6-16-2017

Smart Grid Privacy through Distributed Trust

Benjamin Lipton
bgl9668@rit.edu

Smart Grid Privacy through Distributed Trust

Benjamin Lipton

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science in Computing Security

Rochester Institute of Technology

B. Thomas Golisano College of Computing & Information Sciences

Department of Computing Security

June 16, 2017

Rochester Institute of Technology

B. Thomas Golisano College
of
Computing and Information Sciences

Master of Science in

Computing Security

Thesis Approval Form

Student Name:	Benjamin Lipton
Thesis Title:	Smart Grid Privacy through Distributed Trust

Thesis Committee

Name	Signature	Date
Sumita Mishra		
Chair		
Stanisław Radziszowski		
Committee Member		
Matthew Wright		
Committee Member		

Contents

Abstract

Though the smart electrical grid promises many advantages in efficiency and reliability, the risks to consumer privacy have impeded its deployment. Researchers have proposed protecting privacy by aggregating user data before it reaches the utility, using techniques of homomorphic encryption to prevent exposure of unaggregated values. However, such schemes generally require users to trust in the correct operation of a single aggregation server. We propose two alternative systems based on secret sharing techniques that distribute this trust among multiple service providers, protecting user privacy against a misbehaving server. We also provide an extensive evaluation of the systems considered, comparing their robustness to privacy compromise, error handling, computational performance, and data transmission costs. We conclude that while all the systems should be computationally feasible on smart meters, the two methods based on secret sharing require much less computation while also providing better protection against corrupted aggregators. Building systems using these techniques could help defend the privacy of electricity customers, as well as customers of other utilities as they move to a more data-driven architecture.

1 Introduction

1.1 Background

Today's electrical grid is based on a primarily one-way relationship between electricity producers and consumers. Electricity is produced centrally and then distributed outward to consumers. Producers receive feedback from consumers only via a monthly meter reading, or a phone call if a serious problem occurs. The proposed next-generation electrical grid, or Smart Grid, will greatly increase the flow of data from consumers back to the utility. Smart Meters report electricity usage much more frequently, on the order of minutes. This quick feedback allows the utility to run the grid more efficiently and handle problems more quickly. However, this increase in data collection has created privacy concerns, in some cases resulting in rollouts of smart metering being delayed or canceled. We aim to help provide electric utilities with the data they need without compromising the privacy of the users, helping to alleviate this roadblock to adoption of the smart grid.

An important component of any privacy-preserving scheme is managing trust. In a fully trusted system, there would be no need for cryptography. In other cases, a system may use cryptographic techniques to provide confidentiality and integrity that do not depend on trusting the system operator. There are situations where the cryptography satisfying the precise requirements of the scenario is not yet practical, however. Smart metering appears to be one such scenario, as researchers have struggled to find a cryptographic primitive that would allow a group of smart meters to release to the utility an aggregate of their electricity usage:

- without revealing their individual, high frequency consumption data to anyone,
- without communication between meters,
- without requiring much more powerful meter hardware.

Still, cryptography does provide many interesting tools that could aid this effort. Homomorphic encryption, which allows mathematical operations to be computed on encrypted data to produce an encrypted result, has already been used by many researchers in smart grid privacy [3, 5, 20, 26, 29]. Other types of cryptography such as functional encryption [7] and general secure multi-party computation (e.g. [23]) would also be applicable, but so far they do not seem to be computationally feasible for low-power devices. On the other hand, secret sharing [4] is a promising, and less expensive, technique for controlling data access in the smart grid [3, 9, 35]. These tools cannot eliminate the need for trust, but they can certainly provide more options for securely dividing responsibilities in the system.

Since we can not replace trust completely, this thesis is an investigation of how we can improve the trustworthiness of those components that must be trusted. We can make it less likely, or rewarding, for a system component to break that trust. We base our work on the basic principle that a consipracy becomes less likely the more participants are required, and thus a group of independent parties is generally more trustworthy than a single party. "Distributing trust" in this manner makes the system more trustworthy because the incentive for an individual party to misbehave is reduced. This thesis presents and evaluates several practical technqiues that use cryptography to enforce the distribution of trust, protecting user privacy.

1.2 Contribution

The contribution of this work is a detailed evaluation of three different systems for privacy protection in the smart grid. The systems themselves are presented, or at least hinted at, by prior work; however, there has been no research directly comparing them. In particular, we implement all of the systems and test their feasibility on real low-power hardware, and we provide theoretical results comparing their resilience to privacy and integrity attacks, as well as their costs in terms of computation and network traffic overhead.

1.3 Organization

The remainder of this report is organized as follows: Section 2 contains a broad review of the relevant literature, with additional detail in Section 3 about specific concepts used in our work. Section 4 describes the high-level components and requirements of the systems we will present, and Section 5 presents the systems in detail. Section 6 describes the details of our software implementation of the systems. Section 7 presents results evaluating the systems on several metrics, and Section 8 discusses these results as well as presenting points of potential future work. Finally, we draw some conclusions in Section 9.

