MASKER: Masking for privacy-preserving aggregation in the smart grid ecosystem

Georgios Karopoulos[a,*], Christoforos Ntantogian[b], Christos Xenakis[b]

[a] *Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece*
[b] *Department of Digital Systems, University of Piraeus, Piraeus, Greece*

Abstract

The introduction of information and communication technologies to the traditional energy grid offers advantages like efficiency, increased reliability, resilience and better control of demand-response, while on the other hand poses customers' privacy at risk. Aggregation of electricity consumption readings in intermediate nodes is needed for efficient network utilisation; however, by using information collected by a smart meter, an attacker can deduce whether a house is empty from its residents, which devices are being used, residents' habits and so on. Here, we propose a privacy-preserving aggregation protocol that obfuscates individual consumption readings, while at the same time allows their aggregation without loss of accuracy. The same protocol is easily extensible to support privacy-preserving customer billing as well. Our solution is lightweight and presents additive homomorphic properties based on standard and easy to implement cryptographic operations, while it does not require an always available trusted third party for its operation. Finally, we show that knowledge of the obfuscated values does not affect customer privacy, since they cannot reveal enough information for an attacker to infer real consumption values.

Keywords: Industrial communication, communication system security, data security, power grids, smart grid, power system security, privacy

*Corresponding author
Email addresses: gkarop@di.uoa.gr (Georgios Karopoulos), dadoyan@unipi.gr (Christoforos Ntantogian), xenakis@unipi.gr (Christos Xenakis)

1. Introduction

The smart grid is the result of the modernisation of the existing energy grid in such a way that customers, as well as utilities, have the ability to monitor, control, and predict energy usage. To this end, the EU has plans to replace at least 80% of its electricity meters with smart ones by the year 2020 [1]. Moreover, according to a US report [2], the smart meter installations in the USA have reached 50 millions of devices as of July 2014.

The advantages of the smart grid in a large scale are national energy independence, emissions control, and global warming combat. In the grid operator/utility level it enables more granular definition of pricing policy, better capacity and usage planning, increased resilience and protection against cyber-physical attacks, while it provides more flexibility to energy markets. Regarding customers, the smart grid will enable them manage actively their energy usage, control energy bills, and be involved as renewable energy producers.

Despite the numerous benefits from its adoption, the smart grid comes with several security and privacy concerns. Customers need to frequently send their energy usage to the utility, something that exposes them to privacy invasions. Here, we study energy metering data aggregation and its privacy implications. An example of open energy metering data is the website `bwired.nl`, which presents, among many other things, real consumption data in real time since 2008. Since the website serves a different purpose (i.e., promoting Home Automation) than ours, a large amount of personal data are exposed and provided without any privacy protection. From there, residents habits can be easily tracked by analysing the relevant smart meter data (gas, water, and electric consumption). Using such metering data, it is possible to determine the number of people living in a household, when they are absent, the TV program they are watching [3], even their religion or other habits based on profiling [4]. The aforementioned facts clearly support the opinion that fine-granular smart meter measurements constitute a serious privacy and security threat for energy consumers.

In order not to overwhelm utility servers with excessive traffic, the main approach followed by existing work is to use intermediate aggregators to locally aggregate consumption data for a geographic area before sending the result to the utility operator. An alternative approach is to group multiple messages together with a common protocol header before forwarding to the utility [5]. The comparison of the two approaches, based on simulations, has shown that the former reduced the total size of received messages on the utility side by 98.5% compared to the later [6]. Thus, in our proposal we chose to follow the first approach.

In the past few years, several privacy-preserving billing and metering data aggregation schemes have been proposed. The main goal of these schemes is to transmit electricity measurements to utility providers in a secure manner. A privacy-preserving aggregation protocol is defined as follows:

Definition 1 (Privacy-preserving aggregation protocol). *In the context of smart grid, a privacy-preserving aggregation protocol is a protocol that provides in-network aggregation of end-user consumption data, while at the same time protecting the privacy of individual consumptions and the linkage of a house with a specific consumption.*

Accordingly, the definition of an aggregation protocol as secure is as follows:

Definition 2 (Secure aggregation protocol). *An aggregation protocol is considered secure when it provides confidentiality, integrity, and non-repudiation of individual end-user consumption data.*

Privacy-preserving aggregation approaches need to protect customers from third parties, that wish to gain access to their consumption data when these data are in transit or are being processed. Also, they need to protect energy consumers against intermediate aggregators, since the latter cannot always be considered trusted. We argue that, regarding the aforementioned definitions, a privacy-preserving aggregation protocol should also be secure, so that consumption data are not disclosed or modified by unauthorised parties. Such schemes

have to meet several other requirements to be considered appropriate for supporting aggregation in the smart grid, like accuracy, scalability and efficiency. While a significant volume of work has been done on privacy-preserving aggregation for the smart grid, related schemes fail to satisfy all the necessary requirements, while in many cases they present drawbacks that make them inadequate for large scale deployment.

In this paper, we propose a privacy-preserving aggregation solution that responds to the aforementioned issues: (a) it preserves the privacy and security of energy consumers, and (b) it fulfils all requirements that are needed so as to be appropriate for the smart grid. Our proposal is a masking protocol where each smart meter shares a series of cryptographically generated pseudorandom values with the utility (in contrast to homomorphic-based, like [7, 4], or asymmetric encryption-based solutions, like [8, 9]; an extensive comparison of our solution with related work is provided in Section 6). These values act as masks and are used to obfuscate the real consumption readings of the smart meter. This way, an intermediate aggregator can provide the utility with an aggregated consumption from several smart meters without actually knowing the masks or the real consumptions. Moreover, no special aggregator architecture is needed, thus, intermediate aggregators can be organised in an abstract way allowing for multiple levels of aggregation. Later on, the utility subtracts the used masks from the total sum received by the intermediate aggregators, with the result being the real combined consumption of all relevant smart meters. This way, the only entity that has access to a real consumption value is the smart meter itself. Our contribution is a secure and scalable solution that imposes low computation overhead, and is easily implementable. Based on the entropies of consumptions and masked readings, we prove that there is no information leakage during the operation of our protocol. The evaluation of our method provides also computation analysis, showing that it imposes low overhead, especially on smart meters which have limited hardware capabilities.

On the smart meter side, these sensitive information are protected by utilising a Trusted Execution Environment (TEE) for storing data and executing

critical operations (similar to [10]). Regarding the reliability of the TEE, it is a technology that is being used for more than a decade [11]. Moreover, it is used by large manufacturers like Nokia and Samsung, and as stated in [12]: *"Almost every smartphone and tablet today contains a TEE like ARM TrustZone"*. The cost of a System on Chip (SoC) that incorporates a TEE is low; for example, Artik 5[1] from Samsung costs under 60$. The security assurances provided by the TEE differ slightly depending on the implementation; one of the most common is ARM's TrustZone (our proposal is based on Open-TEE which follows TrustZone's specifications). According to these specifications, the main building blocks of the TEE include: secure boot, secure storage, isolated execution, and remote attestation. These mechanisms provide confidentiality and authenticity of the executed code and stored data, integrity of CPU registers, memory and sensitive I/O, and proofs of trustworthiness to third-parties [11].

The rest of the paper is organised as follows. In Section 2, we present a reference architecture, the security model and requirements for privacy-preserving aggregation. We also present in more detail the motivation behind our work, based on the weaknesses we identified after studying existing work in the field. Next, Section 3 presents assumptions and the operation of our protocol; we also discuss issues related to its operation and how billing can be supported by our protocol. Section 4 demonstrates the evaluation of our proposal in terms of security and performance. Section 5 presents related work in the field of privacy-preserving aggregation, while Section 6 compares existing work with MASKER. Finally, Section 7 concludes the paper.

2. Background

2.1. Architecture

In this paper, we focus on high frequency metering data [8] that are used for demand response. In such cases, the exact measurement from each smart meter is not necessary for the utility in order to perform the required actions

[1] https://www.artik.io/modules/artik-520/

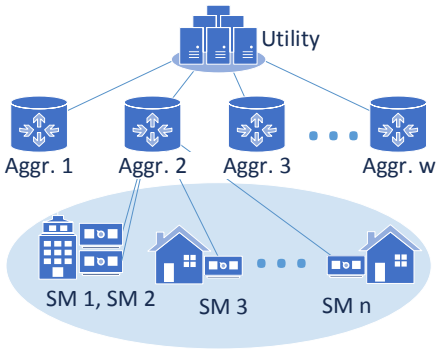


Figure 1: Smart grid reference architecture

(e.g. respond to electricity demand, load forecasting, and outage management); a collective sum from an area comprising a limited number of smart meters is enough. This makes it possible to protect the privacy of individual consumers, without breaking the correct operation of the demand response mechanisms. Nonetheless, our protocol can also be adapted for low frequency data [8] that are used for billing purposes, as discussed in Section 3.4.

When in-network aggregation is performed, only the aggregation result is transmitted, thus, the amount of data communicated in the network can be decreased, which consequently reduces the bandwidth consumption and the energy consumption for communication. In the simplest case, several smart meters transmit their consumption to an intermediate aggregator, which sums the individual consumptions and sends the result to the collector. The reference architecture shown in Figure 1 comprises the utility operator (i.e., *Utility*) which acts as a data collector, several intermediate aggregators that aggregate consumption data from smart meters (i.e., *Aggregator 1, Aggregator 2, ..., Aggregator w*), and finally smart meters that reside in a large area or neighbourhood (i.e., *SM 1, SM 2, ..., SM n*).

Utility. The utility/collector accumulates high-frequency (i.e., every 15 minutes) aggregated values. It can either use these values as is for demand response (e.g., control the electricity consumption in a specific area) or sum them

to come up with the total grid consumption. The collector is also responsible for billing by computing the total consumption of a customer at the end of a billing period (e.g., one month) using low-frequency metering data. In the rest of the text, the terms "collector" and "utility" will be used interchangeably.

Aggregators. The aggregators are intermediate nodes between the collector and the smart meters, which sum the individual readings received by the meters and transmit the result to the collector. This way, processing is distributed among the aggregators and data for demand response become available without putting too much load on the collector.

Smart meters. The smart meters reside in houses, or other buildings, and are responsible for collecting electricity consumption readings. The number of smart meters in each building varies, depending on the size of the building; usually, detached houses have one smart meter, while apartment buildings have one smart meter per apartment. In some cases, the architecture might not include explicit entities that perform intermediate aggregation, and smart meters can play this role as well.

2.2. Security model

In the following security model we consider the cases that are meaningful for privacy-preserving aggregation.

Collector. The collector/utility follows the honest-but-curious model, which is what most related works on privacy-preserving aggregation depend on. According to this model, the utility is assumed to follow the protocol properly, while at the same time it is possible to view the received data and try to extract private information from them.

