A Secure and Privacy–Preserving Approach to Communications in Smart Grids

Christian Callegari, Stefano Giordano, Michele Pagano, and Gregorio Procissi

CNIT and Dept. of Information Engineering,
University of Pisa,
Pisa, Italy
E-mail: {c.callegari,s.giordano,m.pagano,g.procissi}@iet.unipi.it

Abstract: Smart grids are electricity networks that can intelligently integrate the action of all connected users - producers, consumers, and prosumers (i.e., producers and consumers at the same time) - in order to efficiently deliver sustainable, economic and secure electricity supplies. Smart grids rely on a two–way communication infrastructures in which energy measurement data taken at users located smart meters, as well as control information, must be exchanged in real–time or – at least – near real–time, while simultaneously guaranteeing a secure and privacy preserving communication. This paper addresses such a problem by presenting a distributed communication architecture that guarantees the privacy of fine–grained users data while enabling the energy supplier to access aggregate energy measurements and per–user coarse–grained data for billing purposes. The key idea underlying the proposed architecture is to adopt a Secure Multiparty Computation method based on a verifiable secret sharing of the keys used by the smart meters to encrypt their sensitive data. This approach allows to remove the need for any intermediate aggregator element with clear benefits in terms of scalability and robustness.

Keywords: Smart Grid; Secure Multiparty Computation; Homomorphic Secret Sharing; Bi-Homomorphic Cryptography

1 Introduction and Motivation

The power grid is probably the most fundamental and crucial large scale infrastructure as everyone depends on it in all daily activities. However, in spite of that, present power grids have been conceived on technology dating from the beginning of the last century when renewable energies and consumers able to produce energy were far to come.

As such, huge modernization efforts are undergoing to migrate from current grids to the so called *smart grids*, namely grids capable of meeting new requirements such as energy efficiency, reduced environmental impacts, incorporation of alternative energy sources, and so forth.

The term *smart grid* [1, 2] commonly denotes the new engineering paradigms and challenges in electrical grid that go beyond the traditional *broadcast* grids, aiming at

integrating the action of all connected users - producers, consumers, and prosumers (i.e., producers and consumers at the same time) - in order to efficiently deliver sustainable, economic and secure electricity supplies.

A smart grid leverages on emerging products and services together with monitoring, control, communication, and self-healing technologies [3] to:

- facilitate the connection and operation of generators of all sizes and technologies;
- allow consumers to play a part in optimizing the operation of the system;
- provide consumers with greater information and choice of supply;
- significantly reduce the environmental impact of the whole electricity supply system;
- deliver enhanced levels of reliability and security of supply;
- save energy, reduce cost and increase reliability and transparency.

The bottom-line feature of the *smart grid* concept is that its components are interconnected by means of a two–way communications network, fostering real–time or – at least – near real–time communication of monitoring and control information. Such an information should be effectively and reliably communicated across the four conceptual components of the power grid, namely generation, transmission, distribution and customers, as well as the entity in charge of operational control.

The key players in *smart grid* are the *smart meters*, namely meters introduced at prosumers locations that measure energy consumption and production in much more detail than conventional meters and –mostly automatically– communicate this information to associated parties (typically utility providers) over IP networks, to perform several operations, such as billing and accounting functions as well as innovative user profiling operation to optimise the energy distribution.

These last mentioned operations raise significant concerns both in term of *scalability* (the volumes of data generated by a large number of devices may easily reach high figures) and *privacy*. Indeed, it is pretty obvious that such detailed data may reveal sensitive information about the users behavior (e.g., if she is at home and to some extent which devices she is using) [4] and therefore need to be transferred from the smart meter to the utility server (i.e., the energy supplier) over a secure channel.

The above mentioned privacy concerns have been recently addressed in [5], by elaborating a set of guidelines regarding the transmission of such data and the type and amount of information that has to be transmitted to the different parties in the energy grid (also see works referred to in Section 2). In particular, a lot of research efforts have been devoted to the need of securely exchanging data produced by the frequent metering of the energy production/consumption and needed by a utility or electrical energy distribution network for operational reasons. Conversely, it has been advocated that such data may not necessarily need to be attributable to a specific smart meter or consumer, while it must be securely attributable to a specific location (for example a group of houses or apartments) within the electricity distribution network. Therefore, with the exception of the information required by billing purposes, a large amount of such data can be anonymised at the user level and then transmitted, while still preserving the location information.

This paper finds its roots (and borrows several functional components from) in the previous papers [6, 7] as it proposes a privacy preserving communication architecture for *smart grids* that relies on a Secure Multiparty Computation method based on Verifiable

Secret Sharing of data. However, differently from [6, 7], it brings in a completely new approach about the type of data used for the secret sharing phase and simplifies the overall architecture. Indeed, the current proposal bases the security and privacy of the transmission of the measurement data on the sharing (i.e., secret sharing) of encryption keys rather than of energy consumption values themselves (please note that the idea will be better described in the following). On one side, in a trusted environment, this allows a less frequent exchange of data (indeed, if energy consumption changes instantaneously, the key used to encrypt such data may be updated only periodically) with obvious benefit in terms of the scalability of the overall system. On the other side, the overall complexity is strongly reduced as this approach allows to totally remove the need for intermediate aggregators (the so–called *Privacy Peers)* that, in turn, increases the robustness of the system itself (the intermediate Privacy Peers may possibly represent vulnerable points of failures).

The rest of the paper, which significantly extends the preliminary work presented in [8], is organized as follows: section 2 reviews the literature more directly connected to our proposal, while section 3 provides the reader with the theoretical background information needed for better understanding our proposal. Then, section 4 presents the innovative contribution of this paper by introducing the distributed communication architecture idea at a system level, while postponing the the analysis of the security of the proposed scheme to section 5. Final remarks are given in section 6.

2 Related Works

Security and privacy issues related to the emerging *smart grid* and smart metering networks have stimulated a pretty large number of works focusing on this topic. Far from being fully exhaustive, the following review presents the papers that – to the best of our knowledge – are somewhat related to our approach.

Peers for Privacy (P4P) [9, 10, 11, 12] is a generic framework for distributed and privacy–preserving computation that may apply to many real–world applications. It features a novel heterogeneous architecture and a number of efficient tools for performing private computation and ensuring security at large scale, as well as providing resiliency against realistic adversaries. However, P4P does not address the issue of billing or account management for which the metering data need to be attributable, i.e. securely attached to a particular consumer and/or account holder.

The work [13] introduces a communication architecture based on a new set of functional nodes, the so–called Privacy Preserving Nodes (PPNs), in charge of collecting customer data masked by means of Shamir's Secret Sharing scheme and aggregate them directly in the masked domain, according to the consumer's needs and access rights. The information consumers, such as utilities and third parties, can recover the aggregated data by collecting multiple shares from the PPNs. However, such an architecture assumes that both the node and the PPN are trusted entities and it adopts the honest-but-curious adversary model only.