2 Related Work

2.1 Smart Metering Privacy

Though electricity usage data may not seem terribly sensitive, people are right to be concerned about its release. Research has shown that the high-frequency usage data can reveal the number of people in a home, and their patterns of activity [31]. Further, if the data are collected at sufficiently high frequency, it is even possible to determine the TV channel being watched in the home [22]. These are types of information that most people do not wish to share with their electric company, and that could be damaging if released, for instance in a security breach of the utility.

2.2 Privacy via Anonymity

Many researchers have worked to alleviate smart metering privacy concerns. One approach they have taken is to anonymize the data sent to the utility. Efthymiou and Kalogridis [16] propose a system in which the meter has two built-in identifiers, one (the LFID) which is used to identify it when transmitting low-frequency data for billing purposes, and one (the HFID) which identifies it when transmitting high-frequency data. The HFID is cryptographically authenticated via an escrow service that knows valid identifier pairs, so that the utility can trust data sent with that HFID but not know to which customer it belongs. A related approach is to store all data with a trusted third party (TTP), under an anonymous identifier issued by the utility [18]. In this case, neither the utility nor the TTP has enough information to reconstruct a particular customer's usage data. This scheme is highly vulnerable to deviations from the protocol by either party, however. The utility can maliciously change a user's password at the TTP and retrieve all their data, or the TTP could simply decide to share more fine-grained data with the utility than it is supposed to. It is this type of concern we aim to avoid by distributing trust.

2.3 Privacy via Aggregation

The most common alternative to anonymizing the fine-grained data is to provide only aggregated data to the utility, as outlined in Section 3.1. Kursawe et al. propose one example of aggregation in which smart meters in an area communicate to generate a set of random values that sum to zero [28]. Each smart meter adds its random value to the measurement it sends to the utility. This is a simple way to ensure that the individual readings are meaningless, but become useful under spatial aggregation. However, having smart meters communicate with one another introduces significant challenges, both because it complicates the network architecture of the grid and because customers must now trust other smart meters as well as their own to protect their privacy.

Homomorphic encryption can allow us to keep individual measurements encrypted while still allowing aggregation. Several researchers have taken advantage of this by having smart meters communicate encrypted results to one another so that they can be aggregated before they are transmitted to the utility [29, 20]. However, meter-to-meter communication is problematic for the reasons just mentioned. Another proposal replaces meter-to-meter communication with a second meter-to-utility round trip in which the meter decrypts data already partially aggregated by the utility [20]. This adds additional latency but simplifies the network.

Keoh et al. perform aggregation without this additional round trip by sending encrypted measurements to a concentrator node, which aggregates them homomorphically and sends the sum to the utility [26]. However, this preserves the privacy of the measurements only if the concentrator can be trusted to adhere to the protocol. Callegari et al. [9] propose a similar scheme but distribute the trust in the aggregator; they use multiple aggregators so that misbehavior of one does not compromise the whole system, using the secret sharing and zero-knowledge proof techniques of P4P [14, 15] to prevent cheating. Although this approach provides some assurances even against a very strong attacker, that paper fails to provide detail about what those assurances are, or results on how the system performs in practice.

Another alternative is to use a variant of homomorphic encryption in which the key changes along with the plaintext as homomorphic operations are applied. Barletta et al. call this type of cryptosystem *bi-homomorphic encryption* and use it to ensure the utility can only decrypt aggregated data points [3]. In their proposal, meters generate individual keys, but then collaborate with other meters to generate the sum of all the individual keys and share it with the utility. Then, when the utility receives the measurements from the meters, encrypted with the individual keys, it must aggregate them to be able to decrypt with the combined key it has. Biselli et al. take a similar approach, except that groups of keys are aggregated by a trusted gateway admin, rather than by the meters themselves [5].

2.4 Homomorphic Encryption

In addition to the additive homomorphic properties of the Paillier cryptosystem [33], as described in Section 3.2, there are designs such as the BV cryptosystem [8] that allow both addition and multiplication of encrypted values. However, so far these systems allow only a limited number of consecutive multiplications before a costly "bootstrapping" operation is required to prevent noise from overwhelming the signal. This requirement has given homomorphic encryption the reputation of being computationally infeasible. However, Naehrig et al. show that it is possible to compute a variety of useful functions homomorphically, as long as the number of multiplications is known in advance [32]. It is not clear that such advanced homomorphic encryption schemes are required for smart metering, though. Addition of encrypted values and multiplication by constants seem to be sufficient for our temporal and spatial aggregation requirements.

Even Paillier encryption could be challenging for a smart meter, as these are low-power, low-memory devices, so optimizations to make it less resource-intensive are valuable. Damgård et al. propose an alternate algorithm with precomputation that makes encryption four times faster than the original, as well as several variants of the cryptosystem that are useful under other circumstances like electronic voting [12]. Jost et al. present performance results for several further optimizations, including precomputing values that can be used to speed up the two exponentiations that happen during encryption [24]. Neither of these papers mentions the feasibility of running the algorithm on resource-constrained devices, however.

