet with an additional WiFi interface.

Figure 1 summarises our architecture. On the left side, the sensor jaw is hooked up the fuse cable of the energy meter of the building. The Envir receiver is located inside the building and shows the current energy use. The displayed values are updated every six seconds, and an outgoing serial cable connects the receiver to the Raspberry Pi.

2.1 Recording Energy Consumes

The data stream sent from the Envir device to the Raspberry Pi is evaluated using a Python script, see code listed in 1.1. The `port` variable expresses the mounting point of the serial port into the Raspberry Pi operating system, the `baud` rate specifies the number of symbols sent per second, while the `timeout` specifies the number of seconds to wait until the program stops with a connection

[6] Available here: http://www.currentcost.com/product-envir.html

error. We use the object "Serial" to read the stream from the cable, while the meter variable is a string containing all data read from the serial cable.

```
port = '/dev/ttyUSB0'
baud = 57600
timeout = 10
...
meter = serial.Serial(port, baud, timeout=timeout)
```

Listing 1.1. Python code to read from the serial port

From the data stream, we extrapolate two main values: i) last monitored energy consume, and ii) the current indoor temperature. These two values are saved into local variables and, then stored into a MySQL table. In the same SQL-insert tuple, we introduce two more variables, indicating time and date. For performance reasons, we decide to store data into the database roughly each minute although the data stream is read every six seconds.

2.2 Remote Reading

To make more powerful our Smart Meter, we installed a web-server into the Raspberry Pi. It aims at exposing one or more web services that can be remotely called to provide real time information. Web-services are called through a unique web address and the output is given as a "html" page.

The web-server we adopted is Apache Tomcat[7], and web-services implementation usually follows two approaches: *SOAP*[8] and *REST*[9]. The first one exposes a set of methods that can be invoked remotely by clients, while REST allows developers to define a set of resources that clients can request using the HTTP/HTTPs protocol.

In our implementation we prefer the REST technology since it is lightweight and fast, which are essential features for developing technologies on Smart Meters. For our purpose, we expose a single web-service in which users can visualise through a web browser last monitored energy consumption and internal temperature.

Figure 2 completes the architecture of our smart meter connected to the Internet. The traditional energy meter is linked to the home area network by means of the Envir device and the Raspberry Pi. This latter exploits the Internet connection to expose the web-server. At the same time, the web-service allows any authorised user to reach the Smart Meter information using her own device. Access control is managed using username and password authentication for users who want to access the web-service.

2.3 Pros and Cons

We acknowledge this architecture has some limitations. For instance, we admit that a customer has only the power to "read" data from his/her meter and not

[7] http://tomcat.apache.org
[8] http://www.w3.org/TR/soap/
[9] http://tiny.cc/esswhx

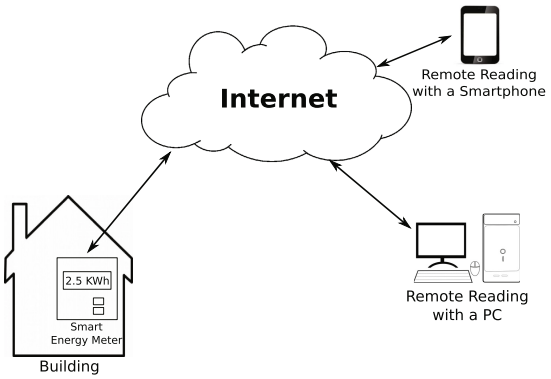

Fig. 2. Our Smart Meter connected to the Internet

to "write". In fact, a customer is not able to change any kind of data saved in the meter. However, this sort of limitation can be seen also as positive feature since a potential customer/attacker has not ability to corrupt smart meter data.

A strength of our architecture is its versatility. It is enough to plug the sensor jaw to the fuse cable of the meter to know the energy used, then the Raspberry Pi completes the architecture to turn a meter into a smart one. In this way, customers are able to manage their consumes, and can forecast the billing price as well as compare the energy used declared in the bill with that one estimated by the smart meter.

