Despicable Me(ter): Anonymous and Fine-grained Metering Data Reporting with Dishonest Meters

Moreno Ambrosin*, Hossein Hosseini†, Kalikinkar Mandal†, Mauro Conti*, Radha Poovendran†
* University of Padua, Padua, Italy. Email: {ambrosin,conti}@math.unipd.it
† University of Washington, Seattle, WA, USA. Email: {hosseinh,kmandal,rp3}@uw.edu

***Abstract*—The Advanced Metering Infrastructure (AMI) is a fundamental component of modern Smart Grids, and allows fine-grained and real-time monitoring of the electricity consumption of utility customers. In an AMI, intelligent devices commonly called Smart Meters (SMs) communicate with an operation center for the purpose of management and billing. However, while on one hand this technology has the potential for advanced load balancing and grid management, it poses a threat to customers privacy. Indeed, an adversary can infer sensitive information about the end users by analyzing the metering data reported by the SMs. In this paper, we present the design of a privacy-preserving AMI for fine-grained metering data collection. We propose a collaborative protocol among SMs that achieves anonymous metering data delivery via a random multi-hop path. Our construction enables a verifier entity to detect any inconsistent behavior from SMs by accessing their internal log. Our scheme is scalable with the number of SMs in the network, and unlike existing methods, does not rely on trusted third-parties. We consider an adversarial setting where SMs are either honest-but-curious or controlled by a powerful adversary, whose aim is to deanonymize the received metering data. Finally, we prove that our protocol is secure and computationally efficient for the resource-constrained SM devices.**

***Index Terms*—Smart Grid; Smart Metering; Advanced Metering Infrastructure; Security; Privacy; Anonymity.**

I. Introduction

The advent of Smart Grid technologies has the potential to bring an advanced control over both energy generation and distribution in energy networks. A fundamental component of modern Smart Grids is the Advanced Metering Infrastructure (AMI), which intends to periodically communicate with intelligent metering devices, namely Smart Meters (SMs), to collect and analyze the energy usage of private houses or buildings. SMs periodically transmit usage data to a Metering Data Management System (MDMS), which collects fine-grained time series of metering data, for management purposes, or aggregates energy consumption data for billing [1]. The adoption of AMIs is supported and encouraged by governments; for example, in United States in 2013, about 50 million SMs were registered operating in AMI mode [2], and in United Kingdom up to March 2015, about one million SMs are operating in AMI mode [3].

This work was supported by NSF grant CNS-1544173. This work is also partially supported by the EU TagItSmart! Project (agreement H2020-ICT30-2015-688061), the EU-India REACH Project (agreement ICI+/2014/342-896), and the project "Content Centric Networking: Security and Privacy Issues" funded by the University of Padua. Mauro Conti is supported by a Marie Curie Fellowship funded by the European Commission (agreement PCIG11-GA-2012-321980).

Motivation. The privacy implications of AMI adoption have been highlighted in several works in the literature [4]–[7]. It has been shown that the use of AMI and in particular the collection of fine-grained consumption measurements from SMs reduces the privacy of customers. Indeed, privacy-sensitive information, such as the detailed user activities, can be inferred from the raw metering data [6], [7]. Therefore, it is vital to deploy mechanisms for preserving the privacy of customers, against both the MDMS and external attackers, without compromising the utility of the data.

One common approach is to aggregate frequent metering data at the edge of the AMI network and deliver the result to the MDMS [8]–[12]. However, compared to having the individual measurements, aggregation significantly reduces the utility of the data, restricting the type of the statistics that can be extracted from them. Another solution for providing strong privacy guarantees is using trusted or semi-trusted third-party anonymizers [1], [13]. This solution however might be expensive and difficult to deploy and manage. Also, a privacy preserving protocol for billing purposes has been proposed in [14], which, however, requires the intervention of the customer in the computations.

Contribution. In this paper, we present the design of a privacy-preserving smart metering, based on our preliminary design presented in [15]. Our protocol allows SMs to anonymously transmit individual fine-grained metering data to the MDMS, thus maintaining the utility of the data. We consider an honest-but-curious model for SMs and MDMS, as well as an attacker who have access to the raw metering data and can take control of a portion of the SMs in the network. We propose a collaborative protocol among SMs to guarantee the anonymity of the reporting process for the utility customers. SMs securely log a minimal set of their activities, such that a trusted Verification Center (VC), who periodically analyzes the logs, can verify the functionality of SMs. Additionally, our solution guarantees authenticity and integrity of the metering data received by the MDMS. Our work considers realistic system model and assumptions and aims at providing a practical and easily deployable solution, with minimal modifications to the existing infrastructure.

Organization. The rest of the paper is organized as follows. Section II presents our system model, security assumptions, and the reference attacker model. Section III describes our protocol for anonymous metering data reporting. In Section V, we analyze the security of the proposed protocol, and, in Section VI, we evaluate its implementation aspects. An overview of the existing literature is presented in Section VII, and Section VIII concludes the paper.

II. Reference Model and Assumptions

In this section, we present our reference system model (Section II-A), our design and security goals (Section II-B), and our security model (Section II-C).

A. System Model

We intend to develop a privacy-preserving protocol for fine-grained smart metering in AMI. We propose a realistic system model, presented in Figure 1, which is similar to the previous works [16]–[18]. In the proposed infrastructure, Smart Meters (SMs) periodically (e.g., every 10 minutes) transmit *reports*, containing timestamped metering data, to a Metering Data Management System (MDMS). MDMS is one of the units of the Operation Center. Other units are Electricity Utility Provider and Load Monitoring Center, who queries MDMS to compute statistics on metering data.

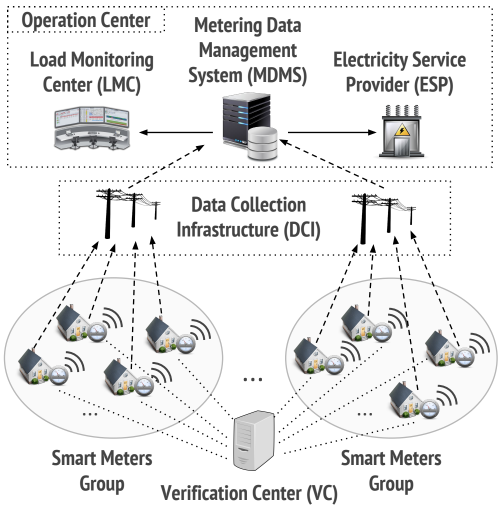
Operation Center
Metering Data Management System (MDMS)
Load Monitoring Center (LMC)
Electricity Service Provider (ESP)
Data Collection Infrastructure (DCI)
...
...
...
Smart Meters Group
Verification Center (VC)
Smart Meters Group

Fig. 1. System Model.