2.5 Secret Sharing

A different way of protecting secrets from discovery while they are operated upon is to divide them into shares, such that a threshold number of shares are required to reconstruct the secret. Although a simple secret-sharing scheme can be derived from addition as described in Section 5.3, the more interesting forms of secret sharing are those with a threshold; in these schemes, any k of the shares can reconstruct the secret, while fewer shares leave it completely undetermined. Shamir [36] and Blakley [6] presented two such systems, based on polynomial interpolation and the intersection of planes in vector spaces, respectively. Asmuth and Bloom propose a system based on the Chinese Remainder Theorem, which they claim has better asymptotic complexity for secret reconstruction [1]. Both Karnin et al. [25] and Kothari [27] present generalized schemes that include the Shamir and Blakley schemes as special cases.

Several proposals have addressed how to protect secret sharing schemes against malicious participants. Many researchers, including Feldman [17] and Pedersen [34] have investigated a category of system called *verifiable secret sharing*, in which the recipient of a share can verify that it is a share of the correct secret. This protects against corruption of a share by a party that handles it, such as the aggregator in our system. Further, Benaloh [4] notes that many popular secret sharing schemes have homomorphic properties, and shows how these schemes can be made verifiable as well. We make use of the homomorphic properties of Shamir's scheme in our work. Taking verifiability a step further, P4P [15] and Prio [11] show how secret sharing schemes can be augmented with zero-knowledge proofs to place constraints on the shared values themselves. In the smart grid, this could be used to make sure that the reported measurements are within a reasonable range without revealing the exact measurements.

Other researchers have proposed using secret sharing to implement privacy preserving solutions for the smart grid. For example, Rottondi et al. [35] use Shamir's scheme to implement secure aggregation. Like ours, this paper also considers the effect of dropped shares on the recoverability of aggregate values, though they model errors using a probability of dropped packets rather than studying the effect of an absolute number dropped. Further, the Privacy Preserving Nodes (PPN) in this paper, which have the same role as our aggregators, are all trusted; there is no discussion of making privacy resilient to corrupted PPNs.

2.6 Other Cryptographic Tools

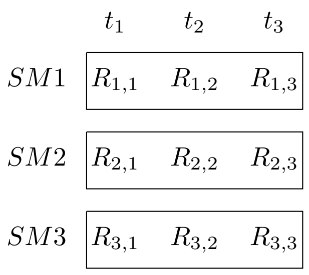
In addition to those already discussed, there are other forms of cryptography that allow greater control over the release of data than the cryptosystems in common usage. One of them is functional encryption [7], in which keys can be issued that compute specific functions over encrypted data. The raw data is still hidden; only the function output is revealed. The function of interest in the smart grid is an aggregation of inputs from different sources, so the work of Goldwasser et al. extending functional encryption to functions of multiple inputs might be necessary to make the smart grid fit into this model. [21]. However, their techniques have not yet been shown to be practical. Another cryptographic model for this problem is multi-party computation, which describes algorithms that enable a function to be computed from secret inputs held by multiple parties. To achieve our goal of not requiring communication between meters, a multi-party computation algorithm would need to be non-interactive like the one presented by Ishai et al. [23]. It would also need to be able to handle a large number of parties (all the smart meters in an area) and be efficient enough to work within the meters' computational power.

3 Building Blocks

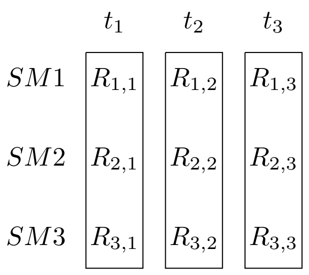
This section provides a more detailed presentation of some of the aspects of prior work that will be used directly in this thesis.

3.1 Aggregation

The primary goal in smart metering privacy is to protect the high-frequency data of individual users from disclosure. One way to do this is to provide only aggregated data to the utility. Two forms of aggregation are of interest: *temporal aggregation* computes the total electricity usage over a large time interval, as would be needed to produce a monthly bill, while *spatial aggregation* computes the total electricity usage over a group of smart meters in a much shorter time interval, which is needed to track demand in the grid and respond to problems such as power line failures or overloaded circuits. Figure 1 shows the difference between these two types.


t_1 t_2 t_3
$SM1$ $R_{1,1}$ $R_{1,2}$ $R_{1,3}$
$SM2$ $R_{2,1}$ $R_{2,2}$ $R_{2,3}$
$SM3$ $R_{3,1}$ $R_{3,2}$ $R_{3,3}$

(a) Temporal aggregation for billing.


t_1 t_2 t_3
$SM1$ $R_{1,1}$ $R_{1,2}$ $R_{1,3}$
$SM2$ $R_{2,1}$ $R_{2,2}$ $R_{2,3}$
$SM3$ $R_{3,1}$ $R_{3,2}$ $R_{3,3}$

(b) Spatial aggregation for grid maintenance.