3 Privacy Study

The architecture introduced in the previous section found its application in a flat located in Italy. Currently, ENEL's customers have installed at home the *Telegestore*[10] meter. Advantage of the "Telegestore" is to be already connected with the Enel servers, through narrow-band Power Line Communication (PLC). This kind of communication avoids that Enel-employees pass house-by-house to collect the consume of energy of each meter. Nevertheless, customers at home do not have direct control of the meter as well as remote reading of the current energy consume. In fact, they can read the energy consumption through a display embedded into the meter. Generically, meters are located outside the building and in some cases they are not easily accessible.

For our experiments, we started collecting energy consumes at the end of April 2014, and we had collected data for two months, up to the end of June.

In Figure 3 we plot the consumes recorder in a single hour. Values collected are 54 since the measurements are saved into the database roughly every minute. Observing all consumes it is possible to extrapolate a particular detail. We found

[10] http://www.enel.com/en-GB/innovation/smart_grids/smart_metering/telegestore/

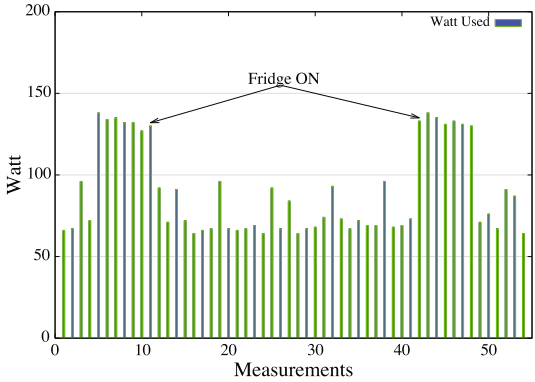

Fig. 3. One hour flat consumes

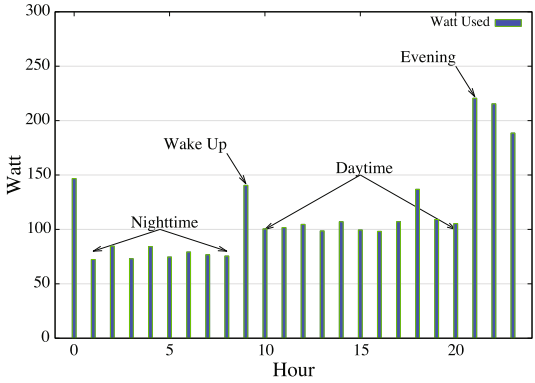

Fig. 4. 24 hours flat consumes

an energy peak every 30 minutes related to an appliance that we link to the fridge. In fact, it remains "on" for about seven minutes and then switches to "off". After half an hour it goes "on" again.

In Figure 4, we plot the energy consumes per hour in a single day. Values reported for each hour are obtained averaging all consumes. Results of this figure are very interesting since we identify different periods of the day. Starting from midnight, in which it is recorded a high consume of energy, then we observe a low consume of energy during the nighttime. Afterwards, around "09:00" in the morning a new peak of energy identifies the moment in which there was activity in the flat. During the daytime the consume of energy is low and pretty uniform saying that nobody was in the flat except an usage activity monitored around "18:00". Finally, after "20:00" new peaks of energy are registered representing again more request of energy.