Acs and Castelluccia [14] proposes an architecture based on the use of SSS applied to noisy data, without the need for an intermediate aggregator. Nonetheless, as discussed in the paper our proposal is much more efficient, not requiring the application of the SSS scheme at each transmission round.

The authors of [15] focus on the privacy and on the vulnerabilities of smart metering data and propose a solution for anonymising high–frequency metering data. In a few words, they address the smart metering privacy issue by anonymising the identity of high–frequency

metering data through an escrow service. They point out that the key to the security offered is the trust level of such an escrow service, along with the random time intervals between the setup of attributable and anonymous data profiles at the smart meter.

The approach of [16] is based on the use of a trusted third party (TTP). Smart meters encrypt their measures and send them to an intermediate node that performs aggregation and forwards the result to the energy supplier. Such an approach is based on conventional cryptography; the intermediate node can see the single values of measurements, although it may not be able to map such values to the specific originating smart meters. Hence, the system guarantees privacy at the user level as long as the TTP and the energy supplier do not maliciously collaborate.

The big benefit of our architecture over that of [16] relies on the amount of trust given to the aggregator. Indeed, in our approach, the aggregator is the energy supplier itself which, thanks to bi–homomorphic cryptography, cannot see the single measures of smart meters as it does not have the single keys used to encrypt such data. Only a coalition of at least k smart meters and the energy supplier may reveal single keys to open the smart meters secrets.

In [17] the authors propose a two–protocol that allows the energy supplier to know the aggregate consumption values of a group of smart meters by taking advantage of a combination of additive homomorphic cryptography and secret sharing of data. In each reference period, each smart meter executes $n - 1$ data encryption and one data decryption to aggregate the measures taken in a group of n smart meters. With respect to our proposal, such a scheme is more computationally intensive: indeed, our scheme achieves a lower computational cost at the energy supplier due to the use of a (lighter) symmetric key cryptography versus a public key cryptography and to the aggregation of the keys themselves instead of the measured data. This way, such an operation may be performed less frequently. For these reasons, our system turns out to be less resource consuming that those of [6, 7, 18] as well.

In particular, the authors of [18] present an approach based on homomorphic cryptography, in which data are sent through the smart meters that, on their side, are logically organised in a tree. Each smart meter performs a homomorphic aggregation and passes data up stream all the way to the root node (the collector node). This scheme assumes that all smart meters are honest, hence they do not (either on purpose or by mistake) cheat. In [19], instead, a smart meter at–a–time is elected as key aggregator of a group. As such, it needs to be trusted against malicious attackers as it collects (and knows) all the keys.

Finally, as widely discussed in the introduction, the works [6, 7] originated this work. However, the current proposal simplifies their architecture by changing the nature of the data (the encryption keys rather than the energy consumption data) to be shared amongst the system components.

3 Theoretical Background

This section presents the details on the functional building blocks adopted in our approach, i.e. cryptography, verifiable secret sharing scheme, and secure multiparty computation. For the sake of clarity, in the presentation some background notions about such techniques are also given.

3.1 Definitions

The following definitions [10] will be used in the paper :

- **Homomorphic Commitment** Given an integer value a, a homomorphic commitment to a with randomness r is written as $C(a, r)$. It is homomorphic in the sense that $C(a, r)C(b, s) = C(a + b, r + s)$. It is cryptographically hard to determine a given $C(a, r)$. We say that a prover "opens" the commitment if it reveals a and r.
- **Zero-Knowledge Proof (ZKP)** ZK protocols allow a prover to demonstrate the knowledge of a secret to a verifier, without revealing any partial information that would help the verifier to infer the secret, other than the fact that the prover knows the secret.
- **ZKP of Knowledge** A prover who knows a and r (i.e. who knows how to open $C(a, r)$) can demonstrate that it has this knowledge to a verifier who only knows the commitment. The proof does not reveal anything about a or r.
- **ZKP for equivalence** Let $A = C(a, r)$ and $B = C(a, s)$ be two commitments to the same value a. A prover who knows how to open A and B can demonstrate to a verifier in ZK that they commit to the same value.
- **ZKP for product** Let A, B and C be commitments to a, b, c respectively, where $c = ab$. A prover who knows how to open A, B, C can prove in ZK to a verifier who has only the commitments that the relationship $c = ab$ holds among the values they commit to.
- **Bit commitment** Let $A = C(a, r)$ be a commitment to a value a where $a \in \{ 0, 1 \}$, which is called a bit commitment. A prover who knows how to open A can prove in ZK that it commits to either 0 or 1 (but not to which value).
- **ZKP for boundedness** Let $A = C(a, r)$ be a commitment to a value a. Using the above methods, a prover can show that A contains a k-bit integer, i.e. that it encodes the same value as $B_{k-1} \cdots B_0$, where each B_j encodes 0 or 2^j . If the leading "bit" B_{k-1} instead encodes 0 or $L - 2^{k-1} + 1$ where $k = \lfloor \log_2 L \rfloor$, then the ZKP proves that $a \in [0, \ldots, L]$ for any k-bit positive L. Adding an additional bit which encodes 0 or $-L$, gives a proof of boundedness in the range $[-L, \ldots, L]$.

3.2 Bi–Homomorphic Cryptography

Homomorphic encryption refers to a set of encryption algorithms with the property that computations carried out on cipher-text generate an encrypted result that, once decrypted, matches the result of operations performed on the plaintext. As an example, we can consider the case in which the decryption of the sum of several cipher-texts (encrypted by using the same encryption key k) gives the sum of the corresponding plain-texts. Formally, we can write:

$$E\left(X_1 + \cdots + X_n, k\right) = E\left(X_1, k\right) + \cdots + E\left(X_n, k\right) \tag{1}$$

Bi-homomorphic encryption represents an extension of the homomorphic encryption, in which the homomorphic property is also applied to the encryption keys and not only to the cipher-texts. To better clarify such an example, let us refer to the following:

$$E\left(X_1 + \cdots + X_n, k_1 \cdots + k_n\right) = E\left(X_1, k_1\right) + \cdots + E\left(X_n, k_n\right) \tag{2}$$

in which the sum of several cipher-texts, each encrypted with a different encryption key k_i, can be decrypted with a key given by the sum of the encryption keys, obtaining the sum of the corresponding plain-texts.

A simple – though effective – solution is provided by the bi–homomorphic additive encryption scheme from Castelluccia [20].