Figure 1: Types of Aggregation. The symbol $R_{m,t}$ denotes the reading from smart meter m at time interval t.

3.2 Paillier Encryption

The Paillier cryptosystem [33] is an asymmetric cryptosystem with additive homomorphic properties. That is, if $E(m_1)$ and $E(m_2)$ are encryptions of messages m_1 and m_2 with the same public key, it is always true that

$$D(E(m_1) \cdot E(m_2) \bmod n^2) = m_1 + m_2.$$

It is also possible to multiply the encrypted value by a plaintext value k:

$$D(E(m)^k \bmod n^2) = km.$$

This additive property is extremely useful for aggregation, because it allows the summation of several encrypted data points, taken at different times or from different sources, to be computed by an aggregator with no knowledge of the private key.

The cryptosystem is set up by selecting prime numbers p and q. Then the private key is

$$\lambda = \text{lcm}(p-1, q-1),$$

and the public key is

$$(n, g),$$

where $n = pq$ and g is an element of $\mathbb{Z}^*_{n^2}$ chosen such that

$$\gcd(L(g^\lambda \bmod n^2), n) = 1.$$

(In this expression, L is the function $L(x) = \frac{x-1}{n}$, defined when $x \equiv 1 \bmod n$.)

As in RSA, the security of the key depends on the secrecy of the factors p and q. The NIST recommendations for RSA [2], which suggest that n should be at least 2048 bits long, should thus provide a minimum key size for Paillier as well.

Encryption is achieved by raising g to the power of the message and blinding with a random number r:

$$c = E(m) = g^m \cdot r^n \bmod n^2.$$

Decryption uses exponentiation by the private key and division within the modular space $\mathbb{Z}_p$:

$$m = D(c) = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n.$$

3.3 Shamir's Secret Sharing Scheme

Shamir's scheme [36] is the most popular example of a *threshold secret sharing scheme*, which allows a secret value D to be divided into n pieces $D_1, \ldots, D_n$, such that the original value can easily be computed from any k of the pieces, but any $k-1$ pieces provide no information about the secret. Such a scheme is useful to our scenario because it allows measurements to be shared securely from a meter to a group of aggregators. The measurements are protected from being revealed by corrupted aggregators, but also robust against a certain number of shares being lost.

The scheme relies on the fact that for any k points in the 2-dimensional plane with distinct x-coordinates, there is exactly one polynomial of degree $k-1$ that contains all k points. Thus if a polynomial $q(x)$ is chosen, and n points on that polynomial are distributed as shares of the secret, any k of those points uniquely determine $q(x)$.

All computations take place in a prime field $\mathbb{Z}_p$, where $p > D$. To share a secret D, we choose a polynomial

$$q(x) = D + a_1 x + \cdots + a_{k-1} x^{k-1}$$

by selecting all of the coefficients a_i randomly from $[0, p)$. That is, $q(x)$ is a random polynomial such that $q(0) = D$. The ith share is then computed as

$$D_i = q(i) \bmod p.$$

Each share is stored as a point $(x, q(x))$. To combine k shares, a polynomial interpolation algorithm is used to compute the value of $q(0)$ from the specified points, revealing the original secret.

The polynomial of least degree that passes through the set of points

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

is called the Lagrange interpolation polynomial, and an expression for it is given by

$$\begin{aligned} P(x) = & \frac{(x-x_2)(x-x_3)\cdots(x-x_N)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_N)} y_1 + \frac{(x-x_1)(x-x_3)\cdots(x-x_N)}{(x_2-x_1)(x_2-x_3)\cdots(x_1-x_N)} y_2 \\ & + \cdots + \frac{(x-x_1)(x-x_2)\cdots(x-x_{N-1})}{(x_N-x_1)(x_N-x_2)\cdots(x_N-x_{N-1})} y_N. \end{aligned} \tag{1}$$

(Note that all divisions are in the modular space $\mathbb{Z}_p$.)

Although there are more sophisticated algorithms for polynomial interpolation, the secret can be reconstructed simply by evaluating Equation 1 at $x = 0$ using the points given by the available shares.

4 System and Threat Model

4.1 System Overview

All the systems we analyze consist of the following components:

- One *utility*, the consumer of the data,
- N_a *aggregators*, operated by other utility companies independent from the data consumer,
- N_m *smart meters* within a geographical region, whose output is to be aggregated.

Each smart meter distributes its reading, or shares of its reading, among some of the aggregators. Each aggregator computes an aggregate value and forwards it to the utility, which should then be able to compute the sum of the reported meter readings. This flow is illustrated in Figure 2.

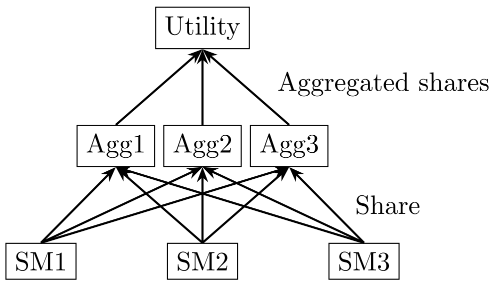


Figure 2: Generalized System Architecture.