In Figure 5 we show the energy consumes monitored during May 2014. Consumes of each days are plotted as the mean of the entire day consumes. During this month we identify three different intervals in which consumes are low. Those periods correspond to a holiday time, from first to fourth of May, and

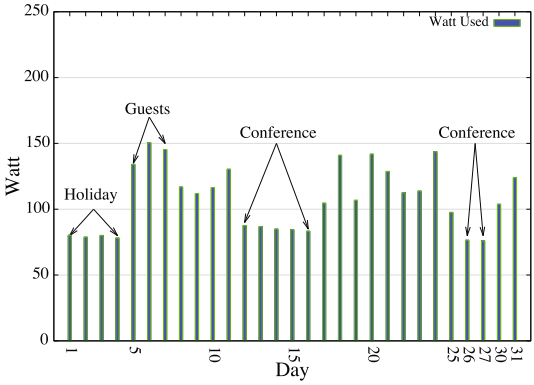

Fig. 5. Flat consumes in May

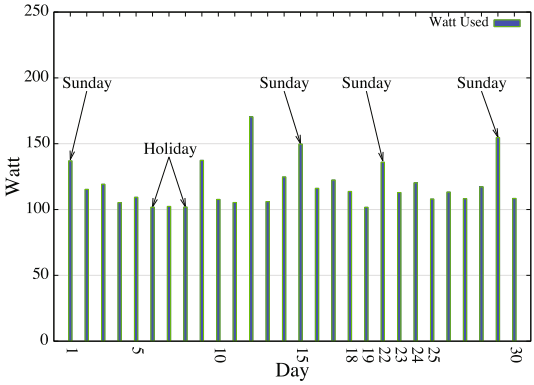

Fig. 6. Flat consumes in June

two conferences, from twelfth to sixteenth of May and at the end of the same month. Then, for three days, from fifth to eighth of May, we observe the highest consume of energy explained by the presence of additional people in the flat. In the same figure we do not have measurements for two days at the end of the month since our Smart Meter lost the connection with the database did not record any energy value.

In Figure 6 we plot the average-consumes per day in June. At first glimpse, we notice weekly high peaks of energy that correspond to those days in which the activity at home is higher than other days, e.g. "Sunday". Then, we tag another interval of days as "Holiday" in which we notice a low consume of energy that corresponds to a period in which nobody was at home. An empty space is present in the data trace during the 20th and 21st of June since we experienced again a network issue.

Energy Trace. Our values of energy may be useful for the community of researchers working in this field. For this reason, we share our consume trace

giving the chance to others researchers to make their sturdies on real energy consumes.

In the file we share, there are energy consumes of the flat that we used as test-bed in May-June 2014 period. Each row in the file is composed by three fields separated by ";" that represent: *"Data"*, *"Time"* and *"Watt"*. Our trace file can be imported into a database or can be directly analysed using the text file. It is available here[11].

4 Privacy Solution

In this section, we propose a privacy solution apt to protect customers' real-time readings done by providers. In particular, our solution makes use of Secure Two-party Computation to indicate in a range the effective customers' energy consumption without revealing the exact smart meter reading to the provider.

4.1 Using STC Inside Smart Meters

We believe that the cryptographic field may help proposing methods that can reduce the privacy issue seen above. In Secure Two-party Computation two parties are involved, generically called Alice and Bob, each holding some private data x and y, respectively. The goal of STC functions computation is to allow Alice and Bob to jointly compute the outcome of a function $f(x, y)$, without disclosing to the other party their own input. The straightforward way to solve the above problem would be to have a Trusted Third Party (TTP) to which Alice and Bob securely send the data, and to have the TTP compute $f(x, y)$ and separately send the outcome to Alice and Bob. The business in secure two-party computation amounts to securely compute $f(x, y)$ without the need of a TTP.

The first implementation of Secure-Two party Computation was made by Yao in the 1980s [1], presenting a solution for the well-known "Millionaire Problem". In the Millionaire Problem, the two parties want to know whom of them is richer without revealing to the other party his/her own amount of money. The problem requires the evaluation of $x < y$ condition, where Alice knows only *"x"*, and Bob knows only *"y"*. At the end of the protocol execution, Alice knows only the outcome of the evaluation of condition $x < y$, without knowing y (similarly for Bob).

Over the last ten years, researchers have proposed different Secure Two-party Computation frameworks able to run secure functions. FairPlay [13] is a well-know framework that allows users to write functions using its high level language (SFDL), and to compile functions into garbled boolean circuits, which will mask the real inputs of both participants. FairPlay has strong security properties in the context of two-party computation. The framework is shown to be secure against a malicious party; in particular i) a malicious party cannot learn more

[11] http://tiny.cc/5v41vx

information about the other party's input than it can learn from a TTP that computes the function; and *ii*) a malicious party cannot change the output of the computed function. New versions of this framework are FairplayMP [2], which is the extension of Fairplay that works with more than two parties, and Mobile-Fairplay [5],[6], which is the version of Fairplay ported to Android Smartphones.

A more recent STC framework is MightBeEvil [11]. It allows people to easily write functions that can be run in a secure way, in a similar way done by Fairplay, however, MightBeEvil is faster and less memory-hungry than Fairplay.

The STC framework that we use in our smart meter is CBMC-GC [10]. It is composed by two main parts: one is the compiler that translates functions written in C into garbled circuits, while the other part is the interpreter able to execute compiled functions [12]. Compared with Fairplay, CBMC-GC offers a more flexible high language that allows developers to express more detailed and complicated functions. Moreover, thanks to its optimisation steps during the compilations phase, CBMC-GC runs STC function using less memory than framework like Fairplay.

STC attacker model. The aforementioned STC frameworks are secure against the *honest-but-curious* attacker model. In this model, an attacker follows all protocol steps as per specifications, but she can try to learn additional information about the other party, with the purpose of acquiring at least part of her private profile. Moreover, notice that, as customary in secure two-party computation, there is an asymmetry on the provided security guarantees: in particular, there is no way to prevent Alice from terminating the protocol prematurely, and not sending the outcome of the computation to Bob. This situation can be detected by Bob, but cannot be recovered from.

4.2 Our Solution

Here, we propose the use of STC to preserve customers' privacy when providers remotely read the energy value from the smart meter. We ported the CBMC-GC framework to our smart meter and we allow both provider and customer to run STC functions. To preserve customer's privacy, we wrote a function in which the customer's smart meter provides its current usage of energy as input, and the provider uses three intervals of energy as indication of energy usage. The output of our function indicates to the provider in which range the smart meter's input falls on, but the provider will never know its exact value. For instance, let suppose that the provider uses the following ranges in Watt (W): $[0, 1000]W$, $[1001, 2000]W$, $[2001, 3000]W$, and last meter reading is 547W. After running the STC function the provider knows that last monitored consume ranges in the interval $[0, 1000]W$, but it does not know the exact value, although it had been used to evaluate the energy consumption.

The listing in 1.2 shows the STC function that we wrote in C. The range function calculates the exact range of usage and it uses four variables: x is the reading done by the meter, y, t, and z are respectively the first, second, and third interval limit. In addition, each range is mapped as single progressive number,

i.e. $[0, 1000]W \rightarrow 1$, $[1001, 2000]W \rightarrow 2$, $[2001, 3000]W \rightarrow 3$. To obtain a more fine-grained output, our program may consider more than three intervals.

```
int range(int x, int y, int z, int t)
{
   int output = 0;
   if ((x >= 0) && (x <= y))
     output = 1;
   else if ((x > y) && (x <= z))
     output = 2;
   else if ((x > z) && (x <= t))
     output = 3;
   return output;
}

void meterCheck(int INPUT_A_x, int INPUT_B_int1, int INPUT_B_int2, int
     INPUT_B_int3)
{
  int OUTPUT_meterCheck = range(INPUT_A_x, INPUT_B_int1, INPUT_B_int2,
       INPUT_B_int3);
}
```

Listing 1.2. STC function written in C

Implementation. The C-function listed in 1.2 is translated into garbled circuits by the CBMC-GC compiler, and the garbled circuits are distributed to the provider's computer and to customer's smart meter. The CBMC-GC running environment is also installed both on the smart meter and the provider's pc as it is depicted in Figure 7. In particular, the CBMC-GC instance installed in the meter waits for incoming queries from the provider, and once the connection is established they start the secure computation. Only at the end of the computation the provider is able to know in which range the smart meter reading falls on.

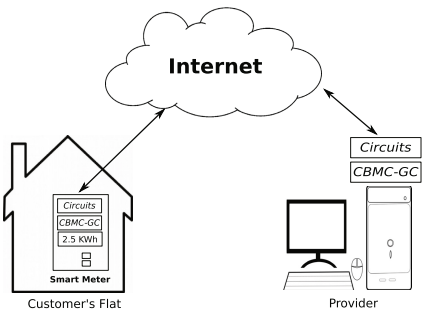

Fig. 7. Customer's smart meter and provider's PC with CBMC-GC

Empirical test conducted in our lab showed that using a Virtual Machine (VM) with Ubuntu OS and 16Gbyte of RAM to represent provider's PC, and the Raspberry Pi to represent smart meter, the time needed to run the function with CBMC-GC framework is about *40* seconds. The high computational time is due

to the complexity of mathematical operations required by the STC framework and, particularly by the low computation power of our Raspberry Pi. We believe that in this context 40 seconds are a feasible time, and to support our belief, in this document[12], Enel claims that its Smart Meter saves every two minutes the KiloWatt taken. So, in practice Enel would have the time to run the STC function without losing any data precision.

5 Related Work

Privacy issues related to smart meters' reading have been studied in papers [8], [16], [17] and previously acknowledged in [4], [9]. The authors of [3] show how to apply homomorphic schemes for aggregation in smart metering data to protect customers' privacy, and in particular they discuss the applicability of the method presented by Mármol et al. [14]. A similar data aggregation is performed by Sankar et al. [19] in which data collected by multiple meters are aggregated before sending them to the utility. Rial and Danezis in [18] propose a protocol for privacy-preserving meter readings, and at the same time prove the reading correctness of the meter. Molina et al. in [15] shows how to apply Zero-Knowledge Proofs to low-cost microcontrollers to produce certified meter readings. In [7] the authors propose a solution to preserve customers' privacy providing encrypted meter reading with the adoption of secret-sharing-based secure multi-party computation techniques. Thoma and Cui in [20] make use of secure multi-party computation to know customers energy demand keeping hided the amount of energy needed by each consumer.

In the market there exist commercial tools to monitor energy consumption using indoor display or web-browser. TED pro-home[13] is a commercial for residential electricity monitoring and its price starts from 300$. The Egauge System company[14] sells its powerful product "EG3000" for energy monitoring starting from 500$. Openenergymonitor[15] is a project to develop open-source energy monitoring tools. This project proposes different hardware solutions and sensors that can be built together to create a monitoring system accessible by a web-application.

None of these contributions is totally related to ours, although we are not the first that apply cryptographic solutions to protect customers' privacy. To the best of our knowledge, our research paper is the only one that integrates an architecture to turn current meters into smart ones, with a cost of around 100€, and a mechanism to protect customers' privacy all in a single feasible solution.

[12] There exists only the italian version: https://www.enel.it/it-IT/doc/reti/enel_distribuzione/Contatore_Monofase_v1.pdf

[13] http://www.theenergydetective.com/tedprohome.html

[14] https://www.egauge.net

[15] http://openenergymonitor.org/emon/

6 Conclusion

In this paper we have presented an architecture to turn traditional energy meters into new Smart Meters with a solution to preserve customers' privacy. In particular, our prototype of Smart Meter allowed us to record energy consumes for two months in a fine-grained fashion, i.e. roughly each minute. With the collected energy values, we illustrated how it is possible to identify human patterns, and for this issue, we implemented a solution that makes use of Secure-Two-party Computation to preserve customers' privacy during the real time monitoring. Finally, we showed the feasibility of our intuition by porting the CBMC-GC framework inside our smart meter prototype.

As further step, our intention is to use the new Raspberry Pi 2 —quad-core of 900Mhz and 1GByte of RAM— within our architecture. Benchmark tests claim that the Pi 2 is 6x faster that the previous model. This means that our STC computation will get a substantial speed up respect to the current implementation.

Acknowledgments. Work partially supported by project *Security Horizon* and by the EU project FP7-295354 *SESAMO*.

References

1. Andrew, C., Yao, C.: Protocols for secure computations. In: 23rd IEEE Symposium on FOCS, pp. 160–164 (1982)
2. Ben-David, A., Nisan, N., Pinkas,B.: Fairplaymp: a system for secure multi-party computation. In: Proceedings of the CCS Conference, pp. 257–266. ACM, New York, NY (2008)
3. Biselli, A., Franz, E., Coutinho, M.P.: Protection of consumer data in the smart grid compliant with the german smart metering guideline. In: Proceedings of the First ACM Workshop on Smart Energy Grid Security, SEGS 2013, pp. 41–52. ACM, New York, NY (2013)
4. C. P. U. Commission. The future of privacy forum and truste launch a smart grid privacy seal program, May 2011. http://tiny.cc/jlpwhx
5. Costantino, G., Martinelli, F., Santi, P.: Investigating the Privacy vs. Forwarding Accuracy Tradeoff in Opportunistic Interest-Casting. Transactions on mobile computing (2013)
6. Costantino, G., Martinelli, F., Santi, P., Amoruso, D.: An implementation of secure two-party computation for smartphones with application to privacy-preserving interest-cast. In: Proceedings of the 18th International Conference Mobicom, pp. 447–450. ACM (2012)
7. Danezis, G., Fournet, C., Kohlweiss, M., Béguelin, S.Z.: Smart meter aggregation via secret-sharing. In SEGS@CCS, pp. 75–80 (2013)
8. Greveler, U., Justus, B., Loehr, D.: Forensic content detection through power consumption. In: IEEE ICC, pp. 6759–6763, June
9. Heck, W.: Smart energy meter will not be compulsory, April 2009. http://tiny.cc/9vpwhx

10. Holzer, A., Franz, M., Katzenbeisser, S., Veith, H.: Secure two-party computations in ansi c. In: Proceedings of the CCS Conference, CCS 2012, pp. 772–783, NY, USA (2012)
11. Huang, Y., Chapman, P., Evans, D.: Privacy-preserving applications on smartphones. In: Proceedings of the 6th USENIX Conference on Hot Topics in Security, HotSec 2011, pp. 4–4. USENIX Association, Berkeley, CA (2011)
12. Kolesnikov, V., Schneider, T.: Improved garbled circuit: free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008)
13. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay—a secure two-party computation system. In: Proceedings of the 13th Conference on USENIX Security Symposium - vol. 13, SSYM 2004, pp. 20–20. USENIX Association, Berkeley, CA (2004)
14. Mármol, F., Sorge, C., Ugus, O., Pérez, G.: Do not snoop my habits: preserving privacy in the smart grid. IEEE Communications Magazine **50**(5), 166–172 (2012)
15. 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)
16. Molina-Markham, A., Shenoy, P., Fu, K., Cecchet, E., Irwin, D.: Private memoirs of a smart meter. In: Proceedings of the SenSys Workshop (Builsys), BuildSys 2010, pp. 61–66. ACM, New York, NY (2010)
17. Quinn, E.L.: Privacy and the new energy infrastructure, p. 43, February 2009
18. 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, NY (2011)
19. Sankar, L., Kar, S., Tandon, R., Poor, H.V.: Competitive privacy in the smart grid: an information-theoretic approach. CoRR, abs/1108.2237 (2011)
20. Thoma, C., Cui, T., Franchetti, F.: Privacy preserving smart metering system based retail level electricity market. In: Power and Energy Society General Meeting (PES), 2013 IEEE, pp. 1–5, July 2013