Aggregators. The intermediate aggregators follow the honest-but-curious model as well. Aggregators are assumed not to drop or modify messages routed through them, allowing the system to run smoothly, while at the same time try to infer private information from the received messages.

Smart meters. The smart meters also follow the honest-but-curious model. Similar to aggregators, smart meters are assumed to send valid energy consump-

tion data to intermediate aggregators, and do not drop or modify messages routed through them. At the same time, they may attempt to retrieve or infer other smart meters' electricity usage data.

External adversaries. There is also the possibility of external adversaries that might want to: (a) determine a customer's consumption; (b) determine that a customer's consumption is zero or close to zero, which means that no one is in the house; (c) discover customer habits and activities that have detectable power consumption signatures, like watching television; (d) inject false data; (e) mount Denial of Service (DoS) attacks.

2.3. Requirements

Based on the aforementioned security model, as well as the operational goals of the smart grid, we specify the following requirements for a privacy-preserving aggregation protocol:

Intermediate aggregation. First of all, the protocol in question should preserve consumer privacy but at the same time support intermediate aggregation, i.e., accumulation of consumption readings from smart meters in intermediate entities and transmission of the result to a central collector. This requirement is necessary for network and processing efficiency, since a smart grid is expected to comprise millions of nodes.

Arbitrary architectures. An important characteristic of an aggregation protocol is the imposed architecture. Some protocols require a tree architecture or specific node grouping, making their deployment and operation troublesome in an environment like the smart grid, where nodes join and leave the network continuously. Thus, the support of arbitrary architectures is considered a significant advantage for a protocol.

Billing. A privacy-preserving aggregation protocol should operate in such a way so as to support billing of customers for a specified period (e.g., one month) as well. If not, a separate billing application will be required, something that will put unnecessary additional burden on smart grid nodes.

Data confidentiality. Metering data should only be known to the utility

and the consumer. This protection must be offered against honest-but-curious nodes (i.e., aggregators and utility) and adversaries that: (a) eavesdrop communication links, (b) have access to reports sent by smart meters and intermediate aggregators, and (c) collude to combine legitimately acquired information.

Message authentication. Intermediate aggregators should be able to link metering reports to specific smart meters. The utility should be able to link aggregation reports to specific aggregators.

Data integrity. The entities of the smart grid should be able to verify that the received data have not been altered en route.

Non-repudiation. A privacy-preserving aggregation protocol should preserve non-repudiation of the exchanged messages. There should be a way to prove that a metering or aggregation report was send by a specific smart meter or intermediate aggregator.

Efficiency. The communication and computational overhead imposed for privacy protection should be as small as possible.

Accuracy. An aggregation protocol should preserve consumers' privacy without affecting consumption data accuracy.

Adaptability. A protocol should adapt to nodes joining and leaving the network with minimum reconfiguration cost.

Scalability. A large number of smart grid entities should be supported, in the order of millions of devices.

No single point of failure. The smart grid has high availability requirements; thus, single points of failure that ease an attacker to mount DoS attacks should be avoided.

Physical protection. The customer or other third parties have physical access to the smart meter and potentially to other smart grid entities, like intermediate aggregators. This way, they can extract consumption values and cryptographic material used for protection of sensitive information. For instance, several attacks have been reported that manipulate the smart meter's firmware for energy theft [13]. Thus, special care should be taken to protect smart grid entities from unauthorised access to their hardware, software and

stored data.

Our main goal is to propose a secure and efficient privacy-preserving aggregation protocol for the smart grid that enables intermediate aggregation of consumption readings. The proposed protocol covers all the above requirements, and preserves end-user privacy in the presence of honest-but-curious entities. The protocol that we propose does not cover: (a) attacks against the protocol, where a scheme for detecting node misbehaviour would be more suitable, and (b) DoS attacks, where a more general solution is needed, covering all aspects of the smart grid, and not privacy-preserving aggregation only.

2.4. Motivation

The main motivation behind our proposal is that existing works in the field do not cover all the aforementioned requirements. We delay a detailed analysis of each solution's operation and inefficiencies until Section 5, so as to have the opportunity to compare related work with our protocol.

We have identified three broad categories into which existing works on privacy-preserving aggregation in the smart grid can be classified, based on the data protection method employed:

- homomorphic encryption,
- asymmetric encryption, and
- data masking

We should also note here that there is a recent survey on privacy-preserving smart metering [14] that uses a different classification approach. That survey focuses on architectural differences, examining mainly whether a Trusted Third Party (TTP) is used or not. In the following we briefly discuss the three aforementioned categories, summarising our findings from the literature review provided in Section 5.

Several existing schemes depend on *homomorphic encryption*, mainly due to its ability to allow aggregators sum encrypted readings, without actually knowing the plaintext value. Schemes belonging to this category present efficiency

issues due to the high computational overhead imposed by homomorphic encryption [15]. In [16], it is experimentally shown that computation on encrypted data using homomorphic encryption is several orders of magnitude slower compared to computation on plain data. Some of the schemes belonging to this category present scalability and lack of adaptability issues as well.

Solutions that are based on *asymmetric cryptography* present medium efficiency, due to the overhead of cryptography. The usual procedure is as follows: smart meters encrypt their readings and send them to intermediate aggregators, which decrypt these readings, compute their sum, encrypt the result and send it to the utility, and finally, the utility decrypts all these encrypted results. The decryption of all metering data from smart meters, however, entails many cryptographic operations compared to schemes where aggregation is performed directly on the encrypted/masked data. Such schemes make also use of TTPs either for anonymising customer IDs or for aggregating smart meter readings. Moreover, such protocols cannot ensure users' consumption privacy in the case of a honest but curious aggregator. In both cases, the TTPs constitute single points of failure and are susceptible to relevant attacks.

In solutions that are based on *data masking*, the real consumption is transformed with the use of light cryptography, to provide more efficient schemes. Usually, these schemes utilise some sort of symmetric cryptography, thus, the most common issue is the lack of protection against non-repudiation. Another common issue is the lack of adaptability, since most schemes require a particular node organisation or extensive key exchange among their nodes.

From the analysis provided in Section 5, it is deduced that none of the existing schemes on privacy-preserving aggregation can cover all fundamental requirements sufficiently. Our main goal is to propose a privacy-preserving aggregation solution that can meet all requirements set in Section 2.3, while at the same time be lightweight, considering the hardware capabilities of smart meters, scalable, and decentralised, so as to be appropriate for the smart grid.

3. Our proposal: MASKER

In this section, we propose MASKER, a protocol that prevents leakage of private energy consumption data to non authorised parties, while meeting aggregation requirements. First, we analyse the security assumptions upon which the operation of our protocol is based. Then, we give operation details about MASKER, as well as explain how it can support billing.

3.1. Assumptions

The proposed protocol follows the reference architecture and comprises the entities presented in Section 2.1. The main functional and security assumptions are the following:

Cryptography. Smart grid nodes are able to perform symmetric and asymmetric cryptographic operations. They are also able to produce non-predictable pseudorandom number sequences, i.e. sequences of numbers whose properties approximate the properties of sequences of real random numbers, based on appropriate sources of randomness.

Key management. Every smart grid node has a valid certificate in possession. A distributed, like [17], or other key management system appropriate for smart grids is needed.

Smart meter ID. The aggregator maintains a list of acceptable smart meter IDs which can send consumptions to this aggregator. It is not necessary for all these IDs to send their consumption for the protocol to function properly; it can operate with a subset of these IDs as well (in cases like smart meter unavailability and network outages). The list is necessary in order to avoid counting the same consumption more than once. The lack of such a list would only affect the accuracy of the results and it would not prevent the protocol from performing the aggregation operation.

3.2. Operation

The operational description of our proposal is divided into 4 phases: (a) preparation, (b) consumption masking, (c) intermediate aggregation, and (d)

unmasking aggregated consumptions. For demonstration purposes, in the most part of this description we have used AES-256 in counter mode as an example of a deterministic Pseudo Random Number Generator (PRNG) and consumption readings of 16 bits length; the protocol will have similar operation with different PRNGs and reading lengths. For protection against attackers that have physical access to smart meters and aggregators, we utilise a TEE [18]. The latter is an isolated, integrity-protected execution environment running along side a standard operating system (e.g., Linux). A TEE can provide several security-related operations including: (a) secure execution of applications running on top of a TEE (also called trusted applications), (b) generation of cryptographic keys, and (c) secure storage of sensitive data (e.g., user credentials, cryptographic keys). An application can execute non-critical (from a security point of view) code in a standard operating system and perform a transition to the TEE to execute security critical functionality encapsulated in the trusted application. In MASKER, we utilize a TEE in smart meters for: (a) generation and secure storage of cryptographic keys, (b) generation of random numbers, and (c) execution of the cryptographic computations for producing the masked readings (see Section 4.2). This way, we protect smart meters from attacks against the hardware or runtime attacks by malicious software, including attacks that try to influence the generation of keys by limiting their randomness.

Preparation. During this phase, the masking values are created, and the information used to create these masking values are sent to the utility. First, the smart meter selects a non-secret value V, which is used as an initialisation vector, and generates a secret key K in the TEE; both values have the same size. The two values are used together with a symmetric encryption algorithm in counter mode (like AES-256 as Counter mode Deterministic Random Byte Generator - CTR_DRBG [19]), in order to form a deterministic PRNG. Each masking value m is computed as follows:

$$m_{seq} = Enc(K, V + seq), \tag{1}$$

where $seq = \{1, 2, 3, ...\}$ is the sequence number of the masking value; the oper-

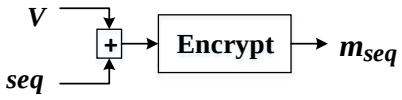


Figure 2: Masking value generation

ation is based on [19] and demonstrated in Figure 2. It is not necessary for the smart meter and the utility to create the whole series of masking values at once; they can produce each masking value only when it is needed.

Secret key K should only be known to the meter and the utility; for this reason, it is created by the smart meter and transferred securely to the utility according to the following protocol. K is (a) created, (b) encrypted with the utility's public key, and (c) digitally signed with the meter's private key, with all these operations taking place inside the TEE of the smart meter; then, it is transmitted directly to the utility. To assure its secrecy, K is always stored inside the TEE and never leaves the module in plaintext. Together with the encrypted K, the smart meter generates and sends V, which does not have to be secret, in plaitext. On the receiving end, the utility only has to verify the digital signature to decide whether to accept K or not, terminating the key exchange protocol. Later on, based on the values (K, V), the sequence numbers, and the PRNG, both the smart meter and the utility can create the same masking values. It is not necessary for intermediate aggregators to know the secret values, thus they do not take part in the transfer of K and V; as it will be explained later, they can perform aggregation operations directly on the masked data.

In general, PRNGs need reseeding before pseudorandom numbers start repeating. In the case of AES-256 in counter mode, reseeding is needed after every 2^{48} blocks [19]. In practice, assuming that a smart meter sends one usage report every 15 minutes, we need 96 masking values per day; therefore, a reseeding would be necessary approximately once every 8 billion years. However, according to NIST's recommendations [20], a master key (like K) used to derive other keys (i.e. the masking values) should be refreshed at least once every year.

Thus, the aforementioned operation of generation and secure transmission of K from the smart meter to the utility should be repeated every year.

Consumption masking. In this step, the smart meter sends its latest consumption reading to an intermediate aggregator. The detailed procedure is described in Algorithm 1. Please note that in our algorithms we produce one masking value for each reading, and each masking value is produced on-the-fly; this is the worst case scenario. For higher efficiency, a series of masking values can be produced in idle periods, and stored until used. Moreover, if masking values are larger than readings, we can divide each produced masking value into submasks and use these as masking values. For example, assuming that we use masking values of 128 bits and readings of 16 bits, we can divide a masking value into 8 submasks of 16 bits each; this way, we would need one AES encryption for every 8 reading transmissions (i.e., one every 2 hours for 15-minute aggregation intervals). In this case, the smart meter should also send the utility a pointer showing which submask is currently used.

Algorithm 1: Calculation of a masked reading

Data: Consumption reading r, last used sequence number $last\_used\_seq$, value V, symmetric key K
Result: Masked reading packet MRP

1 $seq = last\_used\_seq$;
// masked reading should be between T_{min}, T_{max}
2 **repeat**
3 $\quad seq++$;
4 $\quad$ generate masking value $m_{seq} = Enc(K, V + seq)$;
5 $\quad m\_reading = r + m_{seq}$;
6 **until** $m\_reading > T_{min}$ *and* $m\_reading < T_{max}$;
// store the used sequence number
7 $last\_used\_seq = seq$;
// send masked reading to aggregator
8 digitally sign data $d\_signature = (ID_i|m\_reading|seq)$;
9 create $MRP = (ID_i|m\_reading|seq|d\_signature)$;
10 send MRP to the aggregator;

The smart meter always keeps the last used sequence number $last\_used\_seq$; when producing the next mask, this sequence number is increased by one. The

masking value m_{seq} is generated by encrypting the sum of V and seq with the secret key K. The masked reading $m\_reading$ is the sum of the consumption reading r and masking value m_{seq}. If m_{seq} and r are numbers of equal size (e.g. 16 bits) then the resulting $m\_reading$ will be longer by one bit (17 bits in our example).

For as long as $m\_reading$ is below a threshold T_{min}, seq is increased by one and new m_{seq} and $m\_reading$ are produced. The reason for doing this is that, if both m_{seq} and r are very low, they result to a low $m\_reading$, and so an attacker might guess that the electricity consumption is very low and probably nobody is in the house. Threshold T_{min} can be selected by the system administrator and should be well above the minimum consumption of the house; an example is to use the minimum consumption observed in that house multiplied by two. Similarly, a threshold T_{max} is defined and, while $m\_reading$ is above this value, new m_{seq} and $m\_reading$ are produced. If an attacker observed the maximum allowed, or a close enough value, of variable $m\_reading$, that would meant that r and m_{seq} have taken their maximum allowed or very high values, thus, he could imply that the house has a very high or even the maximum allowed consumption. An attacker can deduce in this case when all members of the family are in the house, whether there are visitors, usage of high consumption household appliances, daily routine of the family, and all these remotely, without the need to observe the house physically. Also in this case, threshold T_{max} can be selected by the system administrator, and should be well below the maximum value that variable $m\_reading$ is allowed to take; an example is to use the highest consumption observed in that house multiplied by two. After that, $last\_used\_seq$ is updated to be equal to the seq that was finally used.

The reason we use thresholds and not modulo operations is because the latter are lossy operations after which it will not be possible to compute the aggregation of consumption readings. An existing work that uses such operations in some point of the aggregation, i.e., [21], is not directly comparable with our method since it uses multiplicative cyclic groups. The thresholds can be defined in a per-house basis by selecting the overall minimum and maxi-

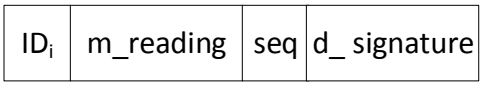


Figure 3: MASKER smart meter packet format

mum observed consumption values; this neither modifies the protocol nor they should be known to the aggregator and the utility. In the Evaluation section (Section 4), we show that these thresholds do not affect privacy; in fact, we have experimented with several other threshold values obtaining the same results. Thus, we argue that threshold selection is a simple enough operation that can be automated and selected by the smart meter based on the minimum and maximum observed house consumption and following some simple rules, like the *"two times the highest consumption"* used above.

In the end, the smart meter sends to the aggregator: the ID of the smart meter ID_i, $m\_reading$, seq, and a digital signature $d\_signature$ of the message for ensuring message authentication, integrity, and non-repudiation. For better performance, the message could be sent without a digital signature sacrificing non-repudiation; this, however, is not a realistic case and it will prevent the protocol from being used in real world scenarios. The smart meter ID is a serial number that is unique for the meter and, while there is no standardised format, it usually includes information like the meter manufacturer, year of production, and utility company; its size is between 8 and 10 numeric digits. The resulting masked reading packet format is shown in Figure 3.

Intermediate aggregation. In this phase, the aggregator verifies the received packets, aggregates masked readings, and sends the result to the utility; the procedure is presented in Algorithm 2.

The aggregator keeps a table Tab that contains smart meter IDs, as well as its own ID; for each ID_i, it stores the last seq_i used from the relevant entity. Furthermore, a second table is used, namely Sub, which stores IDs and seqs of smart meters that have submitted consumption data during the current metering period. For example, assuming that meters 1 to 10 belong to the area

Algorithm 2: Aggregation of masked readings

Data: Masked reading packets MRPs, table Tab
Result: Masked total consumption C_{tot}

1 $C_{tot} = 0$; // aggregated consumption
2 clear table Sub ; // meter IDs and masking value seqs
3 load table Tab ; // last used seq_i of each ID_i
// packets from smart meters
4 **while** *next packet* **do**
5 read received packet;
6 **if** $d\_signature$ *not valid* **then**
7 go to line 4;
8 **end**
// can meter transmit readings here?
9 **if** ID_i *not in valid IDs* **then**
10 go to line 4;
11 **end**
12 **if** ID_i *in* Tab **then**
// last seq_i used by this ID_i
13 $previous\_seq_i = Tab[ID_i]$;
// compare with seq_i from MRP
14 **if** seq_i *less than or equal to* $previous\_seq_i$ **then**
15 go to line 4;
16 **end**
17 **end**
// store seq_i used by meter ID_i
18 $Tab[ID_i] = seq_i$;
// Sub contains meter IDs and seqs of the current metering interval
19 $Sub[ID_i] = seq_i$;
// sum received masked readings
20 $C_{tot} = C_{tot} + m\_reading_i$;
21 **end**
// aggregator's seq is increased
22 $Tab[ID_{aggr}] + +$;
23 $seq_{aggr} = Tab[ID_{aggr}]$;
// send aggregated consumption and smart meters list to the utility
24 sign data $dig\_sign = (ID_{aggr}|seq_{aggr}|Sub|C_{tot})$;
25 create $AP = (ID_{aggr}|seq_{aggr}|Sub|C_{tot}|dig\_sign)$;
26 send AP to the utility;

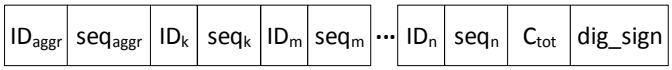


Figure 4: MASKER aggregator packet format

covered by an aggregator and have transmitted consumption data in the past, they are all stored in Tab together with the aggregator's ID and the last used seqs; if only meters 1 to 8 send consumption data during the current metering interval, then only these 8 smart meter IDs and corresponding seqs will be in table Sub. First, the aggregator checks the digital signature of each received packet for message authenticity and integrity violations. Each aggregator is responsible for aggregating masked readings for a predefined group of smart meters; for this reason the next step is checking if the smart meter ID belongs to the designated IDs. If Tab has no data for ID_i, it means that this is the first time the smart meter with ID_i sends reading data, and ID_i together with seq_i are added to the table. If Tab already has a record for ID_i, then the protocol continues only if the new seq_i is greater than the previous one (which is stored in Tab). Each masking value of a smart meter with ID_i has a sequential number seq_i to protect against replay attacks, and for informing the utility which masking value has been used during consumption masking. After the above checks, the masked reading transmitted by ID_i is added to the sum of masked consumptions C_{tot}, and the next masked reading packet is processed. When all packets have been handled, the aggregator increases its own sequence number seq_{aggr} by one; then it creates and sends to the utility a new packet containing ID_{aggr}, seq_{aggr}, table Sub, and C_{tot}, digitally signed by the aggregator. The format of this packet is demonstrated in Figure 4.

A similar approach can be followed when aggregation is performed in multiple levels. In such a case, multiple intermediate aggregators exist in the path between a smart meter and the collector. The main difference, with regard to Algorithm 2, is that aggregator packets are verified and processed instead of masked reading packets. The logic, however, stays the same: the received consumptions are summed and sent together with the list of the related smart

meters to a higher level aggregator or the collector.

Unmasking aggregated consumptions. In the last phase, there are two main alternatives: (a) the utility computes the total consumption of the smart grid, or (b) the utility computes the consumption of a geographic area covered by one or more intermediate aggregators. It should be noted here that the procedure stays the same regardless whether there is a single or multiple levels of aggregation.

In case (a) the utility sums all masked aggregated consumptions that it receives from intermediate aggregators, and removes the utilised masking values; the detailed operation is shown in Algorithm 3. First, the utility verifies the digital signature dig_sign of the received packet, and then checks if the aggregator's ID (ID_{aggr}) is in the group of valid aggregator IDs. Next, it checks the sequence number seq_{aggr} of the aggregated packet for the specific aggregator, which should be greater by one than the old sequence number of the same aggregator. Then, for each smart meter ID_i it generates the masking value that was used during masking, and adds it to the total sum of masking values M_{tot}. The sequence number of the aggregator is updated in table $AggSeq$, and the received masked aggregated consumption is added to the total $Masked_{tot}$. Finally, the real total consumption of the respective group of smart meters is the result of subtracting the total sum of masking values M_{tot} from $Masked_{tot}$.

For case (b), we can follow exactly the same procedure with a minor change. In line 3 of Algorithm 3, instead of using packets from all intermediate aggregators, it would be enough to consider the packet(s) of one or a limited set of aggregators, without further modifications to the algorithm. The result in line 22 will be the real aggregated consumption of the smart meters belonging to the single considered aggregator or set of aggregators.