Encryption:

1. let X be an integer in $\{0, 1, \ldots, M-1\}$ with M large;
2. let k be a random key in $\{0, 1, \ldots, M-1\}$;
3. compute $\hat{X} = E(X, k, M) = X + k \mod M$;

Decryption:

1. $X = D\left(\hat{X}, k, M\right) = \hat{X} - k \mod M$;

Notice that it is trivial to verify the bi–homomorphic additive property as:

$$\begin{aligned} E(X_1, k_1, M) + E(X_2, k_2, M) &= X_1 + X_2 + k_1 + k_2 \mod M \\ &= E(X_1 + X_2, k_1 + k_2, M) \end{aligned}$$

and, conversely,

$$\begin{aligned} D\left(\hat{X}_1 + \hat{X}_2, k_1 + k_2, M\right) &= D\left(\hat{X}_1, k_1, M\right) + D\left(\hat{X}_2, k_2, M\right) \\ &= \hat{X}_1 - k_1 + \hat{X}_2 - k_2 \mod M \\ &= X_1 + X_2 \end{aligned}$$

3.3 Secret Sharing Scheme

One of the key elements to protect personal data is to split the secret data and to share portions of them among all of the parties participating in the communication. In general, the method for a party (the dealer) to share a secret among a group of n participants, such that each of them is allocated a portion of the secret itself, is called Secret Sharing Scheme (SSS).

An SSS consist of two phases:

- a *distribution* phase, in which the dealer who knows the secret s creates and distributes the shares s_i amongst the participants P_i;
- a *reconstruction* phase in which a set of participants form a coalition to recover the secret by combining their shares.

Several schemes are defined on the basis of the method used to create the shares, two of the most commonly used ones are:

- **Additive Secret Sharing** Given a secret $s \in F$, where F is a Galois Field $GF(p)$, the *dealer* D selects $n-1$ random integers $r_1, \ldots, r_{n-1}$ uniformly from F, and then computes

$$s_n = s - \sum_{i=1}^{n-1} r_i \mod p$$

D sends each player P_i, $1 \leq i \leq n-1$, the share $s_i = r_i$, and the share s_n is sent to P_n. The reconstruction of secret s is trivial and it is simply obtained adding all of the shares together:

$$s = \sum_{i=1}^{n} s_i \bmod p$$

The above additive secret sharing scheme requires all participants to contribute their shares in order to reconstruct the secret. If one or more of the participants are missing, no information about the original secret can be recovered; such a scheme is known as a perfect secret sharing scheme[1].

- **Shamir Secret Sharing** Shamir constructs a (k, n)-threshold secret sharing scheme to show how to divide a secret s into n pieces in such a way that the secret is easily reconstructable from any $k \leq n$ pieces, but even complete knowledge of $k-1$ pieces reveals absolutely no information about s [21]. In order to create the shares, the dealer D first selects a secret $s \in F$, where F, as before, is a Galois Field $GF(p)$, then constructs a random $k-1$ degree polynomial

$$f(x) = s + r_1 x + r_2 x^2 + \cdots + r_{k-1} x^{k-1} \bmod p$$

subject to the following conditions:

  - the secret $s \in F$ is the free term
  - the threshold $k \leq n$
  - the coefficients $\{ r_1, \ldots, r_{k-1} \}$ are chosen independently and randomly from the interval $[0, p)$

Each share s_i of the secret can be then created by an evaluation of the function f:

$$s_1 = f(1), s_2 = f(2), \ldots, s_n = f(n).$$

Finally, given n participants $\{ P_1, \ldots, P_n \}$, the secret can be reconstructed by using polynomial interpolation: a minimum of k participants, one more than the degree of the polynomial, will have to contribute to the reconstruction of the polynomial. Some of the useful properties of the Shamir (k, n)-threshold schema are:

1. Secure: information theoretic secure
2. Minimal: the size of each piece does not exceed the size of the original data
3. Extensible: when k is kept fixed, some pieces can be dynamically added or deleted without affecting the other pieces
4. Dynamic: security can be easily enhanced without changing the secret, but by changing the polynomial occasionally (keeping the same free term) and constructing new shares to the participants
5. Flexible: in organizations where hierarchy is important, we can supply each participant different number of pieces according to his importance inside the organisation. For instance, the president can unlock the safe alone, whereas 3 secretaries are required together to unlock it

6. Homomorphic: shares of multiple secrets combine together to form a "composite share"; the composite shares are shares of composite secrets.

In our architecture, we require a (k, n)–threshold scheme, that is a secret sharing scheme in which the secret is divided into n portions, but any k portions of it are sufficient to fully reconstruct it. The main reason for using a threshold scheme is resiliency: indeed, if all n portions were required to reconstruct the secret, even the failure of a single smart meter would prevent the secret recovery.

As far as the architecture concerns, every threshold SSS may work as good as long as the selected SSS holds the homomorphic property with respect to the sum operation. In other words, we require that the sum of the shares of the secret are shares of the sum of the secrets.

Several known SSSs satisfy the previously listed requirement. Out of them, we selected the well known Shamir secret sharing scheme [21], even though others homomorphic schemes such as the Blakeley [22] scheme or the Asmuth–Bloom [23] (to cite the most famous only) may work as well.

It is worth noticing that the Shamir secret sharing scheme proves to be *perfect* (in the information theoretic sense), in that even the complete knowledge of $k - 1$ pieces of the secret does not disclose any information on the overall secret.

3.4 Verifiable Secret Sharing

Since all SSSs rely on the distribution of shares from a dealer to a set of participants, they all can suffer from the following drawbacks:

- during the distribution phase, the dealer may send inconsistent shares, i.e. not all of them simultaneously obey to the mathematical formula used for their distribution (in case of Shamir SSS, that means they may not simultaneously belong to the same polynomial);
- during the reconstruction phase, players may contribute false shares so that $\tilde{s} \neq s$ is reconstructed, instead.

To address the above listed issues, Verifiable Secret Sharing (VSS) schemes have been introduced in order to convince players that their shares are k-consistent, i.e., each player is assured that every subset of k out of n shares defines the same secret.

The basic idea underlying VSS is that the dealer sends some extra information to each participant during a distribution phase and each participant verifies that her secret share is consistent with this extra information. The extra information is called *commitment* [24, 25]. By making a commitment, a player is able to choose a value from some (finite) set and commit to this choice, with no chance to change this value. However, she does not have to reveal her choice – although she may choose to do so at some later time.

In our proposal, we adopt the VSS proposed by Pedersen [26] as a natural complement of the Shamir secret sharing scheme. The Pedersen method proves to be secure in the information theoretic sense, but the consistency of the shares is only computationally secure.

According to the Pedersen VSS scheme, a player is allowed to non-interactively check whether the share she has received is consistent. The method works as follows: Let p and q be prime, and assume q divides $p - 1$ (typically $p = 2q + 1$, for suitable values). Let $\boldsymbol{G}_q$ be a subgroup of $\boldsymbol{Z}_p$ of order q and g a generator of $\boldsymbol{G}_q$. Let g and h be elements of $\boldsymbol{G}_q$ chosen by a trusted party before using the scheme or by some other cryptographic primitive

(like coin flipping) so that $\log_g h$ remains unknown to the participants. Define the operation commit to be:

$$C(s,r) = g^s h^r \bmod p$$

where $s \in \boldsymbol{Z}_q$ is the value, the dealer wishes to commit to, and $r \in \boldsymbol{Z}_q$ is a random value chosen by the dealer. The commit operation is perfectly hiding, that is, for any $s' \neq s$ we can find unique r' such that

$$C(s',r') = g^{s'} h^{r'} = g^s h^r = C(s,r)$$

and computationally binding, that is, the player committing to s cannot open a commitment to $s' \neq s$ unless she can solve $\log_g h$. The scheme is also homomorphic:

$$\begin{aligned} C(a+b, r_a + r_b) &= g^{a+b} h^{r_a + r_b} \\ &= g^a h^{r_a} \cdot g^b h^{r_b} \\ &= C(a, r_a) \cdot C(b, r_b) \end{aligned} \tag{3}$$

In order to create a VSS scheme by using the Pedersen commitment, given $g, h \in \boldsymbol{G}_q$, the Shamir distribution scheme of a secret $s \in \boldsymbol{Z}_q$ changes as follows:

1. the dealer publishes a commitment to s: $c_0 = C(s,r)$ for a randomly chosen $r \in \boldsymbol{Z}_q$;
2. the dealer chooses two polynomials of degree at most $k-1$:
   $f = s + a_1 x + \cdots + a_{k-1} x^{k-1}$ and $f' = r + b_1 x + \cdots + b_{k-1} x^{k-1}$
3. the dealer secretly sends $s_i = f(i)$ and $r_i = f'(i)$ to each participant P_i, for $1 \leq i \leq n$ and broadcasts the commitments $c_1 = C(a_1, b_1), \ldots, c_{k-1} = C(a_{k-1}, b_{k-1})$

When each participant has received her share (s_i, r_i), she can verify its validity by the equation:

$$c_i = c_0 \prod_{j=1}^{k-1} c_j^{i^j}.$$

3.5 Secure Multiparty Computation

SMPC [25] is a method that allows parties to jointly compute a function over their inputs, while at the same time keeping these inputs private.

In Multiparty Computation (MPC), we consider a number of players $P_1, \ldots, P_n$, who initially hold inputs $x_1, \ldots, x_n$, and we want to securely compute some function f on these inputs, where $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$, such that P_i learns y_i but no other information. The goal can be accomplished by an interactive protocol π that the players execute. Intuitively, we want that executing π is equivalent to having a trusted party T that privately receives x_i from P_i, computes the function, and returns y_i to each P_i [27].
This should hold, even if players exhibit some amount of adversarial behaviour.

SMPC is concerned with the possibility of deliberately malicious behaviour by some adversarial entity [28], either an external entity, or a subset of the participating parties. The

aim of this attack may be to learn private information or cause the result of the computation to be incorrect. Thus, two important requirements on any secure computation protocol are privacy and correctness. The privacy requirement states that nothing should be learned beyond what is absolutely necessary (i.e., parties should learn their own output and nothing else). Instead, the correctness requirement states that each party should receive its correct output.

The parties under the control of the adversary are called corrupted, and follow the adversary instructions.

1. **Corruption strategy** The corruption strategy deals with the question of when and how parties are corrupted. There are two main models:

   - Static corruption model: In this model, the adversary is given a fixed set of parties whom it controls. Honest parties remain honest throughout and corrupted parties remain corrupted.
   - Adaptive corruption model: Rather than having a fixed set of corrupted parties, adaptive adversaries are given the capability of corrupting parties during the computation. The choice of who to corrupt, and when, can be arbitrarily decided by the adversary and may depend on its view of the execution (for this reason it is called adaptive). This strategy models the threat of an external "hacker" breaking into a machine during an execution. It can be noted that in this model, once a party is corrupted, it remains corrupted from that point on.

   An additional model, called the proactive model, considers the possibility that parties are corrupted for a certain period of time only. Thus, honest parties may become corrupted throughout the computation (like in the adaptive adversarial model), but may later also become honest again.

   The corrupted parties can be divided into two main categories:

   - Semi-honest adversaries: in the semi-honest adversarial model, even corrupted parties correctly follow the protocol specification. However, the adversary obtains the internal state of all the corrupted parties (including the transcript of all the received messages), and attempts to use this to learn information that should remain private. This is a rather weak adversarial model. Semi-honest adversaries are also called "honest-but-curious" or "passive" adversaries.
   - Malicious adversaries: in this adversarial model, the corrupted parties can arbitrarily deviate from the protocol specifications, according to the adversary instructions. In general, providing security in the presence of malicious adversaries is preferred, as it ensures that no adversarial attack can succeed. Malicious adversaries are also called "active" adversaries.

2. **Complexity** Finally, we consider the assumed computational complexity of the adversary. As above, there are two categories here:

   - Polynomial-time: The adversary is allowed to run in (probabilistic) polynomial-time (and sometimes, expected polynomial-time). We remark that probabilistic polynomial-time is the standard notion of "feasibly" computation; any attack that cannot be carried out in polynomial-time is not a threat in real life.

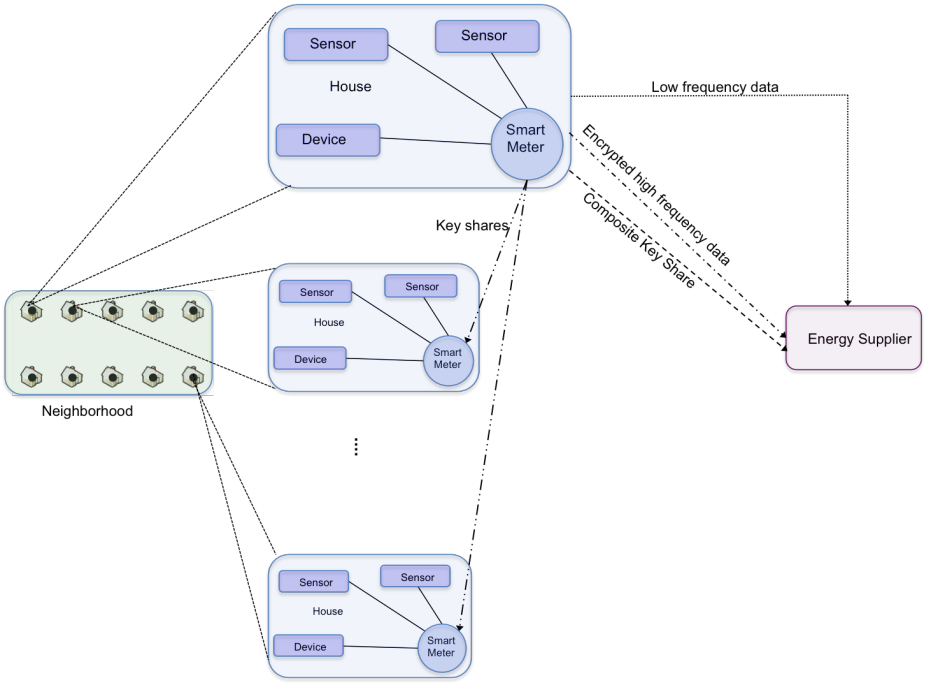

Figure 1 The architecture at-a-glance (operations performed by one smart meter are highlighted)

- Computationally unbounded: In this model, the adversary has no computational limits whatsoever.

