Implementation of Privacy-Friendly Aggregation for the Smart Grid

Benessa Defend and Klaus Kursawe

European Network for Cyber Security
Prinses Beatrixlaan 800, 2595 BN The Hague, Netherlands
{benessa.defend,klaus.kursawe}@encs.eu
https://www.encs.eu

Abstract. In recent years a number of protocols have been suggested towards privacy-preserving aggregation of smart meter data, allowing electricity network operators to perform a large part of grid maintenance and administrative operations without having to touch any privacy-sensitive data. In light of upcoming European legislation, this approach has gained quite some attention. However, to allow such protocols to have a chance to make it into a real system, it is vital to add credibility by demonstrating that the approach scales, is reasonably robust, and can be integrated into the existing and planned smart metering chains. This paper presents results from integration and scalability tests performed on 100 DLMS/COSEM smart meters in collaboration with a meter manufacturer and a Dutch utility. We outline the lessons learned and choices that had to be made to allow the protocols to run in a real system, as well as some privacy challenges that cannot be covered by this technology.

1 Introduction

The introduction of smart grids promises big returns for energy efficiency and cost savings, but at the risk of consumer privacy. Privacy concerns have already caused pushback that has greatly delayed smart grid deployment and increased costs [1]. While there is not definitive agreement on what the use cases are for smart meter data – this can vary from country to country, and even differ within one organization – there are cases where grid companies need access to some level of consumption data in order to do load balancing, power maintenance, distribution automation, incident monitoring, and other grid maintenance functions. Access to this data will allow grid companies to run the grid more efficiently and react to modern trends such as local generation and the potential large-scale deployment of electric vehicles. In many of those use cases, aggregate consumption data gives the grid companies all the information they need to better monitor and maintain the grid without requiring individual readings that might infringe on consumers' privacy.

Using modern privacy preserving protocols, many of the privacy concerns can be mitigated, while providing the network operators with all of the data

they need for grid operations, potentially with even higher data quality. The most prominent class of protocols in this respect are aggregation protocols, in which the aggregate sum of a number of meter readings is computed without revealing the readings themselves. This is done by homomorphic encryption, i.e., encrypting the readings in a way that the encrypted ciphertexts can be added up, and the sum of ciphertexts – and only the sum– can then be deciphered to show the sum of the plaintexts. This approach, where applicable, is superior to collecting data and then restricting access to it (which creates a security problem as well, and has failed numerous times in the past) or anonymizing the smart grid data, which has also been shown to be insufficient [2]. To go one step further and make this technology applicable for a real roll-out, it is vital to show these advantages, demonstrate how aggregation protocols can be used in a real setting, and validate applicability, performance, and robustness. To this end, we implemented two protocols from Kursawe, Danezis, and Kohlweiss [3] on a real smart meter platform, and chose one of the protocols for a larger scale test involving 100 smart meters with one original head-end system.

1.1 Related Work

Although multiple publications [4–9] have presented aggregation protocols for smart meters, they provide only a theoretical analysis of performance and communication overhead. Rottondi et al. [10] and Li et al. [11] implemented smart meter aggregation protocols as a software simulations. Molina-Markham et al. [12] tested the performance of a microcontroller similar to ones used in smart meters for computing certified meter readings used for Zero-Knowledge Proofs in privacy-preserving billing protocols. To our knowledge, our work is the first to present results from integration and scalability tests performed on actual energy meters. Danezis et al. [13] presented protocols towards privacy preserving billing, as well as computing other functions on data generated by an individual meter. An implementation of this protocol on a smart meter has been done in conjunction with the aggregation protocols in the first implementation, though it has not been included in the scalability test.

2 Aggregation Protocol and Requirements

2.1 Smart Meter Restrictions

The most obvious restriction is the meter hardware. Meters are rolled out by the millions, and there is high pressure to deliver cheap solutions. Thus, a typical smart meter does not have a wealth of resources left in terms of computing power, flash memory for program code, and RAM. However, the meters we used for experiments were capable of standard public key operations in reasonable time, so if the protocols do not become too complex or require handling 4000 bit RSA keys, this is a solvable issue.

Bandwidth. There are different communication architectures in the smart meter area, primarily wireless (CDMA, ZigBee, GPRS) or power line communication. Depending on the actual technology used, as well as external factors (such as line noise), bandwidth can be extremely low, to the point of being measured in bits per second. Thus, protocols need to be careful about the communication overhead they add in terms of the number and size of messages.

Protocol Integration. The most surprising impact from our perspective was the requirement to make the protocol intergratable into existing standards. Communication protocols do tend to have maximum message lengths, and not all protocols allow for easy message fragmentation. A more mundane factor is compatibility with existing protocols and message handling. The protocol we chose to use does not change the message format of a smart meter reading since the encrypted values do not increase the length. This did come at the cost of flexibility, as this protocol is the least adaptable one, but allows very easy integration into the existing implementations on the meter, as well as into the message handling on the back-end side.

The other issue is the higher level workflow, e.g., do meters push data or is it pulled, who initiates a readout, and when and where is the data needed. This can, for example, cause synchronization issues if meters are triggered by local events and do not participate all at the same time. A meter that misses such an event will render the output of all other meters unusable, and - as the protocols need some form of readout identifier - it has to be assured that local events do not affect sequence numbers for other readings.

Security Architecture. In an ideal world, all smart meters would be implemented in a very secure way, and attacks through the standard implementation backdoors would be prohibitively difficult. In the real world, however, we can expect some meter implementations to have implementation weaknesses and for meters to have a sizeable risk of attack. One way to mitigate some of this risk is to constrain meter communication. In the Dutch Smart Meter Requirements (DSMR), for example, communication to local devices is done through a data diode to prevent any attacks originating from those devices. With this architecture in mind, a meter should definitely not be allowed to communicate directly with any other meter; a mesh network of meters immediately poses the risk of malware spreading between meters. This restriction also means that, ideally, a privacy protocol should require the meter to communicate only with its head-end, and avoid any direct communication between two meters. In the real world, the final verdict on this architecture is still out. Some communication protocols require meters to act as intermediate repeaters, which allow some communication between meters. Also, the idea of the meter as the hub of the "Internet of things" in a huge mesh network is still promoted, which aims for the opposite architecture.

Explaining the Protocol. For the industry to pick up a new cryptographic protocol, it is vital to be able to explain it in a way decision makers understand, and to build a business case around it. Worse, we did get a conflict with Gentry's fully homomorphic encryption scheme, which had just hit major newspapers

before our protocols were introduced. Thus, the people that had heard about homomorphic encryption immediately associated gigabytes of key material with it, and considered the entire idea to be far from practical use for now. In getting our experiments to run, we have spent significantly more time on marketing than on the actual tests. The first breakthrough came when we could explain the protocols using Lego bricks; the second one was when we could demonstrate that there are use cases where using privacy enhancing technologies is more beneficial than any alternative approach.

Standardization. Finally, for a protocol to actually be used, it does have to be standardized in some form and ideally become part of a larger standard. This adds a political aspect to the protocol design. Any patented technology is frowned upon (though they do routinely find their ways into standards), and the technology needs to provide a high level of stability and inspire trust. Implementers of the standard dislike the idea that some researcher came up with a slightly cooler protocol that now needs to be added too, new use cases that were forgotten in the initial standard, or a wealth of options that inhibit interoperability. This part is the hardest from a researcher point of view; while work is ongoing on improving the protocols, it is also important to stop in time and standardize a pragmatic and working solution. Otherwise, we will have the perfect protocol on paper in ten years, but hundreds of millions of meters with no PET protocols in the field.

2.2 Protocol Choice

In the early stage of the experiment, three protocols were implemented [1] on a smart meter: a Diffie-Hellman-based protocol [3] , a "Dining-Cryptographers" based low-overhead protocol [3], and a billing protocol [13]. All protocols performed well and extremely efficiently, and even the public key operations could be performed almost instantly. The Diffie-Hellman-based privacy aggregation (DiPA) protocol had the advantage that a meter only needs a single key to be part of an arbitrary number of groups. The price of this is some computational overhead on the meter (namely, one elliptic curve operation), some overhead on the receivers side (brute-forcing through a few hundred values), and an increased message size, as the encrypted messages now are composed of one or several points on an elliptic curve. The low-overhead privacy aggregation (LoPA) protocol uses simple additions on the meter with specially-generated keys. The price is that each meter needs a shared key with every other meter, which limits both the number of meters in one group and the number of groups to available memory. (In the actual protocol implementation another limit was imposed by message size – the key distribution was done in one message, which limited the number of groups to 27.) The billing protocol integrated well with DiPA and put similar requirements on the meters.

[1] A large amount of this implementation has been done by George Danezis from Microsoft Research.

The prime argument for the protocol choice in the end was the ease of integration. LoPA does not change the size of the measurement (as it is encrypted by just an additive temporary key), so no change needed to be made to the protocol stack on the measurement; two additional messages needed to be defined to implement the key distribution, but not for the measurement part. Similarly, on the head-end no additional functionality is required for the decryption part – as decryption is simply adding up the encrypted values, integration is comparatively easy.

While our first round of interviews with industry concluded that the protocol has sufficient flexibility to handle their use cases, the critical limit is the number of groups. If aggregation is used for a number of independent use cases, e.g., fraud detection (which aggregates over all meters attached to one substation), energy markets (which aggregates over meters belonging to a specific retailer), prediction of renewables (which aggregates only over local producers), the number of parallel groups might go beyond the limit that the protocol can handle given the available memory. Also, functionality added to the meter in the future through software updates might cut into available memory, limiting the group sizes even further. For the implementation discussed in Section 3, the firmware limited each meter to one group at a time; changes to the firmware can allow one meter to be in multiple groups simultaneously.

For now the billing protocol has not been implemented in the scalability test for similar reasons; while it has shown to be efficient, fitting it into the DLMS/COSEM stack would have required some major changes. In addition, though the protocol is extremely versatile, we were struggling to find a strong use case to use as a vehicle to promote the protocol, and some homework on the integration of this technology into the legal framework is still ongoing. Finally, while a test on aggregation protocols is meaningful once one actually aggregates, the billing protocol runs largely independently on every meter, so a large-scale test for this kind of protocol is not strictly necessary.

3 Experiments

Two types of tests were performed on the meters: the component test and the system test. The component tests measure performance of the LoPA protocol directly on the meter, excluding overhead from the network and communication with the head-end system. The system tests measure performance of the LoPA protocol including overhead from the head-end system and the network.

3.1 Component Test Lab Setup

The meter component tests were performed on the Elster AS300 meters with firmware version 8.11.1. The communication protocol used is the DLMS/COSEM stack, which is a widely-used industry standard. The tests measure values from the RS-485 interface under normal operation, including DLMS set/get. The equipment can measure time with millisecond granularity.

3.2 Component Test Descriptions

Tests C1–C5 were performed for a) 1 group of 3 meters and b) 1 group of 50 meters.

Test C1: Create a new key pair Measures the time required to create a new public-private key pair for one meter.
Test C2: Read public key Measures the time required to read out the meter's public key from storage.
Test C3: Create a new group key for all meters Measures the time to create a new group key for all meters in the group with the list sent via DLMS. It includes the time to calculate and save the shared secret.
Test C4: Create a new group key for one meter Measures the time to create a new group key for one meter in the group, including the time to calculate and save the shared secret.
Test C5: Encrypt one value Measures the time to encrypt one meter consumption reading.
Test C6: CPU Usage Measures the percentage of the CPU that is used for a) 1 meter with LoPA disabled and b) for 50 meters with LoPA enabled.
Test C7: LoPA Components - EEPROM Amount of EEPROM required by LoPA components (keys, etc.) in bytes.
Test C8: Encrypted Load Profile - EEPROM Amount of EEPROM required by the encrypted load profile in bytes.
Test C9: Encrypted Load Profile Capture Objects - EEPROM Amount of EEPROM required by the encrypted load profile capture objects in bytes.
Test C10: Free Space - EEPROM Amount of free space left in EEPROM.
Test C11: Code Size without LoPA - Flash ROM Amount of flash ROM required by the code without LoPA.
Test C12: Code Size with LoPA - Flash ROM Amount of flash ROM required by the code with LoPA.
Test C13: RAM Required without LoPA Amount of RAM required without LoPA.
Test C14: RAM Required with LoPA Amount of RAM required with LoPA, including the increased stack requirement, RAM required by the objects, etc.

3.3 System Test Lab Setup

The system test environment consists of the following main hardware and software components:

- 100 Elster AS300 DLMS/COSEM Meters, Version 8.11.1
- Energy ICT EIServer 8.11.9 Head-End System
- Digi PortServer TS 4 MEI
- Wireshark Network Protocol Analyzer, Version 1.8.4

EIServer can only measure time with a granularity of seconds. To get data in milliseconds, Wireshark network traffic captures were correlated with EIServer

data logfiles. The meter and firmware version are the same as the meter used in the component tests described in Section 3.1. Meter firmware version 8.11.1 records electricity consumption values in 15-minute intervals.

As discussed in Section 2.2, the head-end system transmits all of the aggregation group keys in one message. This message size is 4000 bytes, which limits the number of meters per group to 27. Thus, for the system tests we used 2 groups of 25 meters to approximate 1 group of 50 meters.

3.4 System Test Descriptions

For each test time is measured from the start of the command to the successful completion of the command in the head-end system. The number of bytes transmitted between the head-end system and the meters is also measured. The meters communicate with the head-end system via the serial line. Tests S1–S4[2] were performed for a) 1 group of 3 meters, b) 1 group of 25 meters, c) 2 groups of 25 meters, and d) 10 groups of 10 meters.

Test S1: Create new key pairs Measures the time required to create a new public-private key pair for all meters in the group.

Test S2: Read public keys Measures the time required for the head-end system to read the public key for each meter in the group.

Test S3: Create a new group key for all meters Measures the time to create a new group key for all meters in the group.

Test S4: Read encrypted consumption values Measures the time required to read the encrypted consumption values from the meter registers for 6 15-minute intervals for all meters in the group.[3]

4 Results

4.1 Component Tests

Table 1 shows the results of the component tests directly on the meter for performance and CPU usage. In addition to recording consumption values, an interrupt is triggered every second to execute regular tasks on the meter. These tasks need anywhere between 300–400 ms to complete on this meter, thus consuming around 31% of the CPU. For a healthy meter, the tasks executed every second should never exceed 1000 ms, or the meter will consider the next second as a "missed/lost" second. While the meter will not skip any tasks that were scheduled to take place in the "lost" interval, functions like threshold monitoring will not happen in real time.

[2] Due to the setup of the head-end system software, it is only possible to perform Tests S1-S3 for an entire group of meters, not for an individual meter. This differs from the component tests in which it is possible to measure the values for just one meter.

[3] Through firmware changes it is possible to record meter readings at a higher frequency, e.g., every 5 minutes.

Test C1 shows that the average time to create the public-private key pair for one meter does not significantly increase from a group of 3 meters to a group of 50 meters. In Test C2, the average time to read the public key is ∼1 ms.[4] The size of the meter's public key is 64 bytes and the LoPA-encrypted value is 4 bytes. Thus, reading 16 PET encrypted values would also take ∼1 ms.

As shown in Test C3, in creating the group keys for 50 meters, each meter receives 49 keys and the total average time required is 9100 ms. In theory, disregarding network delays and signal quality, the group keys can be updated every 10 seconds for 50 meters in a group.

Test C5 shows the average number of milliseconds required to encrypt one LoPA value. Since the regular meter tasks require up to 400 ms out of every second, around 600 ms are available for LoPA operations. Encryption of one value for a group of 50 meters takes 96 ms, so theoretically the largest group size is 300 meters. Every time the consumption values are recorded, the meter would spend ∼400 ms on regular tasks and 576 ms on LoPA encryption, for a total time just below the 1000 ms limit. While it is acceptable to have 1-2 "lost" seconds every 15 minutes (when consumption values are recorded), the meter manufacturer recommends keeping the total timing of tasks below 1000 ms. Considering that only about 31% of the CPU is being used, it may be possible to change the firmware to allow, for example, a group of 600 by splitting up the encryption tasks so that half execute in one second and the other half execute in another second. Then the total timing of tasks per second would stay below the 1000 ms threshold while allowing for groups larger than 300.

The results of Test C6 demonstrate the low overhead of LoPA. CPU usage increased by only 0.09% for a LoPA-enabled meter in a group of 50 meters compared to a meter with LoPA disabled. The CPU speed is 72 MHz, i.e., 72,000 cycles/second, and about 20,000 are spent on regular processes and 40,000 are idle. For 300 meters in a group (the maximum), every 15 minutes (900 seconds) about 1 second is when the CPU is being used at max. Thus, 899 seconds of every 900 seconds the CPU is not being used at its maximum.

Test	Description (# of meters)	1 Group of 3 Meters Average Time (ms)	1 Group of 50 Meters Average Time (ms)
C1	Create key pair (1)	124.9	126.875
C2	Read public key (1)	1	<1
C3	Create group keys (all)	360.8	9100
C4	Create group key (1)	173.8	181.232653
C5	Encrypt one value (1)	5.02381	96.708333
		1 Meter - LoPA Disabled	50 Meters - LoPA Enabled
C6	CPU Usage (1)	30.02%	30.11%

Table 1. This table shows the results of component tests C1–C6 on LoPA performance time and CPU usage. The value in parentheses refers to the number of meters in the group used in that test.

[4] The equipment cannot measure time under 1 ms.

Table 2 shows the memory requirements of LoPA in EEPROM, flash ROM, and RAM. The LoPA components and encrypted load profile information (Tests C7–C9) only use 8174 bytes of the EEPROM, leaving 183688 bytes of free space. Adding LoPA to the code only increases the size in flash ROM by 15608 bytes (Tests C11–C12) and the amount of RAM required by 4784 bytes (Tests C13–C14).

Since almost 70% of the time the processor is not being used to its capacity, a firmware change to allow splitting of encryption tasks would support much larger group sizes, which is then limited by memory. Since each meter's public key is 64 bytes and there are 183,688 bytes free in memory, there could theoretically be 2870 meters in one group or in multiple groups simultaneously.

Test	Description	Memory	Size in Bytes
C7	LoPA Components	EEPROM	5062
C8	Encrypted Load Profile	EEPROM	2920
C9	Encrypted Load Profile Capture Objects	EEPROM	192
C10	Free Space	EEPROM	183688
C11	Code Size without LoPA	Flash ROM	246084
C12	Code Size with LoPA	Flash ROM	261692
C13	RAM Required without LoPA	RAM	48568
C14	RAM Required with LoPA	RAM	53352

Table 2. This table shows the results of component tests C7–C14 on memory requirements with and without LoPA.

4.2 System Tests

Table 3 shows the results of the system tests that include overhead from the serial line communication and the head-end system. Table 4 shows the number of bytes transmitted between the head-end system and the meters during those system tests.

Test S1 sends a message to each meter to create a new key pair and Test S2 sends a message to each meter requesting the meter's public key, which is required to later send to all the meters in the group to create the aggregation keys. These two tests each take approximately the same amount of time to complete for a given setup.

The most time-intensive task is creating the aggregation group keys for all of the meters; each meter receives a message with the public keys of all the other meters in the group. This action takes on average 16 seconds for 3 meters, 107 seconds for 25 meters, 317 seconds (∼5.3 minutes) for 50 meters, and 287 seconds (∼4.8 minutes) for 100 meters. Due to a smaller group size of 10 meters per group for the 100 meter setup versus 25 meters per group in the 50 meter setup, Test S3 took less time for 100 meters than for 50 meters. Also due to group size and the total number of aggregation group keys transmitted, the number of bytes increases by only 20% between the 50 meter and 100 meter setups.

As shown in Test S4, reading the encrypted consumption values takes about 39 seconds for 3 meters, 94 seconds for 25 meters, 274 seconds (about 4.5 minutes) for 50 meters, and 429 seconds (∼7.2 minutes) for 100 meters. When the head-end reads the consumption values, it reads the 6 most recent values, which are 4 bytes each. Thus, a total of 72, 600, 1200, and 2400 bytes are read for 3, 25, 50, and 100 meters, respectively. As discussed in Section 4.1, the meter itself takes ∼1 ms to read 64 bytes. For 3 meters the meter takes ∼1.1 ms, for 25 meters ∼9.4 ms, for 50 meters ∼18.8 ms, and for 100 meters ∼37.6 ms.

Test	Description (# of meters)	1 Group of 3 Time (ms)	1 Group of 25 Time (ms)	2 Groups of 25 Time (ms)	10 Groups of 10 Time (ms)
S1	Create key pairs (all)	13588.8360	33759.3447	95786.9113	147946.0895
S2	Read public keys (all)	13642.5570	33451.7854	92019.1679	148378.2688
S3	Create group keys (all)	16244.3580	107500.9829	317488.8615	286593.5395
S4	Read encrypted values (all)	39025.3141	93798.9607	273972.2835	429223.3581

Table 3. This table shows the time results of system tests S1–S4 on performance. The time reflects the average of 10 runs of each test. The value in parentheses refers to the number of meters in the group used in that test.

Test	Description (# of meters)	1 Group of 3 Bytes	1 Group of 25 Bytes	2 Groups of 25 Bytes	10 Groups of 10 Bytes
S1	Create key pairs (all)	60968	528443	1058428	2095841
S2	Read public keys (all)	75504	664432	1259758	2615719
S3	Create group keys (all)	84098	1545730	3121189	3752896
S4	Read encrypted values (all)	289243	2518940	4999469	10024446

Table 4. This table shows the number of bytes transmitted between the head-end system and the meters during system tests S1–S4 on performance. The number of bytes reflects the average of 10 runs of each test. The value in parentheses refers to the number of meters in the group used in that test.

The meter will normally spend 10 ms more than the theoretical measurements from the component tests results given in Section 4.1 because it takes time for the meter to process the message. Thus, the system test time, including serial network and head-end system overhead, is approximately equal to the following equation: $SystemTestTime = ComponentTestTime + 10\text{ms} + Overhead$. The overhead of the network and head-end system can then be approximated by the following equation: $Overhead = SystemTestTime - ComponentTestTime - 10\text{ms}$. For example, the approximate overhead for creating a new group key for 50 meters is $317488.8615 - 9100 - 10 = 308378.8615$ ms, which is around 5.1 minutes. Comparing Table 1 and Table 3 shows that the LoPA protocol operations on the meter are extremely fast, and the head-end system and serial network introduce the most overhead.

5 Challenges

As with probably most new technologies, the biggest challenges stem from integrating them into the existing workflow and infrastructure. Our protocol choice and parameters were quite optimized for easy integration – other protocols can offer more flexibility and elegance, but would require adapting the existing communication infrastructure to a point that makes them hard to deploy.

5.1 Non-technical limits

While ideally all meter readings would be privacy-protected, it is not feasible at this time; too many business processes require individual meter readouts, and not all of them could be implemented using aggregation. Realistically, the best we can hope for is to add PETs as a more or less optional feature into the standard, and then use it instead of individual readouts wherever applicable. We may even conflict with other privacy approaches. For example, informed consent is required for a third party to request a meter readout. Thus, if too few consumers participate in this third party, there may not enough participants to aggregate over. Another example is the privacy filter proposed by the German BSI, where a meter gateway decides which information is allowed to pass on. In this case, it is not possible to meaningfully implement the privacy protocol outside of the gateway (which needs to understand what it passes on), while it is complex to implement it within (a gateway may serve a number of meters, and needs a rather heavyweight certification). Finally, even with a PET integrated, it is not possible to eliminate all trust – if an operator is willing to act in a criminal way, it is hard to prevent them from circumventing the privacy features. There are a number of ways an operator can spy on a specific individual – they can introduce "ghost" meters (other meters in the aggregation group that do not actually exist) or send firmware updates that deactivate the privacy features. While protecting against most of these attacks is in theory possible - meters can be registered by a neutral entity, the firmware update function can be programmed and certified to alert the user of a firmware change along with a hash of the new firmware, etc. – those protections are unlikely to be implemented in reality. Thus, if we assume that data operators (or third parties that can demand access) are flawed, they might get tempted once a large dataset exists. We also assume that those organizations are largely honest and playing by the rules.

5.2 Research Challenges

When discussing our approach with grid operators, we found a number of use cases that are not covered by the existing protocols.

The first issue involves local overload of a power line. Once more consumers produce energy, it is possible that a large amount of current is transferred between two households, without ever touching a substation – an example could be that someone runs a large solar farm, while her neighbor runs a similarly large cannabis plantation. In this case, her neighbor will consume all energy she

produces, which then only has to be transported between two houses. From a grid management point of view, this poses a problem. Power lines can degrade if they are overloaded, so the network operator does need to know if there is any line transporting a massive amount of current. In the above setting, this can only be detected by the two meters involved in this setting. It would thus be useful to have a way to detect that a line is overloaded, without revealing the exact difference between the consumptions of two meters.

A similar issue is voltage spike detection. If a meter is short circuited, it will measure a lower voltage than its two neighbors, whereas normally it should measure a voltage that is between its two neighbors' measurements. In this case, it would be good to have a protocol that can detect that such a spike exists, without revealing the actual differences.

A current hot topic in smart metering is the concept of consumer engagement (or energy experience). The idea is that a consumer that has a wealth of information about their energy usage will become more energy conscious, saving up to 8 percent of their current energy consumption. A trivial solution to this end would be to use only devices on the consumer side to provide this information, and thus keep all energy information localized. In many settings, however, this approach is not feasible due to either the meter lacking a local communication port, or the involvement of consumers that do not have access to the appropriate devices. To this end, it would be preferable to send a detailed paper bill with all required information (e.g., how a household consumed energy compared to its peers) in a reasonable privacy-protecting way.

Another issue on this point that is largely ignored is domestic privacy. The average electricity consumer is not one individual, but a household of several people. Thus, providing the consumer with detailed energy information about themselves is inadvertently providing them with information about other members of the same household – for example, if one member is abroad, it might show whether their spouse spent the nights at home or not. As the meter cannot possibly distinguish which member of the household is the data owner, this does pose a problem that seems rather difficult to resolve.

The final issue is a more political one. In our first implementations on a single meter, it was shown that a more complex protocol can be implemented on a real meter, and the private billing protocol from Danezis et al. [14] can be integrated seamlessly and efficiently. This approach would allow for more flexibility (meters can be in an arbitrary number of groups at the same time, and privacy-preserving billing can be integrated). While it would be preferable to have the more powerful options in a standard, the added complexity – partially implementation complexity, but also a more difficult integration into the existing back-end and communication software – make it more difficult to get the technology introduced into a standard.

6 Conclusions and Next Steps

This paper presents integration and scalability test results from the low-overhead privacy aggregation (LoPA) protocol that was introduced by Kursawe, Danezis, and Kohlweiss [3]. The LoPA protocol adds no significant extra costs and easily fits within the DLMS/COSEM protocol, a widely-used, industry-standard smart meter protocol. Scalability tests on the Elster AS300D meter show that the protocol increases the CPU usage from 30.02% to just 30.11%, the current firmware can theoretically support up to 300 meters in one group, and encryption of 16 consumption values takes ∼1 ms inside the meter.

Since these experiments were done over serial line communication, one next step is to perform experiments with meters communicating over GRPS or power line communication (PLC). Other potential next steps are to do a field trial and explore options for including the aggregation protocol as part of a standard for smart meters.

Acknowledgements. We would like to thank Alliander for financing the project, providing expert advise, and hosting the smart meter test lab. We also thank Elster, particularly Michael John, for their support and technical expertise in writing meter firmware and conducting the experiments. Also special thanks to George Danezis, who contributed significantly to the first implementations of the protocol.

References

1. Cuijpers, C., Koops, B.J.: Het Wetsvoorstel 'Slimme Meters': Een Privacytoets op Basis van Art (in Dutch). Technical report, Tilburg University (2008)
2. Jawurek, M., Johns, M. Riek, K.: Smart metering de-pseudonymization. In Twenty-Seventh Annual Computer Security Applications Conference, ACSAC 2011, Orlando, FL, USA, 5-9 December 2011, pp. 227-236
3. Kursawe, K., Danezis, G., Kohlweiss, M.: Privacy-Friendly Aggregation for the Smart-Grid. In: 11th International Symposium on Privacy Enhancing Technologies (PETS'11), pp. 175–191. Springer-Verlag, Berlin, Heidelberg (2011)
4. Erkin, Z., Tsudik, G.: Private Computation of Spatial and Temporal Power Consumption with Smart Meters. In: Proceedings of the 10th International Conference on Applied Cryptography and Network Security (ACNS'12), pp. 561–577. Springer-Verlag, Berlin, Heidelberg (2012)
5. Erkin, Z., Troncoso-Pastoriza, J.R., Lagendijk, R., Pérez-González, F.: An Overview of Privacy-Preserving Data Aggregation in Smart Metering Systems. In: IEEE Signal Processing Magazine. IEEE (2013)
6. Garcia, F.D., Jacobs, B.: Privacy-Friendly Energy-Metering via Homomorphic Encryption. In: 6th Workshop on Security and Trust Management (STM). (2010)
7. Li, F., Luo, B., Liu, P.: Secure Information Aggregation for Smart Grids Using Homomorphic Encryption. In: First IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 327–332. IEEE (2010)

8. Li, F., Luo, B., Liu, P.: Secure and Privacy-Preserving Information Aggregation for Smart Grids. In: International Journal of Security and Networks, Special Issue on Security and Privacy in Smart Grid, pp. 28–39. Inderscience Publishers, Geneva (2011)
9. Lu, R., Liang, X., Li, X., Lin, X., Shen, X.: EPPA: An Efficient and Privacy-Preserving Aggregation Scheme for Secure Smart Grid Communications. In: IEEE Transactions on Parallel and Distributed Systems, pp. 1621–1631. IEEE (2012)
10. Rottondi, C., Verticale, G., Krauß, C.: Implementation of a Protocol for Secure Distributed Aggregation of Smart Metering Data. In: International Conference on Smart Grid Technology, Economics and Policies (SG-TEP 2012). IEEE (2012)
11. Li, D., Aung, Z., Williams, J., Sanchez, A.: Efficient Authentication Scheme for Data Aggregation in Smart Grid with Fault Tolerance and Fault Diagnosis. In: Proceedings of the Innovative Smart Grid Technologies (ISGT), pp. 1–8. IEEE PES (2012)
12. Molina-Markham, A., Danezis, G., Fu, K., Shenoy, P., Irwin, D.: Designing Privacy-Preserving Smart Meters with Low-Cost Microcontrollers. In: Proceedings of the 16th International Conference on Financial Cryptography and Data Security (FC '12), pp. 239–253. Springer, Berlin, Heidelberg (2012)
13. Danezis, G., Kohlweiss, M., Rial, A.: Differentially Private Billing with Rebates. Technical report MSR-TR-2011-10, Microsoft Research (2011)
14. Rial, A., Danezis, G.: Privacy-Preserving Smart Metering. Technical report MSR-TR-2010-150, Microsoft Research (2010)