In our analysis, we will assume a system with 10 smart meters and 10 aggregators. However, the number of smart meters in a neighborhood in a real-world system would be tens or hundreds of times larger. Even with many more smart meters, the number of aggregators should not need to be increased unless more privacy protection is desired.

4.2 Threat Model

We will evaluate the success of each system at defending against the following threats:

1. Privacy Attacker: An adversary curious about users' fine-grained data, who can gain read-only access to the data held by the utility and some of the aggregators. We denote the number of aggregators thus corrupted as N_c.

2. Integrity Attacker: An adversary able to interrupt communication between a meter and one or more of the aggregators. This could be an effect of network conditions as well as an active attacker. We assume that established network security techniques (e.g. SSL) are able to protect these links from other types of eavesdropping and modification, however.

Since smart meters are generally sealed devices with very good physical security, we assume that they act according to the protocol, except for any network-layer interruptions an attacker could introduce.

The utility and the aggregators are operated by companies with a vested interest in the correct operation of the electric grid, so we assume they value correct data more than invading user privacy. Thus, even though we assume that a coalition of insiders (the privacy attacker) may collect data from all but one of the aggregators, the aggregators do not otherwise deviate from the protocol, nor do they cooperate with the integrity attacker whose goal is to disrupt the system.

5 System Design

All of the systems considered share a similar structure, as described in Section 4.1. In this section we first define notation that will be used to describe the intermediate results within all the systems, then present the systems themselves.

5.1 Notation

We denote the reading at time interval t from meter m as $R_{m,t}$. We are interested in the spatial aggregation of these values over the set M of all meters in a geographic area

$$A_{spatial}(t, M) = \sum_{m \in M} R_{m,t},$$

and the temporal aggregation over a billing period T

$$A_{temporal}(m,T) = \sum_{t \in T} R_{m,t}.$$

In the protocols based on secret sharing, readings are first split into shares, each of which is forwarded to a different aggregator. We denote the share of $R_{m,t}$ destined for the aggregator with identifier s as $r_{m,t,s}$.

Aggregator s receives share number s from some or all of the smart meters. Over time, it may receive multiple shares from the same meter. The aggregator can perform two different aggregations over these shares—Spatial:

$$a_{spatial,s}(t,M) = \text{Aggregate}(\{r_{m,t,s} : m \in M\})$$

or temporal:

$$a_{temporal,s}(m,T) = \text{Aggregate}(\{r_{m,t,s} : t \in T\}).$$

Finally, once all the aggregators send their aggregated shares to the utility, the utility can perform a reconstruction operation to compute the final aggregate:

$$A_{spatial}(t,M) = \text{Reconstruct}(\{a_{spatial,s}(t,M) : 1 \le s \le N_a\}) = \sum_{m \in M} R_{m,t},$$
$$A_{temporal}(m,T) = \text{Reconstruct}(\{a_{temporal,s}(m,T) : 1 \le s \le N_a\}) = \sum_{t \in T} R_{m,t}.$$

The "Aggregate" and "Reconstruct" operators differ depending on the system under consideration.

When defining our systems we use the symbols R_{min} and R_{max} to denote the smallest and largest possible meter readings supported by the system, and A_{max} to refer to the largest possible value of the aggregate measurement.

5.2 Baseline System: Single Aggregator with Homomorphic Encryption

As a baseline, we use a simplified version of the "spatial aggregation" design presented by Keoh et al. [26]. This system works by collecting encrypted values at a single aggregator ("concentrator") node. As such, there is no distribution of trust in this system. The values are encrypted using an additively homomorphic cryptosystem, such as Paillier encryption [33]. Using the homomorphic properties of the cryptosystem, the aggregator computes the encrypted sum of the measurements and sends this aggregate to the utility.

Specifically, the aggregator receives $E(R_{m,t})$ for all meters m and time periods t. Since Paillier encryption has the property that

$$D(E(m_1) \cdot E(m_2) \bmod n^2) = m_1 + m_2,$$

the aggregations can be performed as:

$$E(A_{spatial}(t, M)) = \prod_{m \in M} E(R_{m,t})$$

$$E(A_{temporal}(m, T)) = \prod_{t \in T} E(R_{m,t}).$$

When these values are transmitted to the utility, it uses its private key to decrypt the aggregates.

5.2.1 Security

The security of the Paillier crytptosystem depends on a conjecture Paillier calls the Computational Composite Residuosity Assumption (CCRA). The CCRA posits that it is computationally difficult to determine, given a generator g and number w, the unique value $x \in \mathbb{Z}_n$ for which there exists $y \in \mathbb{Z}_n^*$ such that $g^x \cdot y^n \bmod n^2 = w$. Paillier shows that the ability to factor n allows one to solve this problem easily, so the security of this scheme also depends upon the hardness of factoring. Therefore, the key sizes used for the Paillier cryptosystem should be at least as large as those recommended for other cryptosystems based on factoring, such as RSA, as long as such large keys are feasible for the limited smart meter hardware on which the algorithms much run.