One of the paradigms in SMPC uses secret sharing, which allows unconditionally secure protocols solving the multiparty computation problem.

4 The Privacy Preserving Architecture

In this section we detail the general architecture of the proposed system. It is important to highlight that, for sake of easiness, we describe here a simplified version of our proposal, which, yet complete from the working point of view, does provide full protection against malicious users. For the complete protocol we refer the reader to Section 5, where all the missing details will be provided.

Figure 1 shows the architecture's big picture, its main components and the data communication channels targeted by this work in a pretty common *smart grid* scenario. Several residential apartments (e.g. a large building), each of them equipped with a smart meter in charge of collecting and sending data either from/to nearby smart meters or from/to a higher level *energy supplier*, form a *neighborhood* or *group* that, together with nearby neighbourhoods are connected in a *smart grid*. More precisely:

- smart meters are responsible for collecting the measurements from the different sensors and devices distributed in the house about energy consumption and production both at low and high frequency. Each user must be provided with a smart meter.

Smart meters are not assumed to be trusted, since users can cheat on the energy consumption/production data.

- the energy supplier is responsible for billing, account management, demand side response, demand side management activities operations, as well as user profiling. The energy supplier, by definition, is assumed to be trusted.

Two types of data are typically exchanged within a *smart grid*:

- Low frequency data: these data, equivalent to the measurements collected by the meters of the "classical" electricity grids, are used for billing purposes only and are collected quite scarcely. From a privacy perspective they do not pose any serious concern (not allowing any profiling of the users) and thus they are directly sent by the smart meter to the energy supplier;
- high frequency data: these data are related to the instantaneous energy production and consumption and are used for user profiling purposes. As discussed in the introduction, they pose severe privacy concerns as may reveal information related to the activities of the user, and therefore cannot be directly sent to the energy supplier as plain data.

Smart meters send low frequency data directly to the energy supplier that, in turn, use such data for billing purposes. As this type of data exchange is not privacy sensitive, it does not need any refined protection mechanism.

High frequency data, instead, carry user sensitive content and, as such, they must be sent to the energy supplier as *per–neighborhood* aggregate by means of a secure (and complex) process in order to prevent the energy supplier (and each other smart meter alone) from deriving their *per–user* value.

By assuming that the number of smart meters of the neighbourhood is n and that all smart meters that belong to the same neighbourhood and energy supplier are authenticated within the same group, the scheme for the distribution of such data evolves according to the following procedure:

1. Each smart meter (say the ith smart meter) measures the instantaneous electricity consumption X_i and encrypts this value by means of its own private key k_i to obtain:

$$\hat{X}_i = E\left(X_i, k_i\right), \tag{4}$$

where $E(\cdot,\cdot)$ is a bi–homomorphic encryption scheme with respect to the sum operation.

2. Each smart meter has its own private key k_i. Such key is the *secret* and is not disclosed to any other smart meter; however, each smart meter splits the key k_i in n shares $k_{ij}, j = 1, 2, \ldots, n$ and sends the jth share to the jth smart meter according to a secret sharing scheme with $(+,+)$ homomorphic properties. As such, at this stage, every smart meter acts a the *dealer* of the secret sharing scheme.

3. The jth smart meter receives the jth portions of the keys of all other smart meters in the neighbourhood and computes a new composite key χ_j as the sum:

$$\chi_j = \sum_{i=1}^{n} k_{ij} \tag{5}$$

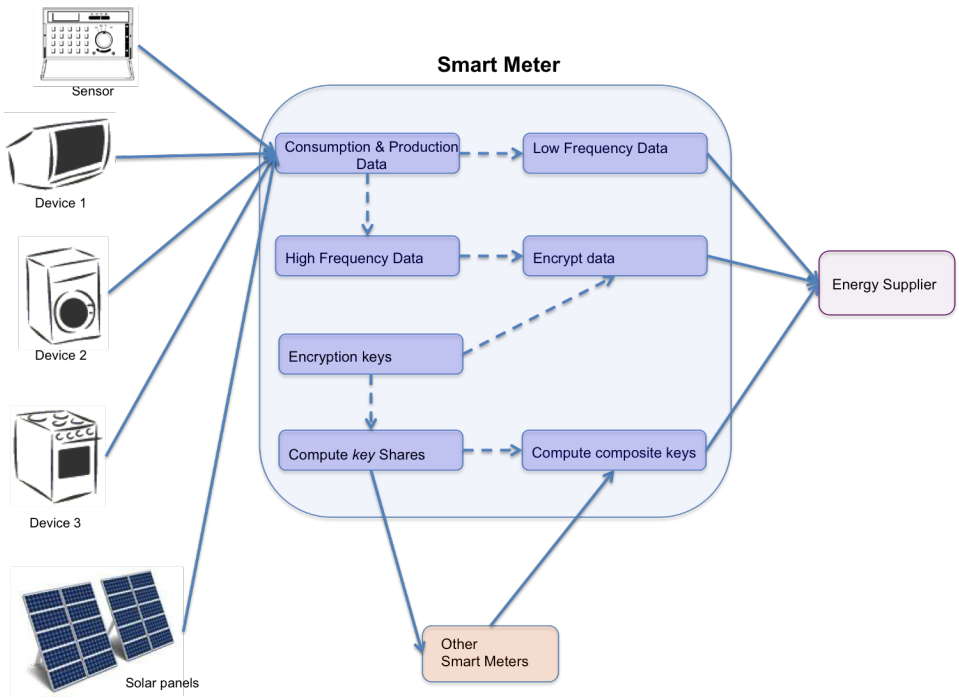

Figure 2 Smart meter operations (simplified scheme)

and sends the pair $\left(\hat{X}_j, \chi_j\right)$ to the energy supplier.

4. The homomorphic secret sharing scheme guarantees that the sum of shares equals the sum of secrets. Therefore, the energy supplier can compute the sum of all private keys of smart meters $k = \sum_{i=1}^{n} k_i$ from its n shares $\{ \chi_1, \chi_2, \ldots, \chi_n \}$.