3.3. Discussion

After having presented the operation of our proposal, in this subsection we clarify several issues that were not be covered above. First of all, we argue that our proposal is *privacy-preserving*, in the sense of Definition 1, since it:

Algorithm 3: Calculation of real aggregated consumption

Data: Aggregated packets AP_i, values V_i and keys K_i of all smart meters involved, table $AggSeq$ of aggregators' last used sequence numbers

Result: Real aggregated consumption $Aggr_{tot}$

1 $M_{tot} = 0$; // masking values' sum
2 $Masked_{tot} = 0$; // masked readings' sum
// packets received from aggregators
3 **while** *next aggregated packet* AP_i **do**
4 read received packet;
5 **if** dig_sign *not valid* **then**
6 go to line 3;
7 **end**
// can aggregator send data here?
8 **if** ID_{aggr} *not in valid IDs* **then**
9 go to line 3;
10 **end**
// last *seq* used by aggregator
11 $new\_aggr\_seq = AggSeq[ID_{aggr}] + 1$;
12 **if** seq_{aggr} *not equal to* new_aggr_seq **then**
13 go to line 3;
14 **end**
// generate and sum all meter masking values
15 **while** *next smart meter* ID_i **do**
16 $m_i = Enc(K_i, V_i + seq_i)$;
17 $M_{tot} = M_{tot} + m_i$;
18 **end**
// store aggregator's *seq*
19 $AggSeq[ID_{aggr}] = new\_aggr\_seq$;
// sum aggregated consumptions
20 $Masked_{tot} = Masked_{tot} + C_{tot}$;
21 **end**
// real total consumption is masked readings' sum minus masking values' sum
22 $Aggr_{tot} = Masked_{tot} - M_{tot}$;

(a) provides multiple-level, in-network aggregation, (b) protects individual consumptions by obfuscating their values, and (c) avoids the linkage of a house with a certain or approximate consumption value, as proved in Section 4.1. Additionally, it is *secure*, according to Definition 2, since it: (a) provides confidentiality, through value obfuscation, and (b) protects integrity and non-repudiation, using digital signatures.

Regarding the association of smart meter IDs with consumption values during the protocol operation, no additional privacy risks are introduced. This association is available to adversaries when a smart meter transmits its consumption to an aggregator; this value, however, is masked in such a way that the adversary cannot recover the real consumption value (as proved later in Section 4.1). On the aggregator side, the data associated with each ID are again masked consumptions, so that only the smart meter is able to recover the raw data (the aggregator is unaware of the real consumption values). The aggregator forwards a list of smart meter IDs when sending the aggregation of masked consumption values to the utility; here, there is no association of a single smart meter ID with a consumption value.

In the context of smart grid, errors or outages can affect the aggregation system. Under such circumstances, the affected smart meter(s) can easily recover and re-synchronise by performing a reset operation, which is similar to the initialisation process that is executed when a smart meter is booted for the first time. When a smart meter is initialised, it first acquires the utility certificate and creates its own certificate (using a key management process, which is out of scope of this paper) if these are not already available, e.g., in the TEE secure storage. It then creates a random seed for mask generation, which is encrypted with the utility's public key and signed by the smart meter's private key; all these operations are executed inside the TEE. Finally, the seq is set to 0 and the encrypted seed together with seq are sent to the utility. Compared to related work, MASKER presents less recovery and re-synchronisation overhead compared to systems which require the organisation of nodes into groups or aggregation trees.

Table 1: Aggregation and billing in MASKER

	T_1	T_2	T_3	...	T_M
SM_1	MR_{11}	MR_{12}	MR_{13}		MR_{1M}
SM_2	MR_{21}	MR_{22}	MR_{23}		MR_{2M}
SM_3	MR_{31}	MR_{32}	MR_{33}		MR_{3M}
...					

3.4. Billing

The description of MASKER is centered around aggregation; in this subsection we will show that MASKER can easily support billing as well, with only slight modifications. We will explain the billing operation based on Table 1. This table shows how a single aggregator can perform aggregation and billing for a number of smart meters in parallel. Each row contains masked readings from a single smart meter (SM_1, SM_2, SM_3, ...) and each column represents a time instance when aggregation is performed (T_1, T_2, T_3, ...); the last column represents the end of a billing period (T_M).

During aggregation, an aggregator sums masked readings from the predefined set of smart meters for a single time instance and forwards the result to the utility; for example, for T_1 this is the sum of the first column ($MR_{11} + MR_{21} + MR_{31} + ...$). For supporting billing, the aggregator will need to sum masked readings of a single smart meter for the duration of the billing period; for example, for SM_2 it will have to sum the second row ($MR_{21} + MR_{22} + MR_{23} + ...$) and send the result ($MR_{2M}$) to the utility. The aggregator does not need to keep all values for the total duration of the billing period, it just has to add them to MR_{2M} as they arrive. For example, in T_1: $MR_{2M} = T_{21}$, in T_2: $MR_{2M} = MR_{2M} + T_{22}$, in T_3: $MR_{2M} = MR_{2M} + T_{23}$ etc. At the end of the billing period the aggregator sends the T_M column to the utility. The utility can regenerate and sum the masking values used for obfuscating the real consumptions, and subtract them from the transmitted aggregation or billing result in order to compute the real aggregation or billing value.

4. Evaluation

4.1. Security

A well-known technique for masking data is random value perturbation [22]. This approach has also been applied in differential privacy models [23], which provide mathematically rigorous definitions of privacy. However, it is important to notice that, in the random value perturbation-based approach, the aim is to preserve data privacy by adding random numbers. The closer the perturbed data are to the original, the less confidential that data set becomes. Thus, there is a tradeoff between accuracy and privacy [24]. MASKER differs from random value perturbation mechanisms in the sense that we are able to accurately derive the statistical properties of the original data (i.e., the sum of the consumption values in our case) without using approximate estimation and therefore there is no information loss. Thus, MASKER is capable of maximizing the privacy level of the original data, without having to consider information loss.

Our solution also differs from secure multiparty computation (SMC) methods. In SMC, the interested parties jointly compute a result over their private inputs, while keeping those inputs private from each other. In MASKER, there is a honest-but-curious party (i.e., the aggregator) that computes a result over other parties (i.e., smart meters) inputs. In this setting, the smart meters do not have access to each other's consumption reading; also, the aggregator does not have access to smart meter consumption readings.

One basic question regarding our protocol is whether an attacker can deduce any useful information for electricity consumption just by seeing masked readings. To this end, we define this information as follows:

Definition 3 (Revealed information). *The revealed information of a privacy-preserving aggregation protocol is the useful information for electricity consumption that an attacker can deduce just by seeing masked readings, and its amount is equal to the mutual information $I(X;Y)$, where variable X represents consumption readings and Y represents masked readings.*

The value of the *revealed information* should be zero or very close to zero, in order to signify that the attacker cannot infer real consumption readings by observing masked readings.

We prove that the revealed information in our case is negligible, and that the proposed protocol can be safely employed in real implementations. This evaluation covers one-off attacks, where the adversary tries to deduce useful information from the observation of a single value, as well as attacks that combine multiple observations. In the rest of this section, we calculate the average amount of information about consumption values that is revealed by masked readings. This is in analogy to [22], where the authors measure the knowledge gained by an adversary for a sensitive value by seeing the distorted sensitive value.

In order to evaluate our method, we first need a set of consumptions that are close to real world values. For simplicity we use one metering value per hour, while the usual reporting interval is 15 minutes; however, this does not influence the results and the same procedure can be applied for 15-minute intervals. We use real, publicly available datasets of hourly electricity consumptions from the European Network of Transmission System Operators for Electricity (ENTSO-E) for 27 European countries during November 2014 [25]. From these raw data, we compute the mean hourly consumption for a 4-person house. We observed that consumption values follow the normal distribution, therefore we apply the *empirical rule* ($\mu - 3\sigma \leq \mu \leq \mu + 3\sigma$) to compute the range of consumption values for each time of the day, where μ is the mean consumption and σ the standard deviation. According to the *empirical rule*, 99.7% of the real consumption values will reside between the minimum ($\mu - 3\sigma$) and the maximum computed consumption ($\mu + 3\sigma$). A graphical representation of the resulting load curve, including mean, minimum, and maximum loads, is represented in Figure 5.

In order not to repeat the same consumption values again and again, in each interval, as a smart meter reading, we use a consumption value randomly selected from within the consumption range for this specific interval (e.g. for

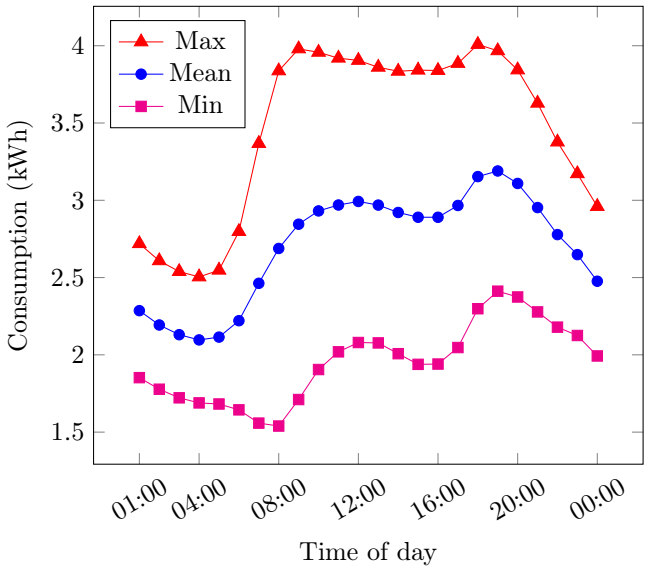
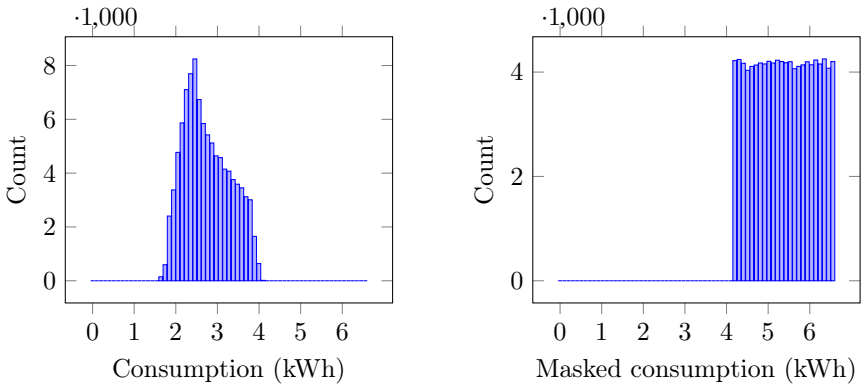


Figure 5: Hourly consumption for a single house (with raw data from ENTSO-E [25])

11:00 AM a consumption between 2 and 3.9 kWhs). For simplicity we use 2-byte integers for consumption and masking values; for ease of calculations we represent consumption values as integers by multiplying the value by 10,000, thus for 4 kWh we get the integer 40,000. In our experiments $T_{min} = 40,960$ which means that the masked reading will be above the maximum consumption value which is 4 kWh as seen on Figure 5. We use 40,960 instead of 40,000 in order to fit the values in "value bins" as explained later in this section. Our upper threshold T_{max} is equal to 65,535, so that masked readings fit into 2-byte integers as well. It should be noted here that the selection of thresholds T_{min} and T_{max} is not related to the distribution that the consumption values follow; the knowledge of the minimum and maximum consumption values observed is enough for the selection of proper thresholds, as discussed in Section 3.2.

For the random numbers to be used as masking values, we utilised CTR-DRBG that is standardized by NIST [19]; the implementation was based on the *mbed TLS* library (former *PolarSSL*) [26], providing AES-256 in CTR_DRBG mode.



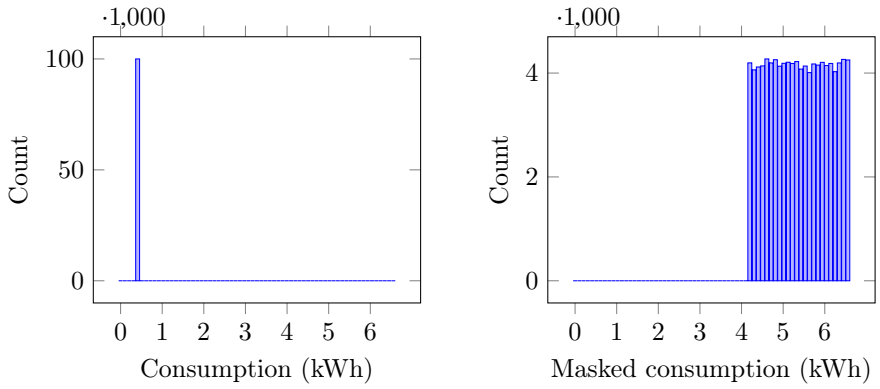
(a) Consumption distribution (b) Masked readings distribution

Figure 6: Distributions for normal electricity consumption

The final step of our evaluation was to collect all consumption and masked reading values, and check for their statistical correlation; we collected approximately 100,000 values from each type. A high correlation of these data would mean that an attacker can predict the consumption values, just by observing the transmitted masked readings. For correlation evaluation we utilise *mutual information* from Shannon's theory [27, 28].

We recall that, as described in Definition 3, consumption values are represented by a random variable X, and masked readings by Y. What we want to prove is that the knowledge of Y cannot help an attacker infer X. We define "value bins" of size 1024, and count how many consumption values we have in each bin; each bin represents a consumption of approximately 0.1 kWh. The resulting bins are: 0 - 1023, 1024 - 2048, ..., 64512 - 65535; the consumption values distribution is shown in Figure 6a. The reason we do this is because it is not necessary for an attacker to guess the exact consumption value, in order, for example, to infer if someone is at home; it is enough to infer that the consumption value falls in the first bin.

Definition 4. *The mutual information $I(X;Y)$ between two random variables X and Y is the amount by which the uncertainty (entropy) about X is reduced*



(a) Consumption distribution (b) Masked readings distribution

Figure 7: Distributions for always low electricity consumption (0.5 kWh)

by learning Y:

$$I(X;Y) = H(X) - H(X|Y) \tag{2}$$

It can also be shown that $H(X|Y) \leq H(X)$, with equality if and only if X and Y are statistically independent. Thus, it is straightforward that $I(X;Y) = 0$ for statistically independent variables X and Y.

We computed the relevant entropies from the collected values and found that:

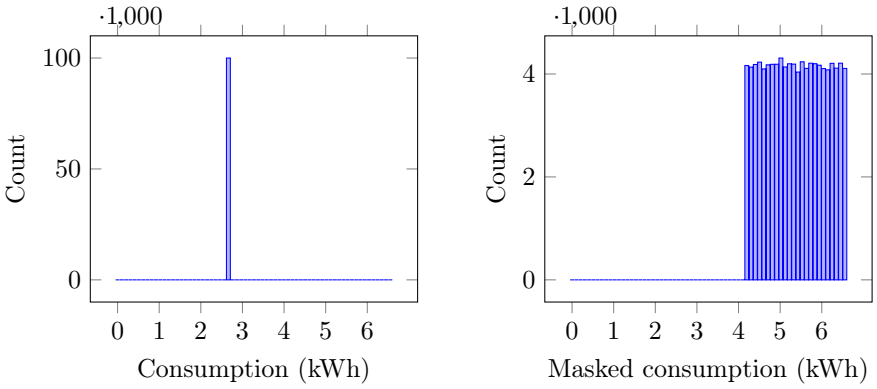
$$H(X) = 4.3633, \quad H(X|Y) = 4.3592 \tag{3}$$

The *mutual information* is:

$$I(X;Y) = H(X) - H(X|Y) = 0,0041 \tag{4}$$

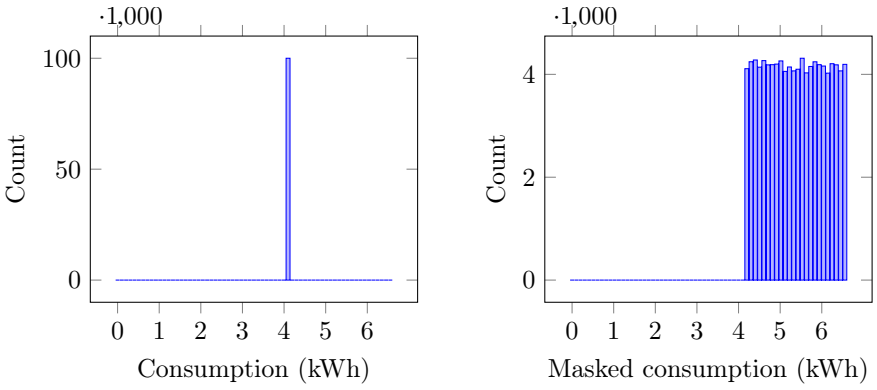
From (4) it is evident that the uncertainty (entropy) about consumption values is reduced only by a very small amount by learning masked readings; thus, we argue that, in practice, the knowledge of masked readings cannot help the attacker infer real consumption values.

The minimal correlation of consumption values with masked readings is also demonstrated by comparing the distributions of consumptions and masked readings. Figure 6a shows the distribution of consumption values using the procedure described above; the distribution is not uniform, as expected from the load curve



(a) Consumption distribution (b) Masked readings distribution

Figure 8: Distributions for always mean electricity consumption (2.7 kWh)



(a) Consumption distribution (b) Masked readings distribution

Figure 9: Distributions for always high electricity consumption (4.2 kWh)

of Figure 5. In Figure 6b, the distribution of the respective masked readings is presented. Here, we mostly have a uniform distribution of values among the allowed values (i.e. from 40,960 to 65,535), making it difficult to guess which masked readings are related to low consumption values. We repeated our experiments with three distinct extremes: (a) constant low consumption (0.5 kWh), (b) constant mean consumption (2.7 kWh), and (c) constant high consumption (4.2 kWh). Again, we considered 100,000 of consumption values in each case and the results show that in all cases masked readings distribution is uniform. Figure 7 shows the low, Figure 8 the mean and Figure 9 the high consumption distributions respectively. Overall, it is not possible for an observer, which has access to the masked readings, to conclude whether the consumer has a constant low, constant high or variable consumption pattern. This way, it is not possible for an attacker to deduce useful information by the observation of a single masked value or the combination of multiple values over time.

4.2. Implementation and Performance Evaluation

We measured MASKER's performance on smart meters and intermediate aggregators experimentally. In most cases these will be embedded devices with low processing capabilities, thus, it is imperative that their performance should not be affected significantly by MASKER. On the other hand, we do not evaluate the performance of the utility, since the latter is considered to be a high-end computer, capable of handling high load efficiently.

To this end, we have implemented[2] two separate programs in C: (a) the algorithm for calculating the masked readings (which is executed by smart meters), and (b) the algorithm for aggregating the masked readings (which is executed by intermediate aggregators). Regarding the first program, which is executed in the smart meters, it has been implemented as a trusted application (i.e., it is executed inside a TEE) using Open-TEE [29]. The latter is a virtual TEE implemented in software. A key characteristic of Open-TEE is that it facili-

[2]https://github.com/gkarop/masker

tates interoperable implementations, since it conforms to the GlobalPlatform specifications [30] for TEE and it is also independent of any specific TEE hardware environment. This allows MASKER's trusted application operations to be executed on any TEE implementation that adheres to the GlobalPlatform's specifications (e.g., ARM TrustZone [31]). In our implementation, the trusted application includes the following functionality that is executed inside the TEE: (a) generation of AES and RSA keys, (b) generation of the masking values using AES-256 in counter mode, (c) computation of the masked readings (i.e., the sum of the consumption value and the masking value), (d) digital signing of the smart meter packet using 1024-bit RSA, and (e) storage of the relevant AES and RSA keys. On the other hand, the standard operating system (i.e., Linux) is used to perform non-critical operations (i.e., obtaining the consumption values, providing the output, etc.).

Regarding the second program, the aggregation of the masked readings has been implemented without utilizing Open-TEE, since we can assume that it is more difficult for the attackers to have physical access to them (although we note here that the implementation of the aggregators can be easily modified to support Open-TEE).

Both programs were executed on a Raspberry Pi (with a single core 700MHz CPU, and 512MB @ 400MHz SDRAM) to emulate an embedded device. Consumption measurements were based on the data presented in Section 4.1. The aim of the experiments was to estimate the maximum CPU utilization, total execution time and virtual memory usage by MASKER. In our experiments, a smart meter creates and sends to an intermediate aggregator a digitally signed masked reading every 15 minutes. On the other hand, the intermediate aggregator receives, verifies and processes 1,000 masked readings every 15 minutes (note that we have used simulation in order to send 1,000 measurements to the aggregator). Table 2 summarizes the results.

We observe that the maximum CPU utilization is negligible in the smart meter (5.1%), while for the aggregator it is a bit higher (6.6%). The execution time in smart meters (i.e., the time required to calculate a masked reading) is

Table 2: MASKER overhead

Metric	Smart meter	Aggregator
Max CPU utilization	5.1%	6.6%
Execution time (sec)	0.05	0.17
Memory consumption (KB)	4,812	5,016

0.05 seconds using Open-TEE, while in the aggregator (i.e., the time required to aggregate 1,000 masked readings) is 0.17 second. Finally, the virtual memory consumption is 4,812 KB in the smart meter and 5,016 KB in the aggregator.

The numerical results show that the overhead of MASKER is low and suitable for smart grid devices; however, we note that further reduction to these numbers can be achieved using multi-threading and code optimization. Moreover, we can improve the performance of MASKER by removing the execution of RSA, which is the most costly operation (in terms of CPU overhead), due to the modular exponentiation performed for the creation and verification of RSA signatures. Instead of RSA, MASKER can adopt the use of an authenticated AES scheme, such as AES-CCM (Counter with CBC-MAC) mode [32], in order to provide not only privacy but also integrity of the readings at the same time. However, the downside in this case is that MASKER will not be able to provide non-repudiation.

5. Related work

In this section we analyse related work following the classification we defined in Section 2.4. Existing work in the field is extensive; thus, we selected a subset based on the following criteria to create a list that: (a) covers all categories, (b) includes the most highly cited works in each category, and (c) includes some more recent solutions. Following this procedure, we had to leave out works like the following, which are referenced here for the sake of completeness: [33, 34, 35, 36, 37, 38].

Table 3 presents a comparison of existing works based on the requirements defined in Section 2.3; the comparison is based on the following signs: +, ∘, –,

(+), (∘), (−). The plus sign shows that the respective requirement consists an advantage of the method over the others, in the sense that the method fulfils the requirement. A small circle shows a neutral capability of the method or that a requirement is partially met. The minus sign shows that the requirement is considered an inefficiency of the method, in the sense that the requirement is not met. When the three signs above are put into parentheses, it means that the respective work does not include all the details needed, and we had to make assumptions to come to a conclusion.

The first category of schemes is based on homomorphic encryption. In [7], the aggregation is actually performed by smart meters that reside on the way to a collector node, in an in-network incremental aggregation approach without dedicated aggregators. In each neighbourhood, an aggregation tree is constructed where the collector node is the root. A careful network design for the tree is essential for the method to perform well; according to the authors: (a) the height of the tree should be small for communication efficiency, and (b) an internal tree node should not have too many children for computation and communication efficiency. The design choice of not having dedicated aggregators poses a high overhead to low resource smart meters, for executing several homomorphic operations; high overhead is also applied to the utility, due to homomorphic decryption. The imposed tree architecture has an impact on adaptability, while other alternative architectures are not supported. Moreover, the two design requirements (a) and (b) mentioned above affect scalability; as the number of nodes increases, special care should be given to fulfil them. Finally, there is no support for privacy-preserving billing.

The approach proposed in [4], comprises homomorphic encryption and additive secret sharing. The aggregator sends the public key certificates of all users in the neighbourhood to all of them. Each meter divides its readings into shares, encrypts each one with a different public key except one, and sends them all to the aggregator. The aggregator aggregates $N-1$ shares that are encrypted using the public key of a smart meter, and forwards this encrypted value to the smart meter, repeating the same procedure for all meters. Each smart meter decrypts

Table 3: Requirements fulfilment of related work

Scheme	[7]	[4]	[39]	[40]	[8]	[9]a	[9]b	[41]	[42][3]	[21]	MASKER
Intermediate aggregation	+	∘	+	−	+	+	−	+	+	+	+
Arbitrary architectures	−	−	+	−	+	+	+	−	−	+	+
Billing	−	+	−	−	+	+	+	−	−	−	+
Data confidentiality	+	(−)	+	+	−	∘	∘	∘	+	+	+
Message authentication	+	+	+	+	+	∘	(−)	(−)	+	+	+
Data integrity	+	+	+	+	+	∘	(−)	(−)	+	+	+
Non-repudiation	+	+	+	−	+	(−)	(−)	(−)	−	+	+
Efficiency	−	−	−	−	∘	∘	(∘)	∘	+	∘	+
Accuracy	+	+	+	+	+	+	∘	+	+	+	+
Adaptability	−	−	+	−	+	+	+	−	−	+	+
Scalability	−	+	+	−	+	+	+	+	+	+	+
No single point of failure	+	+	+	+	−	−	+	+	+	+	+
Physical protection	−	+	−	−	−	−	−	−	−	−	+

[3] The *low overhead protocol* is considered here

Legend:

+	advantage	(+)	advantage under assumptions
∘	neutral	(∘)	neutral under assumptions
−	disadvantage	(−)	disadvantage under assumptions

the received message, adds the value that held back in the previous step to this, and sends the new outcome back to the aggregator which aggregates all received values resulting in the total consumption of the area covered by these meters. Here, only the first level of aggregation is analysed (i.e., from smart meters to aggregators/substations) and no details are provided about whether there is a collector at a second level or not. This approach requires the formation of smart meters into specific groups to operate, thus, it cannot support arbitrary architectures. This method poses: high computation overhead to smart meters due to several homomorphic encryptions and decryptions performed and medium overhead to aggregators since they only perform homomorphic additions. Under the assumption that, at least, $N-2$ of the smart meters belonging to the same group/neighbourhood and the respective aggregator are curious, then they can collaborate in order to reveal a single smart meters consumption affecting data confidentiality. While this can be hard for big groups (large values of N), it can be easier in rural areas. Finally, this method presents adaptability issues during node join and leave, since new smart meter groups must be formed and exchange digital certificates.

EPPA [39] follows a more straightforward approach, where asymmetric homomorphic encryption is used; smart meters encrypt their consumption data, the aggregation is performed by the aggregators directly on these encrypted data, and the result is forwarded to the utility. The main difference from other similar methods is that EPPA can be performed on multidimensional data, which include energy consumption, consumption time, and purpose, processing them all together in the same round. Thus, it achieves better overall performance compared to traditional one-dimensional methods, which have to process every dimension separately. Regarding computation cost, smart meters present high overhead due to homomorphic encryptions, aggregators medium overhead since they only perform homomorphic additions, and utility servers high overhead because of homomorphic decryptions. It is worth mentioning that no billing support is described in the article.

Another scheme, which is based on homomorphic encryption, is [40]; here bi-

homomorphic encryption is used, which is additively homomorphic on both the plaintext and key space. Initially, smart meters are organised into groups, where one smart meter per group is periodically designated as the key aggregator. Each smart meter encrypts its reading with a key, and sends the encrypted reading to the utility through a secure channel. The key is sent to the key aggregator through a secure channel as well. Smart meters do not have access to the individual readings, while the utility does not have access to the individual keys. The key aggregator sends the aggregated key to the utility through a secure channel, so that the latter can decrypt the aggregated consumptions. This way, the utility does not have access to the plaintext readings of each smart meter, but only their aggregation. This scheme presents high complexity, while the organization of the groups and the election of the key aggregation node is not discussed and poses an attack vector. Smart meters are expected to have high computation overhead, due to homomorphic encryptions and the burden put on key aggregators; the utility will also have high computation overhead, due to homomorphic decryptions and lack of aggregators, which causes all smart meters to connect directly to it. Non-repudiation is not preserved, since symmetric encryption is used. This scheme presents also adaptability issues, because of the reconfiguration needed when nodes join or leave the network, as well as scalability issues, since, for large numbers of smart meters, the utility will need to perform a large number of aggregation operations.

The next category of solutions is based on asymmetric cryptography; one of the most well-known and cited works here is [8], where two meter IDs are created: an anonymous, and an attributable one. The manufacturer of the smart meter knows both IDs, while the utility knows only the attributable one; correspondingly, each smart meter has two profiles, using two different certificates: an anonymous and an attributable one. During the bootstrapping phase, each smart meter communicates with the TTP to send both the anonymous and attributable profiles/certificates, and then the TTP sends the anonymous profile/certificate to the aggregator. The key management is performed by a Certification Authority (CA) and a TTP is used to keep the correspondence

between the two IDs, and authenticate the anonymous readings received by the smart meters. The high-frequency, privacy critical measurements (the vast majority of metering data) are send to the designated aggregator using anonymous identifiers, while low-frequency, billing related measurements are reported with known smart meter identifiers. In the former case, the aggregator decrypts measurements, performs the aggregation and forwards the result to the utility. The efficiency of this solution is medium due to the use of asymmetric cryptography for consumption reading transmission: smart meters encrypt consumption readings, aggregators decrypt them and encrypt the aggregation result, and the utility decrypts the received aggregated results. The main issue with this scheme is that both the TTP and the smart meter manufacturer should be fully trusted; if they behave maliciously and leak the correspondence between the two profiles or use it for their own benefit, then confidentiality is at risk. Moreover, the privacy offered is not adequate according to [43], since it is possible to correlate pseudonymous consumption traces with real consumer identities, either by identifying anomalies in consumption during unusual events (e.g., a house repair), or by linking consumption patterns. Finally, the TTP is a single point of failure, and should be adequately protected.

In [9], a solution that is based on asymmetric encryption is proposed, where smart meters send encrypted readings to a TTP; the latter forwards to the utility only the aggregation of these readings. Regarding billing, the TTP sums up the readings of each smart meter and sends the result together with the identity of the meter to the utility. The computation overhead of this scheme is medium for all involved entities, due to the many cryptographic operations involved. The TTP (which is also a single point of failure) should be fully trusted, since it has access to all customers' data and identities; this, however, does not follow the usual honest-but-curious model, and can create issues in data confidentiality, authentication, and integrity. Moreover, in the paper it is stated that readings are sent using "some form of encrypted communication" without any further details; this means that non-repudiation might not be preserved, depending on the protocol used.

In the same paper, a second scheme is proposed, that is based on data masking. The "data masking" category comprises solutions that transform real consumptions using light cryptography, in order to provide more efficient schemes; MASKER falls into this category as well. In the second scheme found in [9], no TTP is used, and each smart meter masks readings with random values before sending them to the collector; this random noise follows a known distribution, such that the final aggregation will be close to the real one. Therefore, using this method the smart meters do not actually send accurate consumption data to the utility, but a close approximation of their aggregation. This method does not perform any intermediate aggregation; smart meters send directly consumption readings to the collector with all the implications this has for network utilisation. The computation overhead of the smart meters is expected to be low, since it is a masking method; this, however, is not clear, since there is no information about the security of the channel used to transmit masked values to the utility, something that can also affect message authentication, data integrity and non-repudiation. Regarding the utility, it is expected to have medium overhead, since all smart meters will be directly connected to it. The main issue here, according to the authors, is that large groups of smart meters are needed, in order to achieve an acceptable level of privacy. This can have two consequences: inadequate confidentiality protection when small groups are used, and adaptability issues during node join and leave for group formation. Also, in this scheme, there is no clue about how information about the utilised distribution is transferred to the utility, something that can also affect data confidentiality.

Another similar scheme is Elderberry [41]. In this scheme, smart meters are organised randomly into P2P groups, forming an aggregation tree with small groups of smart meters as leaf nodes. Each smart meter breaks a consumption reading into random numbers, that sum up to the actual reading, following the Slice-Mix-Aggregate algorithm [44], and sends these numbers, except one, to its group members. Then, each meter adds the number it kept, with the ones received by its neighbours; every meter in the group does the same, so that the final sum is the total consumption of the group. Finally, the root of the aggre-

gation tree sends this value to the utility. This scheme imposes low computation overhead to most smart meters, while medium overhead is posed to those smart meters that serve as aggregation tree roots, since they play the role of aggregators as well. There are some issues which are not discussed in the paper, like link encryption, billing and key management. Regarding data confidentiality, it is possible for colluding neighbours (even when link encryption is employed) to recover a metering. What is more, the authors argue that only by eavesdropping on all outgoing and all incoming communication links of a smart meter, the anonymization of this meter can be broken; this, however, can be possible in rural areas where smart meters will have only a few neighbours. While the use of TLS and client certificates are described in the paper, these are limited to the overlay bootstrapping, and are not compulsory for consumption messages transmission; since there seems to be no other measure taken to preserve message authentication, data integrity, and non-repudiation, these properties cannot be guaranteed. Also, the proposed scheme has potentially high complexity to construct the aggregation tree, leading to adaptability issues when nodes join or leave the smart grid.

In [42] four protocols for privacy-preserving aggregation are presented, where each party masks their measurements in such a way that, when these values are summed, masking values cancel each other out or add up to a known value so that the aggregator can obtain the total consumption. In this work the authors do not discuss the applicability of their method on billing, thus, it is questionable whether it can be supported or not. The proposed protocols present better efficiency compared to an homomorphic solution (in the paper they are compared to [4]), but they show increased complexity, affecting resilience during node join and leave. While the existence of signature keys is referenced, there are no details of a non-repudiation mechanism for the messages exchanged among smart grid nodes. The only protocol which is efficient, namely the *low-overhead protocol*, should be initialised with the public keys of all other smart meters in a group; this affects adaptability when meters join or leave the group, as well as support to arbitrary architectures.

While PARK [21] is described as using symmetric keys, it is essentially a masking scheme, since the "secret key" produced is simply added to the consumption. Smart meters also have in their possession digital certificates, and the result of the aforementioned addition is signed by the public key of the smart meter. The secret key is produced using two independent hash chains; one value from the first chain is concatenated to one value from the second chain, and the secret key is the hash of this concatenation. The seeds used for the hash chains are selected by the utility, and transmitted securely to the smart meters. An aggregator can sum the received values, and subtract from them the added secret keys; the secret keys are sent to the aggregator by the utility, which has knowledge of the respective hash chains. A drawback of this scheme is that it does not consider billing. Also, this method requires from each smart meter to keep a potentially large volume of data in its memory (depending on how long the hash chains are), because the two hash chains are of opposite direction; this is a significant drawback taking into account smart meters' hardware limitations. The computation cost will be medium for smart meters, since they need to perform 3 hash operations to create a single secret key in addition to the operations required for digitally sign the message. For aggregators, the overhead will also be medium, due to the many asymmetric decryptions needed to verify digital signatures of the messages received by smart meters. Finally, the utility overhead will be medium, due to the operations needed in order to produce the secret keys for all smart meters. The efficiency of this method is also affected by the additional communication between the aggregator and the utility: the utility sends the secret keys to the aggregator, which computes and sends back to the utility the real aggregated consumption, while it would be sufficient for the aggregator to send the masked aggregated consumption to the utility.

6. Comparative analysis of existing schemes with MASKER

We chose to present our proposal in the end, in order to be able to compare it with related work. The operation of MASKER has been presented in Section 3.2

in detail. Regarding the requirements presented in Section 2.3, our proposal meets all of them as shown in Figure 3 and analysed below.

MASKER preserves consumer privacy while at the same time supporting *intermediate aggregation*. An indicative architecture is the one presented in Section 2.1, which assumes that there are intermediate entities between the smart meters and the collector that aggregate consumption readings. The majority of existing works can also support this feature.

Apart from the reference architecture mentioned above, MASKER can support *arbitrary architectures* with multiple levels of intermediate aggregation and dynamic assignment of the aggregator role. There exist solutions whose operation depends on dedicated aggregators, while others perform aggregation on smart meters. MASKER can support both modes, operating in partial decentralisation, when it depends on dedicated aggregators, or full decentralisation, when specific smart meters are selected to play the role of aggregators. Similar to the last scenario, in-network aggregation is also supported, where intermediate smart meters aggregate received readings with theirs, and forward the result. In the related work presented in the previous section, approximately half of the solutions can support arbitrary architectures; the rest of them require specific formation of smart grid entities, like trees [7, 41] and predefined smart meter groups [4, 42].

MASKER can support *billing* with minor modifications, as shown in Section 3.4; moreover, it reuses high-frequency consumption readings in order to compute privacy-preserving low-frequency consumption readings without requiring from smart meters to send their readings again. It is worth noting here that more than half of the studied related works do not consider billing at all [7, 39, 41, 42, 21].

Consumption data are obfuscated with the addition of random values and *data confidentiality* is preserved against honest-but-curious nodes, which is the prevalent model used in the related literature. It should be noted here that if an aggregator does not aggregate but only forwards the received masked readings to the utility as is, then it is not considered honest-but-curious, since

it does not follow the protocol properly. Likewise, if the aggregation group is of size one (i.e., a single smart meter) the protocol is not considered an aggregation solution. For aggregation group sizes larger than one MASKER is secure, since the utility cannot distinguish individual consumptions from the aggregated sum. Compared to state-of-the-art, almost half of these solutions do not meet the confidentiality requirement adequately. MASKER is secure even when all other smart meters and the aggregator collude to attack the confidentiality of a single smart meter (which is a weakness of [4, 41] and [9]b). It is also robust against consumption pattern creation (the issue that [8] has), since consumption readings are obfuscated. Finally, data confidentiality is preserved even when the aggregator is malicious, since the latter cannot discover the real consumption values (in contrast to the first scheme in [9]).

Consumption data reports are digitally signed using the smart meter's private key, in order to ensure *message authentication*, *data integrity* and *non-repudiation*. Existing works that do not meet one or more of these requirements are [9, 41, 42], which do not define any specific protective measures for the transmission of the consumption values.

Regarding *efficiency*, our proposal keeps the communication overhead to the same level as a non privacy-preserving aggregation protocol, requiring no extra message exchanges in order to protect consumption privacy. The computation overhead of the smart meters is low, requiring an addition and two encryption operations in the worst case scenario. The fact that the overhead is low on smart meters, which are the most limited nodes in the smart grid, and relatively higher in aggregators and utility servers, which are expected to be more powerful, provides for an efficient solution for the smart grid, as shown in Section 4.2. Efficiency can be further improved by having some operations, like masking value generation, executed "off-line", when there is not so much load on the system servers. MASKER presents lower computation overhead compared to [7, 4, 39, 40], since they are based on homomorphic encryption operations that are costly [15, 16], whereas MASKER performs mainly arithmetic operations. Similarly, MASKER will present lower computation overhead

from schemes that are heavily based on asymmetric cryptography ([8] and [9]a), and potentially [9]b which does not describe the transmission channel security mechanism. Compared to PARK [21], which requires 3 hash operations to create a single secret key, MASKER needs 1 AES encryption for creating 8 masking values as demonstrated in the example of Section 3.2. According to [45], in the best case a hash function (SHA-1) is nine times faster than an AES cryptosystem (AES-256). For the previous example, in order to transmit consumption readings for 8 consecutive metering periods, MASKER needs 1 AES encryption, while PARK needs $3 \times 8 = 24$ hash operations; thus, in the long term MASKER will be around 2.5 times faster than PARK. MASKER does not impose extra message exchanges, like [4, 41] that break consumption into shares and send each one of them in a different message. Moreover, [21] requires the utility to send the sum of the masking values to the aggregators, resulting to extra message exchanges compared to MASKER. A comparison of MASKER with related schemes is presented in Table 4. Column "Setup" presents the complexity of the number of messages necessary to establish the keys to provide privacy. Column "Keys" shows the total number of keys that each meter uses for readings protection, not considering the keys needed for other operations, like digital signing, since not all schemes describe such mechanisms. Column "Messages" presents the total number of messages that are sent by round. The comparison shows that MASKER has the same or superior efficiency compared to similar methods.

Our method preserves the *accuracy* of the consumption data, since the utility can completely inverse the masking procedure and retrieve the exact aggregation measurements. In the second scheme described in [9], a very high number of smart meters is required in order to have high aggregated reading precision (the authors state that in an ideal case a group comprises around 3.8 million smart meters). Even if a high aggregation precision is not needed, a small group of smart meters would create data confidentiality issues. The rest of the schemes operate with accurate consumption readings.

Our protocol achieves *adaptability*, since the cost of reconfiguration when nodes join or leave the network is low; this is due to the fact that MASKER does

Table 4: Efficiency comparison with related work

Scheme	Setup	Keys	Messages
[7]	$O(N)$	1	$N + A$
[4]	TTP	n	$3 \times N + A$
[39]	TTP	1	$N + A$
[40]	$O(N)$	1	$2 \times N + A$
[8]	$O(N)$	2	$N + A$
[9]a	TTP	1	$N + A$
[9]b	$O(N)$	0	$N + A$
[41]	TTP	1	$(n \times (n-1) + n) \times A$
[42][4]	$O(N^2)$	n	$N + A$
[21]	TTP	1	$N + 2 \times A$
MASKER	TTP	1	$N + A$

[4] The *low overhead protocol* is considered here

Legend:

n	mean number of smart meters under a single aggregator
N	total number of smart meters
A	total number of aggregators

not form any kind of aggregation group. Related schemes are mainly divided into two categories: the first one includes [7, 41] where the smart meters must form an aggregation tree structure, while the second category comprises [4, 42] which require the smart meters to be organised into groups. These grouping requirements increase the reconfiguration cost on node join or leave and create adaptability issues.

Our solution shows high *scalability* and can support large numbers of devices, mainly due to the fact that aggregators perform aggregation through simple addition of values. The only method that shows poor scalability is [7] due to its conflicting design requirements: (a) the height of the aggregation tree should be small, while (b) tree nodes should not have too many children. As a result, for high numbers of nodes, these two requirements cannot hold both at the same time, thus, affecting efficiency.

MASKER does not include any kind of special entity that can be considered a *single point of failure*. Existing schemes that do not fulfil this requirement base the aggregation operation on a TTP ([8, 9]). Their main drawback is that,

if the TTP is successfully attacked, then it is much easier to mount a DoS attack to the smart grid. Although other schemes, including MASKER, require a TTP for key management, this is not considered a single point of failure because a distributed key management solution (like [17]) can effectivelly add the required redundancy to protect the smart grid against such attacks.

Finally, regarding *physical protection*, MASKER utilises a TEE not only for secure storage of cryptographic material (like secret keys) but also for executing critical operations (including generation of the masking values). As mentioned in Section 4, our implementation is based on the Open-TEE project [29], which is compliant with the GlobalPlatform TEE specifications, meaning that MASKER's trusted computing operations will be supported on any TEE that supports GlobalPlatform's specifications. The majority of the existing schemes do not offer any such protection; the only exception is [4], which assumes that the smart meters are equipped with some kind of trusted computing element that provides secure storage and basic cryptographic functionality.

7. Conclusion

This paper presents a novel, lightweight privacy-preserving aggregation protocol for the smart grid that, compared to state-of-the-art, proves to be more efficient and fulfils all functional and security requirements. First, we presented a reference architecture and a security model for privacy-preserving aggregation, and studied related requirements specifically for the smart grid domain. Then, we proposed MASKER, a protocol based on masking that requires mainly arithmetic and a few cryptographic operations. MASKER places more overhead on the more powerful nodes of the smart grid (i.e. intermediate aggregators and utility servers), while leaving the smart meters with minimal computation cost. Regarding MASKER's security, we proved that it is not possible for an attacker to infer consumption values just by observing the transmitted masked readings. The main advantage of MASKER over similar solutions, as detailed in Section 6, is that it is the only solution that complies to all requirements set

for privacy-preserving aggregation schemes. The comparison and security evaluation showed that MASKER is a secure, efficient and flexible protocol that preserves customers' privacy.

Acknowledgement

This research has been funded by the European Commission as part of the SMART-NRG project (FP7-PEOPLE-2013-IAPP Grant number 612294).

References

[1] Smart grids and meters, http://ec.europa.eu/energy/en/topics/markets-and-consumers/smart-grids-and-meters, accessed 27-10-2015.

[2] T. E. foundation, Utility-scale smart meter deployments: Building block of the evolving power grid, Tech. rep., The Edison foundation (September 2014).

[3] U. Greveler, B. Justus, D. Loehr, Multimedia content identification through smart meter power usage profiles, in: in Computers, Privacy and Data Protection (CPDP, 2012.

[4] F. D. Garcia, B. Jacobs, Privacy-friendly energy-metering via homomorphic encryption, in: Security and Trust Management, Springer, 2011, pp. 226–238.

[5] B. Karimi, V. Namboodiri, M. Jadliwala, Scalable meter data collection in smart grids through message concatenation, IEEE Transactions on Smart Grid 6 (4) (2015) 1697–1706.

[6] T. Shiobara, P. PalenskyÍŃ, H. NishiÍŁ, Effective metering data aggregation for smart grid communication infrastructure, in: Industrial Electronics Society, IECON 2015-41st Annual Conference of the IEEE, IEEE, 2015, pp. 002136–002141.

[7] F. Li, B. Luo, P. Liu, Secure information aggregation for smart grids using homomorphic encryption, in: Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on, IEEE, 2010, pp. 327–332.

[8] C. Efthymiou, G. Kalogridis, Smart grid privacy via anonymization of smart metering data, in: Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on, IEEE, 2010, pp. 238–243.

[9] J.-M. Bohli, C. Sorge, O. Ugus, A privacy model for smart metering, in: Communications Workshops (ICC), 2010 IEEE International Conference on, IEEE, 2010, pp. 1–5.

[10] A. J. Paverd, A. P. Martin, Hardware security for device authentication in the smart grid, in: Smart Grid Security, Springer, 2012, pp. 72–84.

[11] M. Sabt, M. Achemlal, A. Bouabdallah, Trusted execution environment: what it is, and what it is not, in: Trustcom/BigDataSE/ISPA, 2015 IEEE, Vol. 1, IEEE, 2015, pp. 57–64.

[12] J.-E. Ekberg, K. Kostiainen, N. Asokan, The untapped potential of trusted execution environments on mobile devices, IEEE Security & Privacy 12 (4) (2014) 29–37. doi:10.1109/msp.2014.38.
URL http://dx.doi.org/10.1109/MSP.2014.38

[13] B. Krebs, FBI: Smart meter hacks likely to spread, last accessed 27-09-2016 (2012).
URL http://krebsonsecurity.com/2012/04/fbi-smart-meter-hacks-likely-to-spread/

[14] S. Finster, I. Baumgart, Privacy-aware smart metering: A survey, IEEE COMMUNICATION SURVEYS & TUTORIALS 17 (2) (2015) 1088–1101.

[15] M. Naehrig, K. Lauter, V. Vaikuntanathan, Can homomorphic encryption be practical?, in: Proceedings of the 3rd ACM workshop on Cloud computing security workshop, ACM, 2011, pp. 113–124.

[16] J. Liu, Y.-H. Lu, C.-K. Koh, Performance analysis of arithmetic operations in homomorphic encryption, Tech. rep., Purdue University (2010).
URL http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1403&context=ecetr

[17] F. F. Demertzis, G. Karopoulos, C. Xenakis, A. Colarieti, Self-organised key management for the smart grid, in: International Conference on Ad-Hoc Networks and Wireless, Springer, 2015, pp. 303–316.

[18] OMTP Limited, Advance Trusted TR1 Specifications v1.1, last accessed 4-10-2016 (May 2009).
URL http://www.gsma.com/newsroom/wp-content/uploads/2012/03/omtpadvancedtrustedenvironmentomtptr1v11.pdf

[19] E. Barker, J. Kelsey, NIST special publication 800-90A: Recommendation for random number generation using deterministic random bit generators.
URL http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf

[20] E. Barker, Special publication 800-57 part 1 revision 4, recommendation for key management, part 1: General, Tech. rep., NIST (2016).

[21] K. Zhang, R. Lu, X. Liang, J. Qiao, X. Shen, PARK: A privacy-preserving aggregation scheme with adaptive key management for smart grid, in: Communications in China (ICCC), 2013 IEEE/CIC International Conference on, 2013, pp. 236–241. doi:10.1109/ICCChina.2013.6671121.

[22] R. Agrawal, R. Srikant, Privacy-preserving data mining, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD '00, ACM, New York, NY, USA, 2000, pp. 439–450. doi:10.1145/342009.335438.
URL http://doi.acm.org/10.1145/342009.335438

[23] C. Dwork, A. Roth, et al., The algorithmic foundations of differential pri-

vacy, Foundations and Trends in Theoretical Computer Science 9 (3-4) (2014) 211–407.

[24] M. Savi, C. Rottondi, G. Verticale, Evaluation of the precision-privacy tradeoff of data perturbation for smart metering, IEEE Transactions on Smart Grid 6 (5) (2015) 2409–2416.

[25] Https://www.entsoe.eu/data/data-portal/consumption/Pages/default.aspx, last accessed 4-4-2016.

[26] Https://tls.mbed.org, last accessed 14-12-2015.

[27] C. E. Shannon, Communication theory of secrecy systems, Bell system technical journal 28 (4) (1949) 656–715.

[28] D. R. Stinson, Cryptography: theory and practice, CRC press, 2005.

[29] B. McGillion, T. Dettenborn, T. Nyman, N. Asokan, Open-TEE – an open virtual trusted execution environment, in: Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA, TRUSTCOM '15, IEEE Computer Society, Washington, DC, USA, 2015, pp. 400–407. doi:10.1109/Trustcom.2015.400.

[30] GlobalPlatform, TEE System Architecture, version 1.0, last accessed 4-10-2016 (December 2011).
URL `https://www.globalplatform.org/specificationsdevice.asp`

[31] ARM, ARM security technology - Building a secure system using TrustZone technology, last accessed 4-10-2016 (April 2009).

[32] D. Whiting, R. Housley, N. Ferguson, Counter with CBC-MAC (CCM), RFC 3610 (Informational) (Sep. 2003).
URL `http://www.ietf.org/rfc/rfc3610.txt`

[33] X. Liu, Y. Zhang, B. Wang, H. Wang, An anonymous data aggregation scheme for smart grid systems, Security and Communication Networks 7 (3) (2014) 602–610.

[34] S. Cho, H. Li, B. J. Choi, Palda: Efficient privacy-preserving authentication for lossless data aggregation in smart grids, in: Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on, IEEE, 2014, pp. 914–919.

[35] B. Defend, K. Kursawe, Implementation of privacy-friendly aggregation for the smart grid, in: Proceedings of the first ACM workshop on Smart energy grid security, ACM, 2013, pp. 65–74.

[36] S. L. Keoh, Y. H. Ang, Z. Tang, A lightweight privacy-preserved spatial and temporal aggregation of energy data, in: 2015 11th International Conference on Information Assurance and Security (IAS), 2015, pp. 1–6. doi:10.1109/ISIAS.2015.7492762.

[37] Z. Erkin, G. Tsudik, Private computation of spatial and temporal power consumption with smart meters, in: Applied Cryptography and Network Security, Springer Science & Business Media, 2012, pp. 561–577. doi:10.1007/978-3-642-31284-7_33.
URL http://dx.doi.org/10.1007/978-3-642-31284-7_33

[38] F. Knirsch, G. Eibl, D. Engel, Error-resilient masking approaches for privacy preserving data aggregation, IEEE Transactions on Smart Grid.

[39] R. Lu, X. Liang, X. Li, X. Lin, X. Shen, EPPA: An efficient and privacy-preserving aggregation scheme for secure smart grid communications, Parallel and Distributed Systems, IEEE Transactions on 23 (9) (2012) 1621–1631.

[40] F. G. Marmol, C. Sorge, O. Ugus, G. M. Pérez, Do not snoop my habits: preserving privacy in the smart grid, Communications Magazine, IEEE 50 (5) (2012) 166–172.

[41] S. Finster, I. Baumgart, Elderberry: A peer-to-peer, privacy-aware smart metering protocol, in: INFOCOM, 2013 Proceedings IEEE, IEEE, 2013, pp. 3411–3416.

[42] K. Kursawe, G. Danezis, M. Kohlweiss, Privacy-friendly aggregation for the smart-grid, in: Privacy Enhancing Technologies, Springer, 2011, pp. 175–191.

[43] M. Jawurek, M. Johns, K. Rieck, Smart metering de-pseudonymization, in: Proceedings of the 27th Annual Computer Security Applications Conference, ACM, 2011, pp. 227–236.

[44] W. He, X. Liu, H. Nguyen, K. Nahrstedt, T. Abdelzaher, PDA: Privacy-preserving data aggregation in wireless sensor networks, in: INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, IEEE, 2007, pp. 2045–2053.

[45] H. Rifa-Pous, J. Herrera-Joancomartí, Computational and energy costs of cryptographic algorithms on handheld devices, Future internet 3 (1) (2011) 31–48.