5.2.2 Example

We begin by generating a small Paillier keypair. We use the primes $p = 5$ and $q = 7$. Then

$$\begin{aligned} \lambda &= \mathrm{lcm}(4, 6) = 12 \\ n &= 5 \cdot 7 = 35 \\ g &= n + 1 = 36. \end{aligned}$$

We confirm that $\gcd(L(g^\lambda \bmod n^2), n) = 1$:

$$\begin{aligned} \gcd(L(g^\lambda \bmod n^2), n) &= \gcd(L(36^{12} \bmod 1225), 35) \\ &= \gcd(L(421), 35) \\ &= \gcd(\frac{421 - 1}{35}, 35) \\ &= \gcd(12, 35) \\ &= 1. \end{aligned}$$

The public key, $(35, 36)$, is distributed to the smart meters; the private key, 12, is retained by the utility.

Now suppose that two meters need to encrypt the readings $R_{1,1} = 2$ and $R_{2,1} = 4$. Each chooses a random r and computes $E(m) = g^m \cdot r^n \bmod n^2$:

$$\begin{aligned} E(R_{1,1}) &= 36^2 \cdot 8^{35} \bmod 1225 = 22 \\ E(R_{2,1}) &= 36^4 \cdot 19^{35} \bmod 1225 = 59. \end{aligned}$$

The aggregator computes the aggregate:

$$E(A_{spatial}(1, \{1,2\})) = 22 \cdot 59 \bmod 1225 = 73.$$

And finally the utility performs decryption:

$$\begin{aligned} A_{spatial}(1, \{1,2\}) = D(73) &= \frac{L(73^{\lambda} \bmod n^2)}{L(g^{\lambda} \bmod n^2)} \bmod n \\ &= \frac{L(73^{12} \bmod 1225)}{12} \bmod 35 \\ &= L(71) \cdot 12^{-1} \bmod 35 \\ &= 2 \cdot 3 \bmod 35 \\ &= 6. \end{aligned}$$

As we hoped, the result is sum of the two original measurements.

5.3 Probabilistic System: Random Aggregator Selection

In the previous system, users must trust the aggregator to keep the unaggregated data secret. Instead, we can distribute the trust over several aggregators. Each receives only a share of each reading; several shares are required to retrieve the original value. As a bonus, we no longer require homomorphic encryption to keep the data protected from the aggregator, as a single share conveys no information about the secret.

In this design, we use a simple addition-based secret sharing scheme. Shares are elements of the additive group $\mathbb{Z}_{A_{max}}$, chosen so that the shares of a secret sum to that secret within the group.

To divide up a secret $R_{m,t}$ into n shares, the smart meter selects uniform random values for the first $n-1$ of them:

$$\forall s \in [1, n-1] : r_{m,t,s} \leftarrow \mathbb{Z}_{A_{max}}.$$

It chooses the remaining share so that the sum of all shares is $R_{m,t}$, thus

$$r_{m,t,n} = \left(R_{m,t} - \sum_{s=1}^{n-1} r_{m,t,s} \right) \bmod A_{max}.$$

To maximize privacy, we can choose $n = N_a$, so that shares from all aggregators are required to reconstruct an individual reading. In this case, if even one aggregator is honest, user privacy is preserved. However, we can reduce the overhead of the system with only a limited impact on privacy by reducing n to a lower value. The smart meter then randomly selects n of the aggregators and sends shares only to them. Now the privacy protection becomes probabilistic; a corrupted subset of the aggregators may compromise the privacy of some users, but not all, and not the same ones on every reading.

Each aggregator now has some of the shares appropriate to a given aggregation operation. Let us denote by P_s the set of pairs (m, t) such that the meter m sent a share to aggregator s at time t. The aggregator sums all shares it has that are relevant to the requested aggregation operation:

$$a_{spatial,s}(t, M) = \sum_{m \in M | (m,t) \in P_s} r_{m,t,s}$$

$$a_{temporal,s}(m, T) = \sum_{t \in T | (m,t) \in P_s} r_{m,t,s}.$$

Once the utility receives these aggregates, it can compute the final aggregation by again taking the sum:

$$A_{spatial}(t, M) = \sum_{s=1}^{N_a} a_{spatial,s}(t, M) = \sum_{m \in M} R_{m,t}$$

$$A_{temporal}(m, T) = \sum_{s=1}^{N_a} a_{temporal,s}(m, T) = \sum_{t \in T} R_{m,t}.$$

5.3.1 Security

Since $n - 1$ of the shares of each measurement are randomly generated, it is clear that posession of only those shares gives an attacker no information about the measurement. However, it also the case that any group of $n-1$ shares including the deterministically computed share reveals no information about the secret. Let r be the share that is missing, and R' the sum of the remaining shares, so $R = R' + r$. Given R', each possible value for r would produce a different value for the real sum R, and there are $|\mathbb{Z}_{A_{max}}|$ possible values for each, so there is a bijection between r and R. We know that r was selected uniformly from $\mathbb{Z}_{A_{max}}$, so any group element is equally likely. Therefore, any value for R is equally likely as well, even though R' is given. The remaining shares therefore communicate no information about the real value of R.