5. Finally, thanks to the bi–homomorphic cryptography (in our proposal we have used Castelluccia's encryption algorithm [20]), the energy supplier decrypts the instantaneous aggregate consumption X of all smart meters as:

$$\begin{aligned} X &= \sum_{i=1}^{n} D\left(\hat{X}_i, k_i\right) \\ &= D\left(\sum_{i=1}^{n} \hat{X}_i, k\right) \end{aligned} \tag{6}$$

For the sake of clarity, in the above description the secret sharing scheme is applied to a single encryption key. In fact, Castelluccia's encryption algorithm requires the use of a distinct key at each encryption round. This constraint can be easily met by applying the SS scheme to a vector of Q pre-shared keys, hence avoiding the need for a data exchange at any transmission round. Therefore, k_i in the algorithm should be k_i^q, where q refers to the q^{th} transmission round. Obviously the SS scheme has to be repeated every Q rounds.

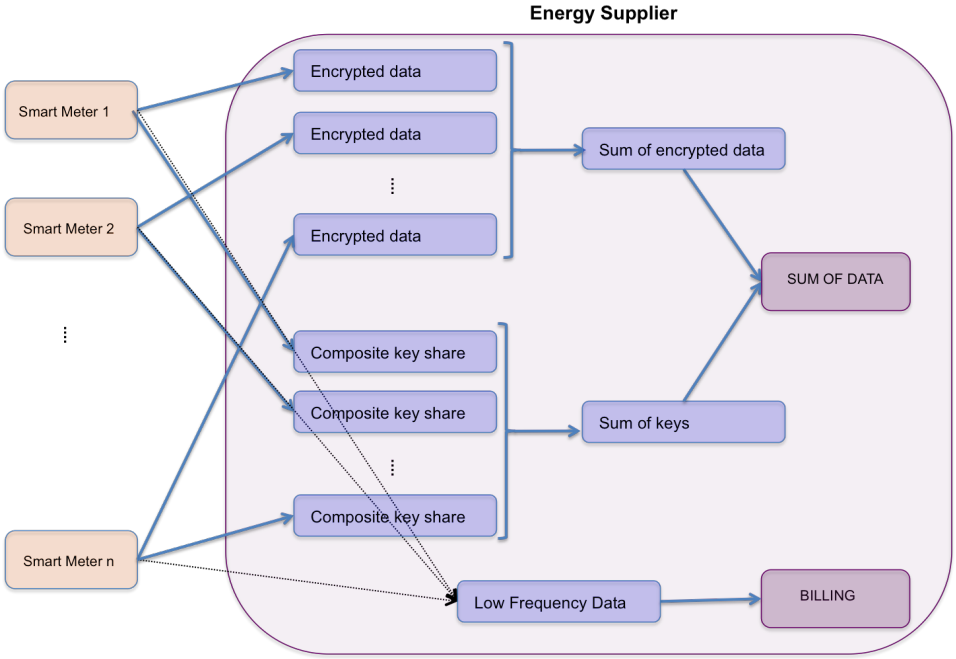

Figure 3 Energy supplier operations (simplified scheme)

Although reasonably intuitive, the proposed architecture consists of several functional stages that need to be further explored to provide an overall feasible scheme. In particular, the process for sharing the encryption keys has to be carried out in a more complex way in order for the system to be robust against cheating users, as discussed in Section 5.

5 Security Analysis

A crucial concern in the proposal of distributed architectures based on a SMPC (such as the one presented) is the possible malicious behavior of some adversarial entity, either an external one or a subset of the participating parties. In our case, possible attacks might be aimed at either disclosing private information or making the overall computation fail because of incorrect data.

The privacy requirement dictates that no extra information should be learnt other than that strictly necessary to perform the computation. The correctness requirement, instead, can be achieved as long as each party receives the correct data.

As already discussed, two main adversarial models can be identified: *semi–honest adversaries* (also known as *honest–but–curious* or *passive* adversaries) and *malicious adversaries* (also known as *active* adversaries).

Before entering in the security discussion of our proposal, it is worth reminding two main assumptions underlying the presented architecture: i) all of the involved players in the system (energy supplier and smart meters) require authentication before starting their communication and ii) the energy supplier is – by definition – assumed to be trusted, while smart meters are not.

Under the above hypotheses, in our architecture the main concerns may arise from two different cases: single smart meters corrupted by an external attacker and single users cheating either on the consumption measures or on the encrypted keys they share. According to the above introduced model, smart meters can be seen as semi–honest adversaries, as an attacker may try to obtain information from them even though they follow a correct behaviour. As an example, consider an attacker that at some time happens to obtain the information stored in k or more smart meters, either corrupting them or taking control of the machines, with the aim of disclosing the measurements data of the users. Such an attack, by itself, would not have effects as the attacker may be able to reconstruct the single keys only. Indeed, smart meters do not receive the high frequency values of energy consumption of users as they are directly sent to the energy supplier. To be effective, such an attack needs to be complemented by the ability of the attacker to "sniff" the information sent by each smart meter to the energy supplier.

A typical example of malicious adversary, instead, is that of a user that cheats on the measurement data in order to get an economical advantage (although in our case this specific type of data is carried through the low frequency communication channel).

This kind of attack cannot be fully dealt by the architecture as presented so far, and requires further processing with the adoption of ZKP techniques [6, 7], as detailed in the following subsection.

5.1 Cheating User

Let us detail the case in which the user cheats on the measurement data to have some economical revenue and can be seen as a malicious adversary. It is important to highlight that the user may want to cheat on both the low frequency and high frequency data, thus differently from what already described (for sake of simplicity), not only are the low frequency data sent directly to the energy supplier, but they also undergo a secret sharing procedure (as described in the following).

To face this kind of misbehaving, a few more operations on the data to be transmitted have to be carried out by every smart meter (as described in Figure 4, where a nearly complete version – with the exception of commitments – of Figure 2 is presented and in Figure 5, where the security related operations are depicted – the numbers in the circles are associated with the different steps of the algorithm).

Before describing the algorithm, it is worth noticing that, according to the "classical" SMPC terminology, each smart meter acts as a *privacy peer* for the other smart meters, hence in the following we will use the term privacy peers for indicating the $n-1$ smart meters other than the one sending the shares. Moreover, given that such a procedure is applied both to the encryption keys and to the low frequency data, in the following we will generically refer to both these data as secret data vector.

In order to protect the secret data vector d, each user generates a uniformly distributed random vector u and computes the vector $v = d - u$. Hence, she directly sends the vector u to the energy supplier, given that it does not include any information on the actual data, so no privacy concern arises from the possibility of assigning u to a specific smart meter. The vector v, instead, is divided into n different shares $v_i,\ i = 1, \ldots, n$ and each share is sent to a different privacy peer. Note that the privacy peers are not required to be honest and the proposed architecture assures that they do not have any information about the user vector d, given that the shares have been randomly generated and are of no use on their own.

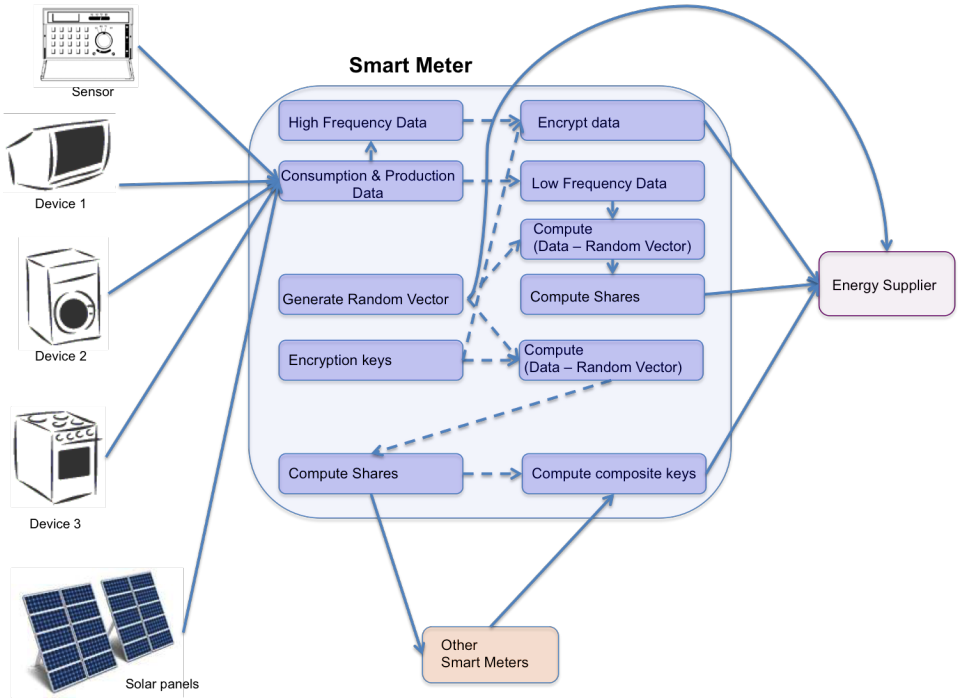

Figure 4 Smart meter operations (for a single smart meter)

At this point, the privacy peer i will sum the shares received from all the users to form a composite share V_i, which is then sent to the energy supplier.

On its side, the energy supplier at first computes the sum of the u vectors of the users, obtaining $\mathbb{U} = \sum_{users} u$. In parallel, it also computes the sum of the composite shares received from the different privacy peers, that is $\mathbb{V} = \sum_{users} V_i$. These two quantities are then summed up to obtain $\mathbb{D} = \mathbb{U} + \mathbb{V}$ that, given the homomorphic properties of the secret sharing scheme, is equivalent to the sum of the secret data vectors of the users, i.e., $\mathbb{D} = \mathbb{U} + \mathbb{V} = \sum_{users} d$. In practice, in this way, the energy supplier is able to reconstruct the sum of the secret data vectors of the users.

Given that, to provide the required protection against cheating users, each party provides ZKP that her input is valid, namely that the $L2$-norm of the secret data vector d is small (i.e., bounded by a constant L). It is worth noticing that the verification operation over the data can be easily changed.

More precisely the user computes a different proof for the energy supplier and for each other smart meter participating in the SMPC.

In our case, the smart meter is the prover, the energy supplier is the verifier and the data vector is the secret.

The ZKP works as follows: the energy supplier generates and broadcasts several challenge vectors c_k ($k = 1, \ldots, 50$ in our implementation), whose elements are drawn from $\{ -1, 0, 1 \}$ with probabilities $\{ .25, .5, .25 \}$; the purpose of the challenge vectors is to come up with a random vector so that the user can project his data vector onto it (via inner product). This projection is a random summary or sketch of the user data that encodes some statistical properties of the data. By computing a number of such projections (onto

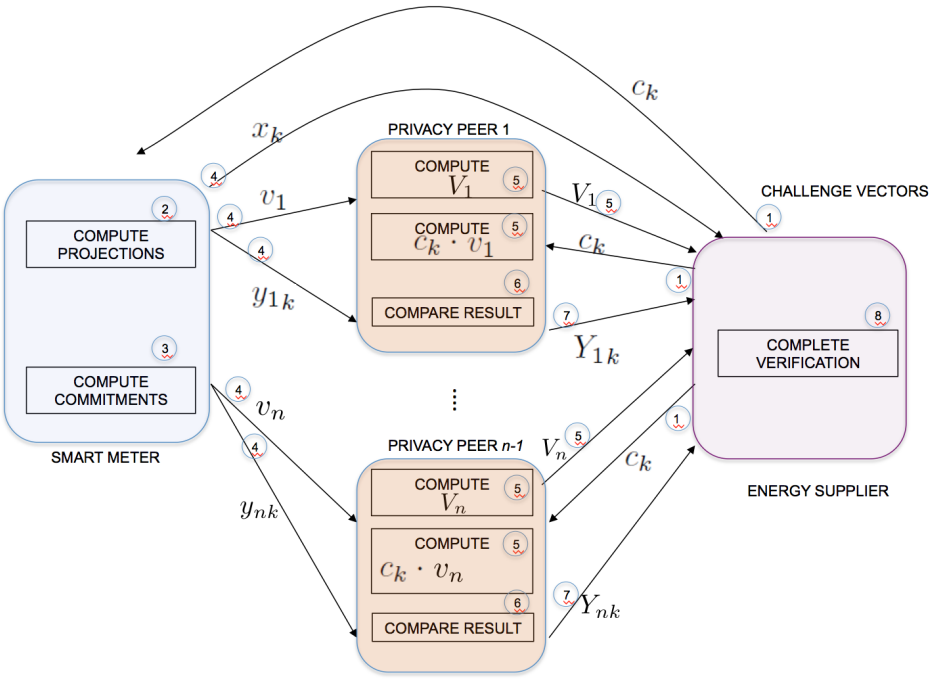

Figure 5 Privacy-related operations (for a single smart meter)

independently generated random directions), we can verify the user data' properties (in our case, the L2-norm). For each challenge vector, user computes projections of u onto c_k for the energy supplier (i.e., it computes $x_k = u \cdot c_k$), and of each single share of the vector v for the privacy peers (i.e., it computes $y_{ik} = v_i \cdot c_k$); in addition user also computes the sum of these projections and the sum of their squared values (note that, for sake of simplicity, these steps are not shown in Figure 5). For the purpose of verifying the user vector L2-norm, we do not have to check the sum of squares. We can just check the magnitude of each individual projection. However, this leaks more information than desired, the verifier can know on which direction the user vector is large or small. Checking the sum of squares is thus more secure, in the sense that the server only knows that the user vector is larger or smaller on some direction(s), which is equivalent to the fact that its L2-norm is larger or smaller than L, but it does not know on which direction. This conforms to the principle of ZKP: you only reveal information strictly necessary for proving the statement.

User then computes the Pedersen commitments to all these values and sends them altogether with the values to the privacy peers and the energy supplier.

In order to completely verify the proof, the energy supplier must first obtain the peers data and verification. The utility then combines the data with its own to form the complete proof. In this scenario, privacy peers only verify the commitments to their shares, by computing the commitments to the received shares and comparing them with the values sent by the users. Note that, actually, given that both the data and the verification are sent by the users, the peers verification always returns true (except in case of corrupted transmission). Indeed this step is introduced for checking the consistency of the transmission and because it is considered that the smart meter can only transmit the low frequency data to the energy supplier, directly.

At this point, the privacy peers forward the commitments to the utility which will perform the complete verification. If this proof succeeds, the user is accepted, while if any of these proofs fails the user is excluded from the set of qualified users. If all users result qualified, the utility will sum all their vectors u, and add the sum of the composite shares received from the privacy peers, obtaining $\mathbb{D}$.

If a user fails in any of these proofs, the utility will not include her contribute u in the computation, while the composite shares received from the privacy peers will also contain her contribute, given that privacy peers do not have any influence in the disqualification of the user. As a consequence, the sum of the valid vectors u and of the composite shares will not match and the verification test will fail. This mechanism assures that a user does not cheat and exert too much influence in the computation.

6 Conclusion

In this paper we presented a privacy preserving communication architecture for *smart grids* that relies on a Secure Multiparty Computation method based on Verifiable Secret Sharing of data.

The main innovation is the use of the encryption key employed by any smart meters as the secret information to be shared among all other smart meters that belong to the same group of authentication. In a nutshell, the idea underlying the proposed algorithm is that each user holds a "sub–secret", and that there exists a "super–secret", which is the sum of the sub-secrets. With an appropriate secret sharing homomorphism, shares of the sub–secrets can be distributed amongst a group of privacy peers, each of whom can then compose its "sub–shares" into a single "super–share" to be forwarded to the utility. In this way, revealing the super-shares determines the super-secret without sharing any information about the constituent sub-secrets.

Hence, this approach guarantees the privacy preservation of personal sensitive data, while allowing the energy supplier to retrieve aggregate measurements as well as individual coarse–grained energy consumption values for billing purposes. Moreover, this approach removes the need for any further intermediate aggregator and with beneficial effects in terms of the overall scalability of the system as well as in terms of its robustness.

As a future work, the authors will perform an in-depth analysis of the performance offered by the proposed system, so as to validate it and evaluate its suitability to "real-world" applications.

Acknowledgment

The authors would like Antonella Barletta for her activities in support of the presented work.

References

[1] C. W. Gellings. The smart grid: Enabling energy efficiency and demand response. *The Fairmont Press & CRC Press*, 2010.

[2] U.S.Department of Energy. The Smart Grid: An Introduction. "http://www.oe.energy.gov/SmartGridIntroduction.htm", 2008.

[3] E. T. P. SmartGrids. Strategic Deployment Document for Europe's Electricity Networks of the Future. "http://www.smartgrids.eu/documents/".

[4] E. L. Quinn. Privacy and the New Energy Infrastructure. *Center for Energy and Environmental Security (CEES), working paper*, 2008.

[5] National Institute of Standards and Technology. Guidelines for Smart Grid Cyber Security: Vol.2, Privacy and the Smart Grid. 2010.

[6] Christian Callegari, Sara De Pietro, Stefano Giordano, Michele Pagano, and Gregorio Procissi. A distributed privacy-aware architecture for communication in smart grids. In *HPCC/EUC*, pages 1622–1627, 2013.

[7] Christian Callegari, Sara De Pietro, Stefano Giordano, Michele Pagano, and Gregorio Procissi. Enforcing privacy in smart grid communications. *Rynek Energii*, 111:2:110–120, 2014.

[8] A. Barletta, C. Callegari, S. Giordano, M. Pagano, and G. Procissi. Privacy preserving smart grid communications by verifiable secret key sharing. In *2015 International Conference on Computing and Network Communications (CoCoNet)*, pages 199–204, Dec 2015.

[9] Yitao Duan, John Canny, and Justin Zhan. P4p: Practical large-scale privacy-preserving distributed computation robust against malicious users. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, pages 14–14, Berkeley, CA, USA, 2010. USENIX Association.

[10] Yitao Duan and John F. Canny. Practical private computation and zero-knowledge tools for privacy-preserving distributed data mining. In *SDM*, pages 265–276, 2008.

[11] Yitao Duan, John Canny, and J. Zhan. Efficient privacy-preserving association rule mining: P4p style. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*, pages 654–660, March 2007.

[12] Yitao Duan and John Canny. Practical private computation of vector addition-based functions. In *Proceedings of the Twenty-sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '07, pages 326–327, New York, NY, USA, 2007. ACM.

[13] C. Rottondi, G. Verticale, and A Capone. A security framework for smart metering with multiple data consumers. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 103–108, March 2012.

[14] Gergely Ács and Claude Castelluccia. I have a dream!: Differentially private smart metering. In *Proceedings of the 13th International Conference on Information Hiding*, IH'11, pages 118–132, Berlin, Heidelberg, 2011. Springer-Verlag.

[15] C. Efthymiou and G. Kalogridis. Smart grid privacy via anonymization of smart metering data. In *SmartGridComm*, pages 238–243, Oct 2010.

[16] J.M. Bohli, C Sorge, and O. Ugus. A privacy model for smart metering. *IEEE (ICC) 2010*, pages 1–5, 2010.

[17] F.D. Garcia and B. Jacobs. Privacy-friendly energy-metering via homomorphic encryption. *IEEE Security and Trust Management*, 6710:226–238, 2011.

[18] Fenjun Li, Bo Luo, and Peng Liu. Secure information aggregation for smart grids using homomorphic encryption. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 327–332, Oct 2010.

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

[20] C. Castelluccia and E. Mykletun. Efficient aggregation of encrypted data in wireless sensor network, 2005.

[21] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[22] George R. Blakley. Safeguarding Cryptographic Keys. In *Proceedings of the 1979 AFIPS National Computer Conference*, volume 48, pages 313–317, June 1979.

[23] C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Trans. Inf. Theor.*, 29(2):208–210, September 2006.

[24] Ivan Damgård. Commitment schemes and zero-knowledge protocols. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*, pages 63–86, London, UK, UK, 1999. Springer-Verlag.

[25] Ronald Cramer. Introduction to secure computation. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*, pages 16–62, London, UK, UK, 1999.

[26] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '91, pages 129–140, London, UK, UK, 1992. Springer-Verlag.

[27] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 136–, Washington, DC, USA, 2001. IEEE Computer Society.

[28] Y Lindell and B. Pinkas. Secure Multiparty Computation for Privacy-Preserving Data Mining. *IACR Cryptology ePrint Archive*, 2008.