We consider the AMI to be a mesh network, where SMs are capable of both wireless connectivity, for communication with peer SMs, and wired connectivity, for delivering data to the MDMS. SMs are organized in separate *groups*, within which each SM is in the wireless communication coverage of its neighborhood. We assume that SMs use Power Line Communications (PLC) to deliver the metering data to the MDMS, as leveraging the existing electrical infrastructure reduces the costs of deployment and maintenance [19], [20]. Thus, SMs are assumed to be connected to a Data Collection Infrastructure (DCI), which relays the data to the MDMS. The DCI consists of multiple collector units geographically distributed in the network, which naturally divide the set of SMs in groups, for a more efficient data collection.

Finally, we assume the presence of a Verification Center (VC), which is a trusted third party responsible for verifying the behavior of SMs. VC *does not* actively participate in the metering data collection; instead, it periodically (e.g., once a month) monitors the system status to detect any inconsistent behavior. VC can be, for example, a government agency.

SMs capabilities and security assumptions. We suppose SMs are *tamper resistant* devices, i.e., they cannot be physically attacked (i.e., accessed or compromised) by an adversary [13], [14]. However, SMs can be subject to software attacks: an attacker can modify or replace the software on SMs and change their expected behavior. SMs are capable of basic cryptographic operations, such as hashing and symmetric/public key encryption [21]. Moreover, similar to [17], and in line with the design of commercial SM devices [22] and recent advances in embedded systems security [23], we assume SMs have a low-cost, trusted component (e.g., a Trusted Platform Module (TPM) chip [24]).

B. Design and Security Goals

We consider the following security goals for our protocol.

Confidentiality. The protocol should guarantee the confidentiality of the raw metering data that SMs deliver to MDMS; unauthorized entities should not be able to access such data.

Authenticity and Integrity. MDMS should be able to authenticate the data, i.e., assess that the data has been originated by a legitimate SM, and verify its integrity.

Anonymity. Of the different types of the anonymity [25], we intend to provide the unlinkability of the metering data and the source SM, i.e., no entity, including the MDMS, other SMs, or an external attacker, should be able to link the metering data with the identity of the source SM.

Verifiability. VC should be able to verify that all SMs are honestly and correctly following the protocol.

Additionally, we require our solution to have *Low Complexity*. Specifically, we intend the protocol to require minimal modifications to the existing infrastructure, to have low computational overhead and also easy deployment. Moreover, the solution also needs to be scalable with the number of SMs.

C. Security Model

We consider a security model where no entity is absolutely trusted. The details of the model are described in the following.

Internal Attacker. Similar to [26], we assume the presence of an internal attacker who is part of the operation center and has access to the raw metering data provided by MDMS. Moreover, we add the assumption that the internal attacker has the ability to compromise (the application software of) part of the deployed SMs, in order to gain additional information from SMs in a group and deanonymize users. Compromised SMs may try to inject arbitrary packets into the network, and access and modify the content of their internal log files. However, while the internal attacker intends to deanonymize users, the metering data are needed for operational purposes. Therefore, compromised SMs have no incentive in mounting Denial of Service (DoS) attacks, e.g., dropping packets.

MDMS. The data collection entity MDMS is honest-but-curious, i.e., it may analyze the received measurements to link them to the users' identities.

External Attacker. External attacker targets the following:

- Functionality of the protocol, by sending fake metering data to the MDMS and/or injecting fake messages into the network.
- Confidentiality and integrity of the data, by eavesdropping the communications and altering the content of the reports.
- Anonymity of the SMs, by trying to link a message with the source SM.

Non-malicious SMs. They are assumed to be honest-but-curious, i.e., they honestly follow the reporting protocol, but are curious to find a link between a message and its initiator.

III. Our Solution

In this section, we present our scheme for anonymous metering data reporting.

We use S to indicate a source SM (i.e., a SM that initiates a message transmission), and I to indicate an intermediate SM in a multi-hop path. Each source SM S assigns a permanent ID, $C_{S,0}, \ldots, C_{S,N_S-1}$, to each of the N_S reachable SMs, i.e., the SMs in its wireless coverage. The assignments are known to the VC. Each SM also shares a symmetric key with its neighboring SMs. Such key can be exchanged at bootstrap using well-known schemes, such as Diffie-Hellman key exchange. SMs are assumed to be loosely synchronized with MDMS. We divide the time into intervals of T seconds, and represent the current time-interval as $\tilde{t} = \lfloor \frac{t}{T} \rfloor$, where $\lfloor \cdot \rfloor$ is the floor operation. Table I summarizes the notation we use in the paper.

We also assume that there is a log file in each SM's memory. The log is not inside the protected area of the memory, but is stored such that: (1) it contains sufficient information for VC to verify the behavior of SMs, and (2) any modification to the log can be detected by VC. Section IV-A describes a construction for the secure log. We assume SMs can append an entry to their log using the LOG_STORE() function.

A. Anonymous Verifiable Metering Data Reporting

At every interval $\tilde{t}$, each SM transmits its metering data to MDMS through a random multi-hop path composed by other SMs in its group. At the beginning of each time interval $\tilde{t}$, each SM generates a random nonce as the hash of the previous nonce $\nu_{\tilde{t}} = H(\nu_{\tilde{t}-1})$. It also generates a random string X, and starts the reporting process by transmitting the metering data to its j-th neighbor, where $j = X_{(2)} (\text{mod } N_S)$. The source SM includes X in the message transmitted to the first hop, which will be used to derive the id of the next hop. Note that the message must traverse at least one hop before being delivered to the MDMS.

Upon receiving the message, the intermediate SM generates a random string Y as the hash of the concatenation of the received random string X with its current nonce $\nu_{\tilde{t}}$. Based on Y, it then decides to either deliver the received packet to MDMS or to forwards it to another peer SM. Both the source and intermediate SMs store a minimal information inside their log file, respectively $\mathbf{0}\|X\|t$ and $\nu_{\tilde{t}}\|Y\|t$. By looking into the log files, VC can reconstruct the paths for each message. Thus, the forwarding mechanism is both random, since it relies on the randomness of a cryptographic hash function, and verifiable by a third-party entity in possess of the log files of all SMs.

TABLE I
NOTATION

Entities	
S, I	Source and intermediate SM in a path, respectively.
$C_{(\cdot),j}$	The j-th SM in the neighborhood of a SM.
Parameters	
$N_{(\cdot)}$	Number of reachable meters for a SM.
$\tilde{t} = \lfloor \frac{t}{T} \rfloor$	Quantized version of the current time t based on the interval duration T.
$d_{S,\tilde{t}}$	Metering data of S at $\tilde{t}$.
$\nu_{\tilde{t}}$	Random nonce of each SM at time $\tilde{t}$.
k_G	Group key shared between a group G of SMs and MDMS.
k_j	Symmetric key of a SM shared with its j-th neighbor.
mk	The secret key of a SM shared with the VC.
pk, sk	Public and secret key pair of MDMS.
h	An positive integer, where $\frac{1}{h}$ is the probability that an intermediate SM delivers a received packet to MDMS.
$X^{(1)}, X_{(2)}$	The decimal value of the first byte/last two bytes of a bit string X.
ID$(\cdot)$	Unique identifier of each SM.
Cryptographic Primitives	
$\mathrm{E}_k(\cdot)$, $\mathrm{D}_k(\cdot)$	Symmetric Encryption and Decryption with key k.
$\mathrm{E}_{pk}(\cdot)$	Public key Encryption with key pk.
$\mathrm{H}(\cdot)$	Cryptographic Hash function.
$\mathrm{HMAC}_k(\cdot)$	Hash-based Message Authentication Code with key k.

In the following, we provide the details of our protocol for anonymous reporting of metering data.

Initialization. Algorithm 1 describes the initialization phase. First, the source SM, S, uses the MDMS's public key to compute M as the encryption of the concatenation of the metering data $d_{S,\tilde{t}}$, the corresponding time-interval $\tilde{t}$, the group ID G, and the HMAC of the metering data with the group key $\mathrm{HMAC}_{k_G}(d_{S,\tilde{t}})$ (line 1 of Algorithm 1). Then, it generates a random bit string $X = H(\mathbf{0}\|\nu_{\tilde{t}})$, where $\mathbf{0}$ is an all-zero string with the same bit-length as X, and computes $j \leftarrow X_{(2)} (\text{mod } N_S)$ (lines 2-3 of Algorithm 1). Finally, S computes $\phi = \mathrm{HMAC}_{k_G}(\tilde{t}\|X)$, creates the message by encrypting the string $(M\|\tilde{t}\|X\|\phi)$ with the symmetric key shared with the $C_{S,j}$, the j-th SM in its neighborhood, and transmits it to $C_{S,j}$ (lines 4-5 of Algorithm 1). SM S also adds a new entry $(\mathbf{0}\|X\|t)$ into its log file by invoking the function LOG_STORE() (line 6 of Algorithm 1).

Forwarding. Algorithm 2 describes the forwarding phase. When an intermediate SM, I, receives a packet, it first decrypts it with the symmetric key k_{prev} shared with the previous SM in the path, to obtain $(M\|\tilde{t}\|X\|\phi)$ (line 1 of Algorithm 2). It then

Algorithm 1 Initialization: S sends its metering data to a first hop.

```
Input: Metering data $d_{S,\tilde{t}}$ at time $\tilde{t}$; MDMS's public key $pk_{\text{MDMS}}$; Group key $k_G$; Group ID $G$; Random nonce $\nu_{\tilde{t}}$ at time $\tilde{t}$; ID and symmetric key of the $N_S$ reachable SMs; Integer $h$.
1: $M \leftarrow E_{pk}(d_{S,\tilde{t}} \parallel \tilde{t} \parallel G \parallel \text{HMAC}_{k_G}(d_{S,\tilde{t}}))$;
2: $X \leftarrow H(\mathbf{0} \| \nu_{\tilde{t}})$;
3: $j \leftarrow X_{(2)} (\text{mod } N_S)$;
4: $PKT \leftarrow E_{k_j}(M \parallel \tilde{t} \parallel X \parallel \text{HMAC}_{k_G}(\tilde{t} \parallel X))$;
5: Transmit $PKT$ to $C_{S,j}$;
6: LOG_STORE($\mathbf{0} \parallel X \parallel t$);
```

checks the following conditions: (1) $\tilde{t}$ is not the current or the previous time-interval, (2) X has been already received at the current or previous time-intervals, or (3) $\text{HMAC}_{k_G}(\tilde{t} \| X) \neq \phi$. If any of these conditions hold, it drops the packet and stops the procedure (lines 2-4 of Algorithm 2); otherwise, it continues the procedure by storing X in a local memory, which will be erased after two time-intervals (line 5 of Algorithm 2). Finally, the intermediate SM computes $Y = \text{H}(X \| \nu_{\tilde{t}})$ and obtains $c = Y^{(1)} (\text{mod } h)$ and $j = Y_{(2)} (\text{mod } N_I)$ (lines 6-7 of Algorithm 2). If $c = 0$, which occurs with probability $\frac{1}{h}$, it delivers M to MDMS (lines 8-9 of Algorithm 2); otherwise, it computes $\phi = \text{HMAC}_{k_G}(\tilde{t} \| Y)$, creates PKT by encrypting $(M \| \tilde{t} \| Y \| \phi)$ with the symmetric key shared with $C_{I,j}$, the j-th SM in its neighborhood, and forwards PKT to $C_{I,j}$ (lines 10-13 of Algorithm 2). Similar to the Initialization, the intermediate SM also adds a new entry $(\nu_{\tilde{t}} \| Y \| t)$ into its log file by invoking the function LOG_STORE() (line 14 of Algorithm 2).

Algorithm 2 Forwarding: I performs one of the following operations: (1) forwards the received packet to another SM; or (2) delivers it to the MDMS; (3) or discards the packet if it is not verified.

```
Input: Received packet $PKT$; Symmetric key $k_{prev}$ shared with the previous SM; Group key $k_G$; Random nonce $\nu_{\tilde{t}}$ at time $\tilde{t}$; ID and symmetric key of the $N_I$ reachable meters; Integer $h$; Local memory.
1: $M \parallel \tilde{t'} \parallel X \parallel \phi \leftarrow D_{k_{prev}}(PKT)$;
2: if $(\tilde{t'} \neq \tilde{t} \wedge \tilde{t'} \neq \tilde{t} - 1) \vee (X$ already received during $\tilde{t}$ or $\tilde{t} - 1) \vee (\text{HMAC}_{k_G}(\tilde{t} \parallel X) \neq \phi)$ then
3:     Discard the packet and stop;
4: end if
5: Store $X$ in local memory for two time-intervals;
6: $Y \leftarrow \text{H}(X \parallel \nu_{\tilde{t}})$;
7: $c \leftarrow Y^{(1)} (\text{mod } h)$; $j \leftarrow Y_{(2)} (\text{mod } N_I)$;
8: if $c = 0$ then
9:     Deliver $M$ to MDMS;
10: else
11:     $PKT \leftarrow E_{k_j}(M \parallel \tilde{t} \parallel Y \parallel \text{HMAC}_{k_G}(\tilde{t} \parallel Y))$;
12:     Forward $PKT$ to $C_{I,j}$;
13: end if
14: LOG_STORE($\nu_{\tilde{t}} \parallel Y \parallel t$);
```

For the description of the protocol, we assumed that $h \leq 256$, i.e., the probability of forwarding a received message to the MDMS is greater than or equal to 2^{-8}. Also, for each SM, the number of neighboring SMs are assumed to be less than or equal to 2^{16}. Although, these are practical parameters for a typical system, they can be adjusted to any desired values.