The security of this system is information-theoretic, rather than cryptographic, meaning that no amount of computational power could determine the secret without the necessary number of shares. Thus the size of A_{max} is not chosen for security reasons, but to ensure that the true sum of the measurements can be reconstructed without overflow.

5.3.2 Example

For our example, we assume that $A_{max} = 100$ and $n = 3$. Suppose that two meters produce the readings $R_{1,1} = 17$ and $R_{2,1} = 6$. Each meter selects its first two shares randomly:

$$\begin{aligned} r_{1,1,1} &= 5 \\ r_{1,1,2} &= 58 \\ r_{2,1,1} &= 93 \\ r_{2,1,2} &= 29. \end{aligned}$$

The remaining shares are chosen so each meter's shares add up to its reading. Note that all arithmetic is performed modulo A_{max}.

$$\begin{aligned} r_{1,1,3} &= 17 - 5 - 58 = 54 \\ r_{2,1,3} &= 6 - 93 - 29 = 84. \end{aligned}$$

Each aggregator receives a share from each meter and performs spatial aggregation:

$$a_{spatial,1}(1,\{1,2\}) = r_{1,1,1} + r_{2,1,1} = 98$$
$$a_{spatial,2}(1,\{1,2\}) = r_{1,1,2} + r_{2,1,2} = 87$$
$$a_{spatial,3}(1,\{1,2\}) = r_{1,1,3} + r_{2,1,3} = 38.$$

The utility sums these aggregates to compute the final value:

$$A_{spatial}(1,\{1,2\}) = 98 + 87 + 38 = 23.$$

As was desired, the final aggregate is the sum of the original readings, 17 and 6.

5.4 Threshold System: Threshold Secret Sharing [36]

A disadvantage of the previous scheme is its low resistance to integrity attacks. If any share is disrupted by the integrity attacker, it can completely prevent the system from reconstructing a value. This problem can be alleviated by using threshold secret sharing, such as Shamir's scheme [36].

In a threshold secret sharing scheme, N_a shares are generated in such a way that any k of them can reconstruct the secret, but any $k-1$ of them do not contain enough information to reconstruct it. These shares are distributed to all the aggregators. This system provides higher reliability; $N_a - k$ of a meter's shares can be lost without damaging integrity at all. However, its privacy guarantees are weaker, as any coallition of k corrupted aggregators can compromise the privacy of every smart meter.

If we use Shamir's scheme for secret sharing, each smart meter randomly generates a polynomial $q(x)$ such that $q(0) = R_{m,t}$. It then computes the N_a shares as

$$r_{m,t,s} = q(s).$$

Share s from each meter is sent to aggregator s, so that each aggregator knows the value of $q(s)$ for the same s and several different polynomials. Each aggregator then computes the aggregation:

$$a_{spatial,s}(t, M) = \sum_{m \in M} r_{m,t,s}$$
$$a_{temporal,s}(m, T) = \sum_{t \in T} r_{m,t,s}.$$

After aggregation, each aggregator has the value of $Q(s)$ for a new polynomial Q that is the sum of all the original polynomials. Thus, these are shares of the secret $A_{spatial}(t, M)$ or $A_{temporal}(m, T)$. The utility, which is in posession of all the aggregates, can reconstruct the secret by using the Shamir's secret sharing scheme "combine" operation. That is, it uses polynomial interpolation to compute the value of $Q(0)$.

$$A_{spatial}(t, M) = \text{Combine}(\{a_{spatial,s}(t, M) : 1 \leq s \leq N_a\})$$
$$A_{temporal}(m, T) = \text{Combine}(\{a_{temporal,s}(m, T) : 1 \leq s \leq N_a\}).$$

In this system, it is very important that each share $a_{spatial,s}$ used as part of the final reconstruction be constructed of shares $r_{m,t,s}$ from exactly the same set of smart meters. Otherwise, they are not points on the same polynomial and the interpolation process produces a nonsense result. So, the aggregators in this system must provide to the utility not only the x-coordinate s and the aggregated y-coordinate $a_{spatial,s}$, but also the set of meters involved in the computation. The utility can then ensure that it only combines shares that are compatible.

5.4.1 Security

This system also posesses information-theoretic security. The shares of each measurement are N_a points on a polynomial of degree $k - 1$ with coefficients in $\mathbb{Z}_p$. Suppose that only $k - 1$ shares (that is, points) are available. We select an additional point $(0, y)$, considering each value in $\mathbb{Z}_p$ in turn as a possible value for y. Now we have k points, and any k points with distinct x-coordinates determine a unique polynomial of degree $k - 1$, so each attempt produces a valid polynomial, and each has a different y-intercept. Since the secret is the y-intercept of the reconstructed polynomial, each value in $\mathbb{Z}_p$ now appears to be equally likely to be the value of the secret. So, we conclude that no information about the secret can be reconstructed from $k-1$ shares.

5.4.2 Example

We show an example computation using 2-out-of-3 secret sharing in the prime field Z_{11}. Suppose a meter produces the two measurements $R_{1,1} = 3$ and $R_{1,2} = 1$. For each measurement, it generates a random degree-1 polynomial with constant term R, for example:

$$q_1(x) = 3 + 7x$$
$$q_2(x) = 1 + 10x.$$

Shares are computed by evaluating the polynomials:

$$r_{1,1,1} = q_1(1) = 10$$
$$r_{1,1,2} = q_1(2) = 6$$
$$r_{1,2,1} = q_2(1) = 0$$
$$r_{1,2,2} = q_2(2) = 10.$$

Aggregator 1 receives $r_{1,1,1}$ and $r_{1,2,1}$ and computes

$$a_{temporal,1}(1, \{1,2\}) = 10 + 0 = 10.$$

Similarly, aggregator 2 computes

$$a_{temporal,2}(1, \{1,2\}) = 6 + 10 = 5.$$

We can see that these aggregates are correct because both $(1, 10)$ and $(2, 5)$ are points on the polynomial $Q(x) = q_1(x) + q_2(x) = 4 + 6x$.

Given these shares of the aggregate, the utility performs polynomial interpolation (using Equation 1) to compute

$$\begin{aligned} A_{temporal}(1, \{1, 2\}) &= \frac{0-2}{1-2}a_1 + \frac{0-1}{2-1}a_2 \\ &= (2)10 + (-1)5 \\ &= 4. \end{aligned}$$

This final value is the correct temporal aggregation of the original measurements, 3 and 1.

5.5 Summary

To assist in comparing the three systems, Table 1 summarizes the operation performed by each component in each system. Although each of the listed operations requires several types of computation, there is generally one computation that dominates the CPU time. Table 2 highlights the computation expected to be the most expensive one performed by each component.

Table 1: Primary computation performed by each system component.

System	Smart Meter	Aggregator	Utility
Baseline	Paillier encryption	Homomorphic addition	Paillier decryption
Probabilistic	Additive secret sharing	Addition of shares	Combining of shares
Threshold	Shamir's secret sharing	Addition of shares	Combining of shares

Table 2: Most expensive computation performed by each system component.

System	Smart Meter	Aggregator	Utility
Baseline	Modular exponentiation	Modular multiplication	Modular exponentiation
Probabilistic	Random number generation	Modular addition	Modular addition
Threshold	Polynomial evaluation	Modular addition	Polynomial interpolation

Although the most expensive computation for smart meters in the probabilistic system is listed as random number generation, that one is especially implementation dependent. If A_{max} is made larger or smaller, changing the cost of modular subtraction, or random number generation is done using a different algorithm, the subtraction could become the more significant operation.

6 Implementation

The implementation developed for this thesis is available online [30]. This section describes some of the choices that were made when building this implementation.

6.1 Assumptions

Given the constrained nature of the target device, we have made our implementations fairly optimized for the application. For example, instead of building libraries that can support encrypting or sharing any size of secret, which would require handling arithmetic on arbitrarily-sized numbers, we have limited the size of the input to simplify the algorithm and allow use of the faster native arithmetic operations.

We assume that individual measurements fit in 16 bits, and that the sum of all measurements within a neighborhood or within a billing interval occupies no more than 32 bits. That is, we assume that $R_{max} = 2^{16} - 1$ and $A_{max} = 2^{32} - 1$. Since each reading is a single energy measurement, 16 bits should contain enough data for at least 4 significant decimal digits of precision, more than found in most power bills. These assumptions allow us to perform secret sharing within 32-bit groups without fear of overflow, and to draw conclusions about the error introduced by missing shares. For consistency, all output values (shares snd ciphertexts) are serialized to byte arrays, and the performance tests include the time to make this conversion.

All of the implementations use the Linux `/dev/urandom` device as their source for random data. This may decrease the realism of the simulation somewhat, since real smart meters may not run operating systems complex enough to have a built-in random number generator like `/dev/urandom`. Still, this choice allows us to use cryptographically sound random number generation with minimal implementation work, and since the same generator is used for all systems it should at least allow for a fair comparison.

6.2 Paillier Encryption

The Paillier encryption algorithm is implemented from scratch, but uses the C library GMP [19] to perform arithmetic with large numbers. A key is represented as a struct of all the values used in the algorithm, with each value contained in a `mpz_t` object provided by GMP. Values to encrypt are accepted as 32-bit integers, then immediately converted to `mpz_t` which are used to compute the encrypted value as shown in Listing 1. