Some Ideas on Privacy Preserving Meter Aggregation

Klaus Kursawe, kursawe@cs.ru.nl

November 1, 2010

1 Abstract

With the upcoming widespread distribution of smart meters for electricity, gas and water consumption, new opportunities are closely linked to new privacy concerns. In this paper, we present protocols that can be used to privately compute aggregate measurements, allowing for fraud- and leakage detection as well as statistical evaluation of meter measurements, without revealing any information about the non-aggregated values.

While the protocols are aimed at the smart metering case, the technologies provided may be of independent interest for aggregating protocols.

2 Introduction

The area of smart metering for electricity, but also other commodities such as gas and water is currently experiencing a huge push; for example, the European commission has formulated the goal to provide 80 % of all households with smart electricity meters by the year 2020 [5]. Simultaneously, privacy issues are mounting – in 2009, the Dutch Senate stopped a law aimed to make the usage of smart meters compulsory based on privacy and human rights issues[3]. While it is not clear yet how much data can be derived from actual meter readings, the high frequency suggested (i.e., about 15 minute reading intervals), together with the difficulty to temporarily hide one's behavior (as one can do, for example, with mobile phones), gives rise to massive privacy concerns.

An important aspect in privacy preserving metering protocols is to take into account the rather limited resources on such meters, both in terms of computation and in terms of communication. We therefore push as much workload as possible to the back-end, leaving the minimal work possible on the meter itself. In terms of communication, one has to keep in mind that meters can be connected with a low-bandwidth medium, and thus the messages sent out by the meters should increase only minimally. The communication architecture also implies that meters should ideally act independently, without requiring interaction with other meters wherever

possible and minimal interaction when not. In addition, meters may be restricted in the amount of available memory, which limits the state information on other meters that can be stored.

If one considers gas- and water meters, this issues are even bigger; those meters are either powered by batteries or need to harvest the energy from their environment, implying severe constraints on both computation and communication.

We show two protocols for load aggregation and fraud detection in the smart grid setting. The first protocol allows an aggregator to compute the sum of a number of input values, without learning the value of the inputs themselves. The trust assumption in this protocol is scalable, from trusting any other entity that provides input (i/.e., other meters) down to trusting that only one of the remains honest. The communication complexity of this protocol is quadratic in the number of distrusted meters, i.e., $O(n^2)$ bits in the most conservative setting, where n is the number of meters in the system. The protocol is information theoretically secure, assuming secure links between meters.

The second protocol allows an aggregator to compare the sum of the input values to a value she already knows (e.g., by her own measurement), and determine if it is (roughly) the same. This protocol only requires every meter to send one message to the aggregator (which is optimal), and assures that the aggregator - as well as the corrupt meters - learn nothing they cannot derive from the aggregated values. The protocol is secure under some cryptographic assumptions, namely in the Random Oracle model with Decisional Diffie-Hellman. We also propose an extension that turns the comparison protocol into an extraction one, i.e., allows to compute the aggregate rather than only comparing it to a known one.

While fraud detection is becoming a major issue - for example, an recent FBI report states that spot checks in one American state have shown 10% of all meters to have been tampered with - the protocols can also be used to detect leakages in other measurements, e.g., water and gas. In this case a good privacy preserving solution is even more relevant, as such measurements should be done in a high frequency to detect potentially dangerous leaks as soon as possible.

2.1 Related Work

The concept of privacy preserving metering aggregation comparison has first been introduced by Garcia and Jacobs[1]. However, their protocol requires $O(n^2)$ bytes of interaction between the individual meters as well as relatively expensive cryptog-

raphy on the meters itself. Fu et all [2] propose an architecture for secure measurements, but rely on trusted components outside of the meter. Rial and Danezis[4] propose a protocol using homomorphic commitments to allow an individual meter to prove the correctness of its bill without revealing the individual measurements.

2.2 Our Contribution

We show two protocols for load aggregation and fraud detection in the smart grid setting. The first protocol allows an aggregator to compute the sum of a number of input values, without learning the value of the inputs themselves. The trust assumption in this protocol is scalable, from trusting any other entity that provides input (i/.e., other meters) down to trusting that only one of the remains honest. The communication complexity of this protocol is quadratic in the number of distrusted meters, i.e., $O(n^2)$ bits in the most conservative setting, where n is the number of meters in the system. The protocol is information theoretically secure, assuming secure links between meters.

The second protocol allows an aggregator to compare the sum of the input values to a value she already knows (e.g., by her own measurement), and determine if it is (roughly) the same. This protocol only requires every meter to send one message to the aggregator (which is optimal), and assures that the aggregator - as well as the corrupt meters - learn nothing they cannot derive from the aggregated values. The protocol also allows to define different aggregation sets, e.g., compute the aggregate of all meters connected to a particular substation and the aggregate of all meters connected to home with photo-voltaic energy generation. This flexibility does not cause extra effort on the meter itself, but only on the distribution of the keys. The protocol is secure under standard cryptographic assumptions, namely in the Random Oracle model with Decisional Diffie-Hellman.

The protocols are designed in a way that multiple aggregation sets can be used simultaneously. Thus, without any additional effort on the meter side, it is possible to simultaneously aggregate for example over all meters connected to a particular substation, and over all meters connected to a household with photo-voltaic energy generation Finally, we describe how to modify the second protocol to also allow to decrypt the aggregates, i.e., to compute the aggregate only from the encrypted inputs provided by the meters.

3 Basic Attack Model

The *Aggregator* may be interested in detailed meter data that she is not allowed to get. We do assume that the aggregator can control some meters, in which case it has all their keys and completely controls their behavior. Note that in a practical setting, we can assume that the aggregator will not behave completely dishonest, but more what can be described as "flawed but non-criminal"; that is, data that is or can easily me made available will be abused, but the aggregator will not commit easy to detect criminal acts (e.g., invent hundreds of non-existing meters in the setup phase, or put a second communication device into the meter) to be able to spy on an individual meter; this will make the real-world key- and device management much easier.

A smart meter has essentially two components, that may be separated by software or even be implemented on separate hardware blocks. The metrology part is the meter component responsible for the actual measurement. This component is usually certified and must not be modified, e.g., it is not allowed that this part is software-updatable. The second part deals with all other aspects of the meter, e.g., handling the communication between the meter and the outside world.

In our model, an attacker that wants to send falsified data has two options:

- A naive attacker will tamper with the meter by shortcutting it altogether, altering its output messages, or running simple attacks on the hardware. In those cases, we assume that our protocols run inside the trusted domain and cannot be altered. We assume that such an attacker can arbitrarily modify the input (i.e., measurement) and output (i.e., messages sent) of the meter, but cannot get internal key material or alter the protocol run. We also assume that the meter has an authenticated communication channel with the aggregator, though there are ways to integrate authentication into the protocol itself.

- An advanced attacker may completely hack the meter and replace the entire firmware. While we do sketch ideas to handle such attackers in the future work section, our protocols as presented here provide little protection if the meters become completely unreliable.

Assuming meters follow the protocol, side, the aggregator only needs to assume that the keys are properly distributed is proper to guarantee fraud prevention – the only fraud still possible is if two meters collude in a way that one meter overreports

be the same amount another one underreports [1]. We do assume some security in the meter that assure that the values reported to the fraud detection are the same reported to the billing system, and that messages from the meter are authenticated (alternatively, if H is a keyed hash function, it is sufficient for the meters to keep the corresponding key private). In this, we assume that attacks on the meter from the customer are usually done by circumventing the meter, rather than reprogramming the entire unit. This is a necessary assumption for any fraud detection, as we need to assure that the values the detection system gets are in some way related to reality; in the future work section, we direct towards a solution that would also allow completely hacked meters to be included.

While it is easy for an individual meter to cause false alarms – and in this, run some form of denial of service attack – this is not an issue for our protocol. As the whole point is to trigger an alarm if something goes wrong, and a certified meter launching a denial of service attack would certainly qualify as such, the protocol will act exactly as desired.

Extra care has to be taken as the measurement values of the meters may come from a very restricted domain, and thus can easily be predicted in a realistic setting.

4 Information theoretically secure Aggregation Computation Protocol

The basic idea is to give each meter a masking parameter that it can add to the actual meter reading (where the addition is performed in some finite group), where the masking parameters of all meters that are to be aggregated sums up to a value known to the Aggregator. Thus, the aggregator will not learn any information about the individual readings, but can add them up and subtract its known key value to derive the total sum.

To generate the masking parameter, we essentially use simple additive secret sharing approach. For each measurement, a subset of the meters are (deterministically) chosen as the *leaders*. For the sake of illustration, we assume for now that there is one leader selected. All parties compute completely random secret shares, and (in encrypted form) send them to the leader. The leader then computes it's final share in a way that all shares together share the public value known to the

[1] There are scenarios, especially with variable tariffs where that actually may make sense, but we safely can assume this to not be an issue for now

Aggregator (or alternatively, sends the sum to the aggregator). If several leaders are chosen, the same principle is applied to all leaders, and the resulting shares are added up.

These shares are now used to mask the meter reading; on the aggregator, they eliminate each other, so the resulting aggregated reading can be computed.

4.1 System Setup

We assume a system of n meters and one aggregator (substation) A. We call p the privacy parameter; this is the number of leaders in each round.

At system setup, each meter receives a private encryption key k_i, as well as the public encryption keys PK_j for all other meters in the same aggregation set.

To initialize a meter reading operation, each meter i computes p random values $s_{i,1}, ..., s_{i,p}$. It then computes the identities $l_1, ..., l_p$ of the p leaders, and encrypts $s_{i,j}$ with PK_{l_j}. These set of p keys is sent to A. A also computes the identities of the leaders, and sends each leader its corresponding share.

Each leader collects $n-1$ shares of its corresponding $s_{i,j}$, and computes its own share such that all shares together share the value 0.

Finally, all parties add all their shares $s_{i,1}, ..., s_{i,k}$ to get the main share s_i.

To send a reading c_i, a meter computes $c_i + s_i$. A collects all this data, and computes $\sum_i c_i + s_i = \sum_i c_i$.

To generate a new masking value for the next reading, the initialization step is repeated with a different set of leaders; the result is then added to the current shares.

4.2 Trust Model

While the protocol guarantees total privacy towards the aggregator, there are some trust assumptions on the meter. For one, privacy is only guaranteed if at least one of the chosen leaders is honest. This can be guaranteed by making every participant a leader (at the expense of performance), but in a practical setting one would try to keep the number of leaders low to avoid performance issues. Furthermore, every leader can easily send corrupted data and thus provide false data to the aggregator.

4.3 Performance

On a normal meter side, the effort is one addition for the reading, a simple computation of the leaders, determining p random values, p public key encryptions and the corresponding communication overhead. For the leaders, an additional $n-1$ decryptions and one polynomial interpolation have to be done.

4.4 Choice of p

The privacy of the scheme is broken if A collaborates with all leaders in a particular round, as this allows her to compute the masking values. However, it should be noted that even this leads only to a limited privacy breach - as the old masking values are still unknown, A can essentially only derive the difference between two consecutive readings, and looses all information in the next round (provided at least one leader there is honest). Thus, in practice, it is tolerable if occasionally all leaders are corrupted (as corrupting a leader would also require a criminal act on the side of the meter manufacturer, and should by design not be possible without physical access to the meter). We therefore figure that a value of 5 leaders is a good compromise between efficiency of the scheme and privacy protection, though those values can still be debated.

4.5 Some remarks on management

In a realistic setting, it will happen from time to time that a meter is withdrawn or added to the scheme. This can be done relatively easily, all that is required is that the leaders add/leave out the corresponding share. On a more practical view, there is some issue on authorization to withdraw or insert meters - a corrupt A could withdraw all meters from a given meters aggregation set and add 100 bogus meters - a setting in which every aggregation scheme is doomed to fail. For the purpose of this work, we simply assume that this is not going to happen, and leave the exact protection (be it legally or technically) to future work. If a meter drops out unexpectedly, the recovery gets a little bit more complicated - essentially, A will have to ask the meters redo the computation with one meter less. This does pose a privacy threat for the dropped meter if it can be done continuously, which is not too unlikely if a meter is connected with an unreliable connection. A potential solution is to allow every meter to be temporary dropped for two readings, after which it gets dropped for the rest of the day (the meter in question itself can be blissfully

ignorant of that, as it keeps protecting its readings with random pads, thus not revealing anything. Similarly, A will need to be able to kill an inactive leader (who would stall the entire protocol); this too should not be allowed too often before an alarm is raised.

Another potential issue is that the scheme only works if all meters in one aggregation set are reasonably synchronized, i.e., no meter 'misses' one leader phase and then comes up with different leaders than the other ones. We would argue though that this can easily be fixed, for example by a simple counter added to messages that would allow to detect and correct such a state.

4.6 Practical Considerations

In practice, it may not be feasible that a meter has a shared symmetric key with all other meters in its peer group – such a set of keys would require too much memory, even if optimizations developed for sensor networks are applied. It is possible to use a public-key infrastructure, and thus allow meters to securely exchange compute a new symmetric key whenever they want to communicate with another meter. This is somewhat feasible, as the reporting in a metering setting occurs in large intervals (e.g., 15 minutes), and thus even a ow powered device has plenty of time to compute and evaluate the necessary public key operations. The biggest value in this protocol, however, is in combination with the protocol described in the next section. Once a shared sum is provided (i.e., every meter has a secret value and the aggregator knows the sum), that protocol will allow for an arbitrary number of private measurements.

5 The Aggregation Comparison Protocol

Above solution works in two settings, both for an architecture where the meter-readings are used to obtain usage data for load management, and for a setting where the meter data is only compared to other measurements to detect energy theft. In this section we present a protocol that works for fraud detection only, and in this increases the overall efficiency of the protocol; communication is reduced to one message from each meter to the aggregator per measurement, and the effort on the meters is one exponentiation (or elliptic curve multiplication) per reading. The workload on the aggregator is somewhat higher, depending on the required precision of the measurements. The privacy of each user is protected as long as at least one other meter is honest.

The property we use in this scheme is that the aggregator already knows the (approximate) sum of of the values she is aggregating, and only needs to compare whether her sum is equal to the aggregate. This makes the task of privacy preservtion significantly easier, as the aggregation protocol only needs to output a (one-way) function of the added values, without the need of the aggregator learning the input value of that function.

5.1 The Scheme

Let G be a suitable Diffie-Hellman group, and $H(\{0,1\}^*) \to G$ a hash function mapping arbitrary strings onto elements of G. For the security analysis, we assume that H has random oracle properties. In practice, a derivate of normal hash-function such as SHA-256 will be fully sufficient. Let x_j be a preshared secret of meter m_j such that $\sum_j x_j = X$. We assume that each reading has a unique identifier R that is shared by all meters and A, e.g., a serial number or the time and date of the reading.

Note that while we describe the scheme in the discrete logarithm setting, the scheme does work as well on elliptic curves, which will reduce both computation and communication efforts in a practical implementation.

For each reading $c_{i,j}$ with identifier R, the meter m_j computes a common group element $\tilde{g} = H(R)$. It then computes $g_{i,j} = \tilde{g}^{c_{i,j}+x_i}$. The value $g_{i,j}$ is then send to the aggregator A.

The aggregator A collects all values of $g_{i,j}$, and then computes $g_a = \prod_j g_{i,j}$. By construction, we have $\prod_i g_{i,j} = \prod_i \tilde{g}^{c_{i,j}} * \prod_i \tilde{g}^{x_i} = \tilde{g}^{\sum_i c_{i,j}} * \tilde{g}^X$. As A has its own measurement c_a of the total consumption of the connected meters, it now needs to verify if g_a roughly equals $g^{c_a X}$. This can be done testing all values of $g^{c_a}, g^{c_a-1}, g^{c_a+1}, ...$ until either a match is found or a sufficiently large interval has been tested to justify a more extensive investigation.

5.2 Security Proof

We will demonstrate protocol security under Decisional Diffie Hellman in the Random Oracle Model.

5.2.1 Proof Outline

For the proof of security, we define an idealized protocol which obviously protects the meters' privacy, and proof that from the point of view of the aggregator, a run of the idealized protocol is indistinguishable from a run of the real protocol. Thus, the aggregator cannot learn any information in the real protocol that it did not learn in the idealized one.

To prove indistinguishability, we construct a run in which some meters follow the ideal protocol, while others follow the real one. One meter starts for some measurements with the ideal one, and then switches over to the real protocol. By embedding an instance of the Decision Diffie Hellman problem into that meter, we show that the aggregator cannot tell where exactly the switchover from ideal to real happened.

This allows us to use a diagonalisation argument to argue that if an attacker cannot tell where the switch from ideal world to real world happens, she also cannot distinguish a fully ideal world from a fully real one.

5.2.2 Attack Model

Assuming the blinding-keys are generated and distributed securely, the end-user does not need to trust either the meter or the aggregator at all (in terms of privacy protection). The protocol itself is completely deterministic with no secrets that an end-user would not be allowed to know, so no information can be hidden inside the messages. It is not even necessary that the meter does the calculation itself in the first place - given the meter reading, an external device (e.g., an internet connected PC) could perform this task as well. In this protocol, the adversary may control an arbitrary number of malicious meters; as the protocol is completely non-interactive and meters do not need to know of each others' existence those corrupted meters have no effect on the security and execution of the protocol.

5.2.3 Notations

We denote with n the number of honest meters. We assume $n \geq 2$, which is the minimum required for any aggregation. In addition to the n honest meters, we allow for an unlimited number of dishonest meters. As there is no communication between meters, the dishonest meters play no real role in the protocol or the proof.

We call m the number of measurements. There is no limit on m, apart from m being polynomial in the security parameter.

Let G be an appropriate group for Diffie Hellman; the following variables are elements in G:

$x_{i,j}$ = blinding value for measurement i on meter j

$c_{i,j}$ = measurement value for measurement i on meter j.

X deblinding value for the aggregator. X is the sum of blinding values $x_{i,j}$ for a fixed i.

In addition, we have a hash function $H : (\{0,1\}^* \to G$. We assume H to have random oracle properties. For readability, we define $\tilde{g}_i = H(i)$. Note that the domain for the $c_{i,j}$ can be small and predictable, i.e., an attacker can brute–force $c_{i,j}$ given g and $g^{c_{i,j}}$.

For the simulation, we have a given instance of the Decision Diffie Hellman problem, i.e., we have given $g, h_1 = g^a, h_2 = g^b, h_3 \in G$ and need to decide if $h_3 = g^{ab}$.

5.3 The Ideal and the Real world

We first define an idealized protocol, in which privacy is assured in an information theoretical sense. In this idealized world, every measurement i at meter j has a unique, independent blinding value $x_{i,j}$ such that for all i, $\sum_j x_{i,j} = X$.

For measurement i, meter j sends $m_{i,j} = x_{i,j} + c_{i,j}$ to the aggregator.

This is information theoretically secure (For everything we send, there are blinding values for all possible measurements that could have led there). We may need to be a little careful with the distribution, as the $c_{i,j}$ are poorly distributed.

If we now choose a (public) generator $\tilde{g}_i$ of an appropriate group G, sending instead

$$\tilde{g}_i^{x_{i,j}+c_{i,j}}$$

is at least as secure as sending $m_{i,j}$ directly. This is our ideal scheme.

Recall that $H : \{0,1\}^* \to G$ is a hash-function with random oracle properties. We call $H(i) = \tilde{g}_i$.

In the real world, we have $x_{i,j} = x_{i',j}$ for all i, i', i.e., a given meter uses the same blinding values for all measurements. In this case, we also denote $x_{i,j}$ as x_j. Let x_j

be the blinding value for meter j, and $c_{i,j}$ the measurement i for meter j. Thus, for measurement i, meter j sends

$$\tilde{g}_i^{x^j + c_{i,j}}$$

5.4 The Simulation

We will now construct a simulator that will use an adversary which can distinguish ideal– from real world to solve DDH. To this end, we introduce $l < n$ and $k < m$, and define that Meters $1, ..., l-1$ behave ideal. Meters $l+1, ..., n$ behave real. Meter l behaves

- ideal for measurements $1, ..., k-1$
- real for measurements $k+1, ..., m$
- for measurement k, it uses h_3 for the blinding.

Now we set $H(k) = h_1 = g^a$; this is where the random oracle property of H is required.

As the next step, we want to set $x_l = b$, even though the simulator only knows $h_2 = g^b$. To this end, we first assume that $l \geq 1$. All meters can behave following the protocol as is, and the first meter uses

$$\tilde{g}_1^{x_{i,1}} = \frac{\tilde{g}_i^X}{\prod_{j=2}^n \tilde{g}_i^{x_{i,j}}}$$

, i.e., it adapts its masking value such that all masks $x_{i,j}$ sum up to X.

Now, if $h_3 = g^{ab}$, meter l sends $g^{ab} = \tilde{g}^b = \tilde{g}^{x_l}$ as its blinding value, i.e., meter l behaves real for measurement k. Else, the blinding value it uses is random, and thus the meter behaves ideal for measurement k.

For $l = 1$, we have no meter left that behaves fully ideal and thus could adapt its randomness to simulate a proper run. However, note that in the case of $l = 2$,

$$\tilde{g}_1^{x_{i,1}} = \frac{\tilde{g}_i^X}{\prod_{j=2}^n \tilde{g}_i^{x_j}}$$

Thus, all values of $x_{i,1}$ used are identical, i.e., there is no difference between the last meter behaving ideal or real, as the real behavior of all other meters force it to behave real as well, and thus the ideal- and real run are indistinguishable by default in this case.

Therefore, we have the following lemma:

Lemma: Given above construction, any attacker that can distinguish whether meter l behaves real or ideal for measurement k, can also solve DDH.

Given this lemma, we can now use a diagonalisation argument to argue that full real behavior is indistinguishable from full ideal behavior. Suppose we have an attacker that can distinguish our real- from our ideal world setting with some advantage ϵ. We then provide that attacker with all our intermediate steps, where some meters/measurements behave real and the others behave ideal. This means there is some setup where the one individual measurement is decisive, i.e., the attacker will tend towards 'ideal' if that measurement is ideal, and towards 'real' otherwise. This is the setting where we can use our above simulator to turn it into a DDH decider.

6 Determining an Aggregate

The scheme as we described allows an aggregator to verify if an aggregate it already knows corresponds to the sum private measurement values it received. In many settings, however, an aggregator cannot measure the aggregated value - for example, a utility may be interested in the aggregate of the power output of all houses with photovoltaic energy generation, which are not connected to the same substation.

A typical smart meter reading is a four byte value. If we assume up to 250 devices in one group, that would give us a 40 bit value for the aggregated reading. However, in most cases, the aggregator has a fairly good idea on the rough value of the aggregater, as energy usage is fairly predictable - this would easily reduce the set of possible values into an area a normal computer can brute-force in a reasonable short time (Note that the brute force will only reveal the aggregate, while the individual contributions are still secure).

If the either the number of measurements of the measurement domain gets too big, the meters can easily split the measurement in a high- and low part and report both parts independently (of course, it can be split into any number of parts, depending on the measurement domain). The aggregator can then brute force both parts individually, reducing the computational effort on the back-end to a level it can handle in a practical setting. The only setting which this approach does to work

is if the aggregation is performed over a large number of devices, e.g., a million meters. In this case, however, the entire protocol can be run independently on different subgroups of the devices without any loss of privacy.

7 Key Establishment

The scheme as it is has no forward security, i.e., if a meter key does get exposed, the individual meter consumption from all measurements that used that key can be exposed if the attacker stored the corresponding data. This, too, would be a reason for frequent rekeying operations. We currently see no way to do this without meters (at least indirectly) communicating with each other. In that case, the first protocol presented in this paper can be used, using the masking values generated there to mask several measurements.

If the aggregator is behaving honest during initialization and changes in the set of participating meters, It is relatively easy in this scheme to add or revoke or add a device (essentially, it can surrender it's share of x_i and x'_i to the aggregator). This does not provide backward privacy, as it would allow the aggregator to decipher old messages. The alternative is for all meters in the new group to agree on new shares, which is doable with communication complexity $O(n^2)$ (if the meters trust each other in so far that none of them tries to disrupt the protocol and create nonsensical parameters.

8 Future Issues

In the current setting, our protocol assumes that a meter is not complete hacked, i.e., the values sent to the fraud detection system are the same that are sent to the billing system (e.g., as proposed in [4]. It would be worthwhile to link those systems cryptographically, and to assure that the same amount reported to fraud detection is also payed for. This would allow to not require any assumptions on the security of the meter anymore. A naive, but realistic approach is to use the same data used in the fraud detection to compute the aggregate bill of all meters in the set, and compare this bill with the aggregate of the bills payed by the corresponding consumers.

The second open issue is dealing with customers that want to opt out. By the very nature of the aggregation comparison, if any household connected to the substa-

tion does not participate in the smart metering scheme, the whole scheme does not work. Worse, if exactly one party opts out, the comparison between the aggregated sums and the measured values at the substation reveal that parties consumption, i.e., the one party that did not want a smart meter in their home is the one loosing their privacy in the end.

9 Acknowledgments

I would like to thank Michael John for insightful comments on the reality of smart metering, and Lejla Batina, Jaap-Henk Hoepman, George Danezis for helpful discussions and for taking the patience to read and comment on early versions of this papers.

References

[1] Flavio D. Garcia and Bart Jacobs, *Privacy-friendly Energy-metering via Homomorphic Encryption*, 6th Workshop on Security and Trust Management (STM), 2010.

[2] Molina-Markham, Andrés and Shenoy, Prashant and Fu, Kevin and Cecchet, Emmanuel and Irwin, David, *Private memoirs of a smart meter*, 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building, 2010.

[3] C. Cuijpers and B.-J. Koops. *Het wetsvoorstel slimme meters: een priva- cytoets op basis van art. 8 EVRM.* Technical report, Tilburg University, oct. 2008. Report (in Dutch).

[4] Alfredo Rial and George Danezis. *Privacy-friendly smart metering*. Microsoft Research Technical Report MSR-TR-2010-150, 2010.

[5] DIRECTIVE 2009/72/EC, European Parliament, 2009