Transmission Times. SMs stack the outgoing messages and transmit them at once, to their respective receivers, when the number of stacked messages reaches a pre-determined number, which is also known to the VC. Note that SMs record the true transmission time in the log, but include the quantized version $\tilde{t}$ of the time interval in the transmitted message.

Receiving data in MDMS. Upon receiving the message M, MDMS first extracts $(d_{S,\tilde{t}} \| \tilde{t} \| G \| \phi)$ by decrypting M with its secret key sk. If $\phi = \text{HMAC}_{k_G}(d_{S,\tilde{t}})$, MDMS verifies both the authenticity and the integrity of the metering data; otherwise, it discards the packet.

B. Verification of SMs behavior

Our scheme allows VC to verify the correct functionality of SMs by accessing their internal log file. Assume that SMs deliver their logs to VC via a secure channel; this can be done, for example, by encrypting the log with the VC's public key, and delivering it via PLC (as for reporting to MDMS) or over the Internet. The verification process consists of two steps: *log integrity verification*, which ensures that the log file has not been modified, and *reporting protocol verification*, which verifies the correct execution of the protocol by SMs. Section IV describes a secure log construction, as well as how VC can verify the log's integrity. In the following, we provide a description of reporting protocol verification.

Reporting Protocol Verification. After verifying the integrity of the log files, VC checks if SMs have followed the protocol correctly. Algorithm 3 presents the verification process performed by VC, which outputs a set E of IDs of malfunctioning, i.e., erroneous or compromised, SMs. The set E is initialized as empty (line 1 of Algorithm 3). For each source SM S, VC selects the entry of the log file corresponding to the message originated at time $\tilde{t}$, of the form $e = \mathbf{0} \| X \| t$, where $\tilde{t} = \lfloor \frac{t}{T} \rfloor$. If no such entry exists, the id of S is added to E (lines 2-6 of Algorithm 3). If e exists, VC iteratively reconstructs the forwarding chain as follows. It keeps track of the current SM $Curr$, initially S, and of the next SM $Next$, initially $C_{S,j}$ (line 7 of Algorithm 3). It then searches $Next$'s log for an entry $e' = \nu_{\tilde{t}} \| Y \| t$, where $Y = \text{H}(X \| \nu_{\tilde{t}})$ and $\tilde{t} = \lfloor \frac{t}{T} \rfloor$ and $\tilde{t} = \lfloor \frac{t}{T} \rfloor + 1$. If e' does not exist, VC adds the id of $Next$ to the set of the potentially compromised SMs and exits the loop (lines 9-14 of Algorithm 3). Otherwhise, VC computes $c \leftarrow Y^{(1)} (\text{ mod } h)$ (line 15 of Algorithm 3). If $c = 0$, it means that $Next$ must have delivered the message to MDMS; thus, VC exists the loop (lines 16-18 of Algorithm 3); if $c \neq 0$, VC computes $j \leftarrow Y_{(2)} (\text{mod } N_{Curr})$, sets $Curr \leftarrow Next$, $Next \leftarrow C_{Curr,j}$ (line 19 of Algorithm 3), and repeats the loop (from line 8 of Algorithm 3). The algorithm verifies all the SMs in a group G with the complexity $O(|G| \cdot h)$, where $h \ll |G|$.

IV. SECURE LOG IMPLEMENTATION

As introduced in Section III, we assume that there is a secure log, i.e., a log file stored inside an unprotected area of each SM's memory, to which modifications can be detected by VC. In the following, we present the design of a possible secure log implementation that leverages a TPM v1.2 secure cryptoprocessor [27]. TPM v1.2 provides secure generation and storage of keys, pseudo-random number generation, and secure storage of a limited amount of data.

Algorithm 3 Log verification.

```
Input: Time $\tilde{t}$; Log files of all SMs.
Output: Set of malfunctioning (erroneous or compromised) SMs $E$.
 1: $E \leftarrow \varnothing$
 2: for Source SM $S$ do
 3:     $e \leftarrow S$'s log entry $e = (\mathbf{0} \parallel X \parallel t)$, for $t$ which $\tilde{t} = \lfloor \frac{t}{T} \rfloor$;
 4:     if $\nexists\ e$ then
 5:         $E \leftarrow E \cup \{\mathrm{ID}(S)\}$
 6:     else
 7:         $Curr \leftarrow S$; $j \leftarrow X_{(2)} (\bmod N_{Curr})$; $Next \leftarrow C_{S,j}$;
 8:         loop
 9:             $Y \leftarrow \mathrm{H}(X \parallel \nu_{\tilde{t}})$;
10:             $e' \leftarrow$ $Next$'s log entry $e' = (\nu_{\tilde{t}} \parallel Y \parallel t)$, for $t$ which $\tilde{t} = \lfloor \frac{t}{T} \rfloor$ or $\tilde{t} = \lfloor \frac{t}{T} \rfloor + 1$;
11:             if $\nexists\ e'$ then
12:                 $E \leftarrow E \cup \{\mathrm{ID}(Next)\}$;
13:                 exit loop;
14:             end if
15:             $c \leftarrow Y^{(1)} (\bmod h)$;
16:             if $c = 0$ then
17:                 exit loop;
18:             end if
19:             $Curr \leftarrow Next$; $j \leftarrow Y_{(2)} (\bmod N_{Curr})$; $Next \leftarrow C_{Next,j}$;
20:         end loop
21:     end if
22: end for
```

A. Secure Log Construction

We assume VC shares a distinct secret mk with each SM. Such secret is stored in a protected memory inside the TPM (i.e., TPM's NVRAM [27]). A TPM v1.2 contains up to 24 Platform Configuration Registers (PCRs), which are 160-bit (20-byte) registers inside the TPM. The value of a PCR can be read and modified only through, respectively, the functions TPM _PCR_READ() and TPM _PCR_EXTEND(), exposed by the TPM's driver [28]. TPM _PCR_EXTEND() takes as input a sequence of bytes of arbitrary size, and a PCR index i; it concatenates the sequence of bytes in input with the current value of the i-th PCR, and computes its SHA-1() hash. Then, it writes the result inside the i-th PCR (overwriting its current value). Therefore, by construction, when a PCR transits from a state to the next one, it is impossible to return back to the previous state.

We leverage this functionality to construct our secure log. At boot time, each SM initializes a specific PCR of its TPM, using mk as input of the function TPM _PCR_EXTEND(). A SM adds a new log's entry e_l using the function LOG_STORE() (see Figure 2). This function first invokes TPM _PCR_EXTEND(), by passing the content of the new entry as input, which updates the value of the reference PCR (Operation 1 in Figure 2). Then, SM reads the new extended PCR value (using TPM _PCR_READ()), and concatenates it with the content of the new log's entry (Operation 2 in Figure 2).

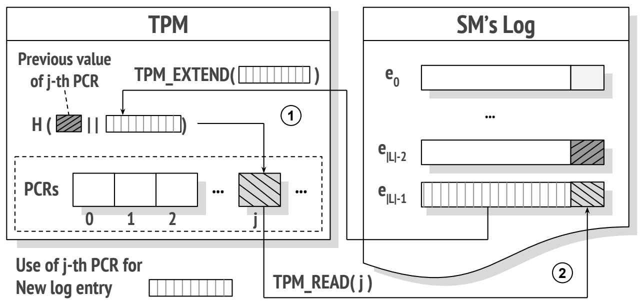

Fig. 2. Operation performed by LOG_STORE() function.

B. Log Integrity Verification

VC knows the value mk and has access to the log file and a "quote", which is a signed value with a TPM's unique key pair of the current value of the PCR (obtained with TPM _PCR_QUOTE() function). Thus, VC can easily reconstruct the chain of hash values appended to each log's entry, verifying the integrity of the log file. Note that this alone is not sufficient for VC to verify the behavior of SMs.

Let ψ_0 be the initial value of PCR$_l$, which VC computes as the hash of the default value of PCR$_l$ (e.g., 20 zero bytes) concatenated with mk. Let e_j be the jth entry of SM's log L, such that $e_j = \{content\} \| \psi_j$. VC first checks whether $\psi_{|L|}$ is equal to the received quote; if this is true, it then simply checks if $\mathrm{H}(\psi_{j-1} \parallel \{content\}) = \psi_j,\ 1 \leq j \leq |L|$, where $|L|$ is the number of entries of L. The integrity verification has the complexity of $\mathcal{O}(|G| \times |L|)$, where $|G|$ is the number of SMs in a group. Note that, even if an attacker fully reconstructs the whole log, VC would still be able to detect this attack. Indeed, due to the properties of PCRs, the attacker would build the new log hash chain starting from a PCR value different to the expected one.

V. SECURITY ANALYSIS

Our data reporting protocol relies on a random forwarding mechanism similar to Crowds [29]. Several works, e.g., [30] and [29], have presented methods for measuring the degree of anonymity for such protocols. However, our specific application scenario makes the proposed verifiable random forwarding mechanism less subject to anonymity attacks. This is true because, at each time interval, *all* SMs generate a new message, containing the measurement data, and encrypt it with the MDMS's public key. Messages will be routed through a random set of SMs before being delivered to the same destination, i.e., MDMS. This makes it impossible to track users based on their activities. Moreover, SMs must follow the protocol, in order to keep their behavior verifiable by the VC. Therefore, they cannot mount collaborative attacks to break other SMs' anonymity.

In what follows, we analyze the security of our proposal based on the requirements (Section II-B) and the security model (Section II-C).

A. Anonymity

To assess the anonymity of the proposed protocol, we compute three parameters: the average number of hops in a path, the average number of hops before the first compromised SM in the path, and the probability that a non-malicious SM, transmitting a message to a compromised SM, is the source of the data. Every intermediate SM delivers a received message to MDMS with probability $\frac{1}{h}$, or, otherwise, forwards it to another SM with probability $p_f = 1 - \frac{1}{h}$. Thus, the parameter h has a direct impact on the anonymity degree of SMs. For example, assume $h = 1$, i.e., all intermediate SMs always deliver the received message to the MDMS. In this case, if a compromised SM receives a message, it can always identify the transmitter as the source SM, since the the number of hops in every path is always 1.

Assume that there are n SMs in a group and, for the sake of simplicity, let every SM be in the communication coverage of all other SMs. Recall that SMs follow a random, yet verifiable, algorithm to choose the next hop in the path. Therefore, the number l of hops in the path follows a geometric distribution supported on the set of positive integers, with expected value $\mathbb{E}[l] = h$. This means that in one time interval an expected number of $n \cdot h$ transmission occurs, and, thus, each SM transmits on average h messages, including its own metering data. As a result, there is a trade-off between anonymity and complexity: while on one hand a large h increases the average path length (and therefore, the anonymity degree for SMs), on the other hand, it increases the computational overhead for SMs.

Assume the internal attacker takes control of $m = r \cdot n$ SMs, where $r \in [0, 1]$ represents the ratio of compromised SMs. We define the effective path-length l_e to be the number of hops before the first compromised SM in the path. The parameter l_e follows a geometric distribution supported on the set of non-negative integers. The process is terminated when the packet is either forwarded to a compromised SM or delivered to the MDMS, which happens with probability $1 - p_f \frac{n-m-1}{n-1}$. Therefore, we have

$$\mathbb{E}[l_e] = \frac{1}{1 - p_f \cdot \frac{n-m-1}{n-1}} - 1 = \frac{h \cdot (n-1)}{h \cdot m + n - m - 1} - 1 \approx \frac{h}{r \cdot (h-1) + 1} - 1, \qquad (1)$$

where $n \gg 1$. Large values of r can significantly decreases the effective path-length $\mathbb{E}[l_e]$. For $r = 0$, we have $\mathbb{E}[l_e] = h - 1$, which is at its maximum value, and, if there is only one non-malicious SM, we have $l_e = 0$.

Finally, let q be the probability that a non-malicious SM, transmitting a message to one of the compromised SMs, is indeed the source of the data. For transmitting the metering data of one time interval, each SM transmits, on average, $h - 1$ messages of other SMs. Therefore, we have

$$q = \frac{1 \cdot \frac{m}{n-1}}{1 \cdot \frac{m}{n-1} + p_f \cdot (h-1) \cdot \frac{m}{n-1}} = \frac{h}{h^2 - h + 1}. \qquad (2)$$

Unlike Crowds [29], [31], q is not a function of r, meaning that even if there is only one non-malicious SM in a group, compromised SMs cannot distinguish its metering data among its other outgoing messages. This happens because the behavior of the compromised SMs needs to be also verifiable by the VC; thus, they cannot stop transmitting their messages to the non-malicious SM, so to later identify its metering data. Even the MDMS or the internal attacker, who have access to the raw data, cannot link the received data to any SM. Again, we can see that there is a trade-off between the anonymity and the complexity, as increasing h will decrease the probability q.
Timing Attack. At each time interval, SMs transmit their metering data following a random delay, which is the waiting time that the number of outgoing messages reaches a pre-determined number. However, SMs include the quantized version $\tilde{t}$ of a time interval in the transmitted message. Therefore, no entity can mount a timing attack even with access to the raw metering data and their corresponding quantized time interval.

B. Confidentiality

Metering data are encrypted with MDMS's public key to ensure that other SMs or an external eavesdropper cannot access the raw data. Therefore, the metering data confidentiality from SMs to MDMS is preserved.

C. Authenticity and Integrity

We intend to protect the MDMS from both *Impersonating Attack*, where an external attacker tries to send fake metering data, and *Replay Attack*, where the attacker tries to re-transmit a previously-transmitted valid packet. We prevent these attacks as follows. Each intermediate SM, I, memorizes the ids of the packets it receives in both the current and previous time intervals. Upon receiving every packet, it checks if the time-interval $\tilde{t}$ specified in the packet does not correspond with the current or the previous time-intervals, or whether it had already received another packet with the same message ID, or if it cannot verify the output of the $\textsc{Hmac}()$ function (line 2 of Algorithm 2). If any of first two conditions hold, the intermediate SM realizes that a replay attack has been performed, and, if the third condition holds, it declares an impersonating attack. In either case, it discards the packet and terminates the forwarding procedure. Finally, MDMS can assess the authenticity and integrity of the received metering data by verifying the HMAC contained in the message.

D. Verifiability

As stated in Section III-B, VC can identify the irregularities in the execution of the protocol by SMs, based on the information contained in the log files. We now explain how VC can effectively detect any malicious behavior from SMs.

A malicious SM may follow different strategies to remain undetected. First, it may stop logging the outgoing packets. The verification procedure in Section III-B can easily detect this behavior by reconstructing the forwarding chain looking for inconsistencies. In this case, for the compromised SM, the verification in line 10 of Algorithm 3 fails. Second, a malicious SM may forward the packet to an arbitrary chosen SM, instead of the one determined by the algorithm, but still writes a correct entry into the log. Similar to the previous case, the verification of line 10 of Algorithm 3 will fail on next node's log file. As a further attempt to remain undetected, the malicious SM may try to modify, insert or delete part of the entries of the log file. In all this cases, log integrity verification will fail.

Finally, by looking at the transmission times in each SM's log, VC can verify whether the SM is correctly following the protocol for waiting to transmit a certain number of messages at once.

VI. Discussion and Evaluation

A. Scalability and Ease of deployment

As stated in Section II-A, our scheme introduces small modifications to the existing electrical infrastructure, which are mainly related to SMs. In particular, our solution brings the following improvements compared to the existing methods.

Unlike the works in [13] and [1], our solution does not rely on a deployed trusted third-party to obtain anonymity, and, unlike [9], we do not require full connectivity among SMs. Furthermore, the average number of packets sent and received by each SM depends only on the parameter h. Indeed, as mentioned in Section V, assuming a constant number of neighbors per SM, each SM transmits an average number of h packets in each time interval. Therefore the overhead on each SM is not affected by the size of the network. This brings several non negligible advantages, such as improved scalability, and easy deployment of new SMs.

Moreover, our protocol does not require complex key management between MDMS and SMs. Indeed, MDMS only pre-distributes a group key k_G to all SMs in a group, which, for example, can be performed at bootstrap time; k_G might be periodically re-generated and efficiently re-distributed via DCI. Furthermore, each SM can advertise its public key or certificate to each of its neighbors, and use it to exchange pairwise keys. By using a simple PKI rooted at MDMS's certificate, each SM can verify whether the received certificate belongs to a legitimate SM (i.e., a SM deployed by MDMS).

B. Communication, Memory and Computation Complexity

Similar to [32], in what follows we estimate overhead and complexity of our solution on SMs under realistic assumptions. We focus on communication, memory, and computation overhead introduced by Initialization and Forwarding Algorithms.

Numeric Evaluation Setting. We assume SMs are low-power battery operated devices, with limited memory and computational capability. Moreover, similar to [17] and existing SM device products [22], we assume each SM is equipped with a low cost TPM chip. In particular, we assume that SMs belong to the same class of low-power devices as a MICAz platform [33] featuring a 4 MHz Atmega128L [34], and a IEEE 802.15.4 compliant transceiver, such as the CC2420, with maximum data rate of 250 Kbps [35]. Furthermore, SMs are assumed to be equipped with an Atmel AT97SC3203S TPM chip [24] designed for embedded systems, and capable of RSA cryptography up to 2048 bit. We assume public key operations are performed inside the TPM chip, to leverage its cryptographic acceleration; therefore, public key encryption and decryption are performed with RSA, using 2048-bit key. SMs also use AES-CBC-128 symmetric encryption.

Communication Cost. Communication for both wireless and PLC [36] is performed over the 802.15.4 link layer protocol, with frame size of 127 bytes and a total of 36 bytes for the header and footer[1] [37]. Moreover, we consider IPHC 6LoWPAN compression [38] for IPv6 and UDP headers, resulting in 2 bytes for each of them. In case of fragmentation, each fragment carries the 802.15.4 header and footer, and IPv6 and UDP headers.

During executions of both Algorithm 1 and Algorithm 2, SMs transmit messages to other SMs encrypted with pairwise symmetric keys. Messages are composed by the concatenation of M (256 bytes)[2], the time reference $\tilde{t}$ (4 bytes), a bit string X (20 bytes), and an $\textsc{Hmac}()$ (20 bytes). The obtained string is then encrypted with AES-CBC-128, resulting in 384 bytes of payload. After fragmentation, we have a total of 635 bytes per message. The message delivered to the MDMS by the last SM in the path consists, instead, of only the 256-bytes M; after fragmentation, this translates into a total of 381 bytes.

Memory Overhead. The memory overhead on each SM depends on the cryptographic parameters and the log file. We assume MDMS's public key and the shared secret mk with VC are pre-installed inside the SM's TPM. Therefore, we do not consider them in our memory overhead estimation. Each SM needs to store only a 128-bit key for each of its neighboring SMs. The log file L introduces a memory overhead linear with the number of stored entries $|L|$, which in turn depends on: (1) the average number of messages transmitted by the SM; (2) the metering data reporting frequency (e.g., every 15 minutes); and (3) the log reporting frequency to VC (e.g., daily). Each log entry consists of 64 bytes.

To better illustrate the total memory overhead on each SM, let us consider an example where: $h = 5$; the average number of neighbors per SM is 10; SMs send a metering report every 15 minutes, while the log file to VC on a daily basis. On average, each SM would need to store approximately: $16 \times 10 = 160$ bytes for pairwise keys; a random 20-byte string X for each transmitted message for two time intervals, therefore $20 \times 2 \times h = 200$ bytes; and, considering an average of h entries stored per each time interval $\tilde{t}$, a log of size 30.00 Kbytes per day[3]. This results in a total approximate

[1]We do not include the 802.15.4 message authentication code field.

[2]M consists of the RSA encryption of the metering data (represented in 8 bytes), $\tilde{t}$, the group identifier G of 4 bytes, and an $\textsc{Hmac}()$.

[3]Each log entry has a size of 64 byte; thus, we have $64 \times h \times 24 \times 4$ byte.

overhead of 30.35 Kbytes, which is easily affordable even by resource constrained platforms.

Computation Overhead. We measure the computation overhead of our protocol on SMs in terms of both required execution time, and energy consumption. We refer to [39] for estimating the cost of TPM operations, while we use the measurements in [34] and [35] to estimate respectively the cost of other cryptographic operations and transmission/reception. Table II summarizes the overhead introduced by the operations we use in our algorithms, in terms of energy and time.

TABLE II
ENERGY CONSUMPTION

Atmel AT97SC3203S TPM [24], [39]		
Function	**Energy** (mJ)	**Time** (ms)
RSA Encryption	0.04 (× byte)	0.03 (× byte)
TPM _PCR_READ()	0.92	5.8
TPM _PCR_EXTEND()	$0.49 \times x + 50$	$0.49 \times x + 250$
TPM _PCR_QUOTE()	268.80	1400
Atmega128L (4 MHz) [34]		
Function	**Energy** × **byte** (μJ)	**Time** × **byte** (μs)
AES-CBC-128 Encryption	1.62	117.39
AES-CBC-128 Decryption	2.49	180.43
SHA-1() and HMAC()	5.9	453.85
Transmit	4.8	73.84
Receive	5.36	74.40

The overall computation cost is dominated by cryptographic operations and data transmission and reception. Transmission and reception of messages between SMs require respectively 46.89 ms (3.05 mJ), and 47.24 ms (3.40 mJ). Similarly, the transmission of the encrypted metering data to MDMS requires 18.90 ms (1.23 mJ). Also, writing into the log roughly consists of the execution of TPM _PCR_EXTEND() on an input of the length $20 + 20 + 4$ bytes, plus the execution of TPM _PCR_READ(). According to Table II, these operations require a total of 277.36 ms, and an energy cost of 72.48 mJ.

Line 1 of Algorithm 1 requires one HMAC() operation on 8 byte, one RSA encryption (inside the TPM) of 36 bytes, resulting in 4.71 ms, and an energy consumption of 1.60 mJ. We assume pseudo-random number generation is no more expensive than computing a SHA-1() operation over 20 bytes. Therefore, we estimate the time overhead of Line 2 as 9.08 ms, and the respective energy cost as 0.12 mJ. Line 4 of Algorithm 1 requires one HMAC() operation on 24 bytes and an AES-CBC-128 encryption on 300 bytes; in total, this has a time overhead of 11.38 ms, and an energy overhead of 0.63 mJ. Finally, the last instruction of Algorithm 1 writes into the log. Thus, the total time and energy required by the Initialization Algorithm can be estimated, respectively, as 349.42 ms, and 77.88 mJ.

Line 1 of Algorithm 2 requires the decryption of 300 bytes, which takes approximately 54.13 ms, with an energy cost of 0.75 mJ. Lines 2-4 of Algorithm 2 require the verification of an HMAC() on 24 bytes, which takes approximately 10.89 ms, at a cost of 0.14 mJ. Line 6 requires the computation of SHA-1() on 40 bytes, resulting in a total execution time of 18.15 ms, and energy cost of 0.24 mJ. Depending on the value of c, the algorithm may deliver M to MDMS, which has a time overhead of 18.90 ms, and an energy overhead of 1.23 mJ, or create an encrypted message and send it to its j-th neighbor, with a total cost in terms of execution time and energy consumption of 58.27 ms and 3.68 mJ, respectively. Finally, in any case the last instruction of Algorithm 2 writes into the log. Therefore, the execution of the Forwarding Algorithm requires a total of 379.43 ms, and an energy cost of 74.84 mJ, in case the SM delivers the metering data to MDMS, while 418.80 ms and 77.29 mJ in the the other case.

Finally, assuming the same technology for SM-to-VC and SM-to-MDMS communication we can easily estimate the time and energy required to report the log file to the VC, based on the number of entries $|L|$ in SM's log file L. Such operation requires the transmission of the whole log (i.e., $|L|$ entries of 64 byte), as well as one execution of TPM _PCR_QUOTE(), resulting in $18.90 \times |L| \times 64 + 1400$ ms, and $1.23 \times |L| \times 64 + 268.80$ mJ. Note that, while this operation is clearly the most expensive, it is much less frequent compared to Initialization and Forwarding Algorithms. The verification frequency can be tuned in order to find a balance between the security and computational costs.

VII. RELATED WORK

In this section, we review the main related works for anonymous metering data reporting.

A first approach to guarantee anonymity for metering data reporting, is to rely on a trusted-third party [1], [13]. In [13], the authors proposed a method for sending aggregate metering data to an honest-but-curious Utility Provider, (the equivalent of the MDMS in our system model). The method uses a semi-trusted third-party entity composed of multiple aggregators, and guarantees SM anonymity by leveraging a secret sharing mechanism.In [1], Efthymiou et al. proposed a trusted third-party escrow mechanism to allow authenticated anonymous meter readings without the possibility to associate the metering data with users.

A different approach is to transmit only an aggregate metering data report to the MDMS, e.g., aggregating the data consumption of a single user over a limited time [8], or aggregating metering data of a population [9]–[12]. In [8], the authors proposed a solution that achieves smart metering data anonymity by adding minimal noise to aggregate consumption of a single meter over a period of time. In [9], the authors proposed a protocol for metering data anonymity, by assuming full connectivity among SMs; data anonymity is achieved by secure data aggregation (performed by a trusted SM in a group) via homomorphic encryption. Similarly, in [12] Steiner et al. provided SM anonymity by combining secret sharing and homomorphic encryption. Other similar approaches are proposed in [10], [21], [26], [40]. In [11] Ohara et al. proposed a method that allows the MDMS to learn only the total amount of consumption during a billing interval for each customer, and the total amount of consumption of customers for each short interval. MDMS, however, cannot learn the detailed information of the consumption profiles. A slightly different approach is to employ trusted hardware to build solutions for anonymous reporting. As an example, the work in [17] uses a TPM v1.2 component to provide aggregate metering data to MDMS.

While being effective solutions, previous works, however, have several limitations. Using a trusted third-party entity that actively participate in the protocol execution is often expensive and difficult to deploy and manage. It also introduces a single point of failure into the system. Also, compared to having the individual measurements, aggregation significantly reduces the utility of the data for the operation center. In contrast, our proposed protocol provides anonymity for fine-grained metering data reporting, relying on a collaborative mechanism among SMs. The protocol is shown to be secure and computationally efficient for the resource-constrained SM devices.

VIII. Conclusions

In this paper, we presented the design of a privacy-preserving fine-grained metering data collection from Smart Meters (SMs). In our proposed infrastructure, SMs follow a collaborative multi-hop protocol, which allows them to anonymously transmit periodic metering data to a Metering Data Management System (MDMS). We designed our solution considering an adversarial setting where MDMS is honest-but-curious, an internal attacker can take control of a portion of SMs, and other non-compromised SMs are honest-but-curious. By construction, our collaborative forwarding protocol is both random and verifiable. Indeed, it allows a trusted Verification Center, an entity who does not actively participate in the execution of the protocol, to evaluate the functionality of each SM by accessing its internal log. We provided a thorough security analysis, and, considering realistic assumptions, we assessed the feasibility of our protocol in terms of the overhead on SMs.

References

[1] C. Efthymiou and G. Kalogridis, "Smart Grid Privacy via Anonymization of Smart Metering Data," in *IEEE SmartGridComm '10*, 2010.

[2] (2015) U.S. Energy Information Administration: Electricity Monthly Update. [Online]. Available: http://www.eia.gov/electricity/monthly/update/archive/ april2015/

[3] D. of Energy and C. Change, "Smart Meters, Great Britain, Quarterly report to end March 2015," Jun. 2015.

[4] G. W. Hart, "Nonintrusive Appliance Load Monitoring," *Proc. IEEE*, vol. 80, no. 12, 1992.

[5] C. Laughman, K. Lee, R. Cox, S. Shaw, S. Leeb, L. Norford, and P. Armstrong, "Power signature analysis," *IEEE Power Energy Mag.*, vol. 1, no. 2, 2003.

[6] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin, "Private Memoirs of a Smart Meter," in *ACM BuildSys '10*, 2010.

[7] M. Lisovich, D. K. Mulligan, S. B. Wicker *et al.*, "Inferring personal information from demand-response systems," *IEEE Security & Privacy Magazine*, vol. 8, no. 1, 2010.

[8] G. Danezis, C. Fournet, M. Kohlweiss, and S. Zanella-Béguelin, "Smart Meter Aggregation via Secret-Sharing," in *ACM SEGS '13*, 2013.

[9] Z. Erkin and G. Tsudik, "Private Computation of Spatial and Temporal Power Consumption with Smart Meters," in *ACNS'12*, 2012.

[10] K. Kursawe, G. Danezis, and M. Kohlweiss, "Privacy-friendly Aggregation for the Smart Grid," in *PETS '11*, 2011.

[11] K. Ohara, Y. Sakai, F. Yoshida, M. Iwamoto, and K. Ohta, "Privacy-preserving Smart Metering with Verifiability for Both Billing and Energy Management," in *ACM ASIAPKC '14*, 2014.

[12] M. Steiner, G. Tsudik, and M. Waidner, "Key agreement in dynamic peer groups," *IEEE Trans. Parallel Distrib. Syst*, vol. 11, no. 8, 2000.

[13] T. Dimitriou and G. Karame, "Privacy-Friendly Tasking and Trading of Energy in Smart Grids," in *ACM SAC '13*, 2013.

[14] A. Rial and G. Danezis, "Privacy-preserving smart metering," in *WPES '11*, 2011.

[15] M. Ambrosin, H. Hosseini, K. Mandal, M. Conti, and R. Poovendran, "Verifiable and privacy-preserving fine-grained data-collection for smart metering," in *IEEE CNS/SPiCy '15*, 2015.

[16] J. Zhou, R. Q. Hu, and Y. Qian, "Scalable Distributed Communication Architectures to Support Advanced Metering Infrastructure in Smart Grid," *IEEE Trans. Parallel Distrib. Syst*, vol. 23, no. 9, 2012.

[17] H.-Y. Lin, W.-G. Tzeng, S.-T. Shen, and B.-S. P. Lin, "A practical smart metering system supporting privacy preserving billing and load monitoring," in *ACNS '12*, 2012.

[18] I. Kitagawa and S. Sekiguchi, "Technologies Supporting Smart Meter Networks," *FUJITSU Sci. Tech. J.*, vol. 49, no. 3, 2013.

[19] "The role of communication technology in Europe's advanced metering infrastructure," 2014. [Online]. Available: https://www.accenture.com/us-en/insight-role-communication-technology-europe-advanced-metering.aspx

[20] R. R. Mohassel, A. Fung, F. Mohammadi, and K. Raahemifar, "A survey on advanced metering infrastructure," *International Journal of Electrical Power & Energy Systems*, vol. 63, 2014.

[21] B. Defend and K. Kursawe, "Implementation of Privacy-friendly Aggregation for the Smart Grid," in *ACM SEGS '13*, 2013.

[22] (2014) Optimized ARM Smart Meter. [Online]. Available: http://www.arm.com/markets/embedded/smart-meter.php

[23] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan, "TrustLite: A security architecture for tiny embedded devices," in *EuroSys '14*, 2014.

[24] (2007) The Atmel Trusted Platform Module. [Online]. Available: http://www.atmel.com/images/doc5128.pdf

[25] "Terminology for Talking about Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management," 2010. [Online]. Available: https://tools.ietf.org/id/draft-hansen-privacy-terminology-00.html

[26] R. Lu, X. Liang, X. Li, X. Lin, and X. S. Shen, "EPPA: An efficient and privacy-preserving aggregation scheme for secure smart grid communications," *IEEE Trans. Parallel Distrib. Syst*, vol. 23, no. 9, 2012.

[27] "TPM Main – Part 1 Design Principles, Specifications version 1.2, Level 2 Revision 116," Mar. 2011.

[28] "TPM Main – Part 3 Commands, Specifications version 1.2, Level 2 Revision 116," Mar. 2011.

[29] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM TISSEC*, vol. 1, no. 1, 1998.

[30] C. Diaz, S. Seys, J. Claessens, and B. Preneel, "Towards Measuring Anonymity," in *PETS '03*, 2003.

[31] M. Wright, M. Adler, B. N. Levine, and C. Shields, "An analysis of the degradation of anonymous protocols." in *NDSS '02*, vol. 2, 2002.

[32] N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann, "Seda: Scalable embedded device attestation," in *ACM SIGSAC CCS '15*, 2015.

[33] (2015) MICAz Wireless Measurement System. [Online]. Available: http://www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf

[34] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *IEEE PerCom '05*, 2005.

[35] G. De Meulenaer, F. Gosset, F.-X. Standaert, and O. Pereira, "On The Energy Cost of Communication and Cryptography in Wireless Sensor Networks," in *IEEE WIMOB '08*, 2008.

[36] D. Popa and J. Hui, "6LoPLC: Transmission of IPv6 Packets over IEEE 1901.2 Narrowband Powerline Communication Networks," 2014.

[37] (2015) CCN and ndn TLV encodings in 802.15.4 packets. [Online]. Available: https://www.ietf.org/mail-archive/web/icnrg/current/pdfs9ieLPWcJI.pdf

[38] J. Hui and P. Thubert, "Compression format for IPv6 datagrams over IEEE 802.15. 4-based networks," 2011.

[39] W. Hu, H. Tan, P. Corke, W. C. Shih, and S. Jha, "Toward Trusted Wireless Sensor Networks," *ACM TOSN*, vol. 7, no. 1, 2010.

[40] F. D. Garcia and B. Jacobs, "Privacy-friendly Energy-metering via Homomorphic Encryption," in *STM '11*, 2011.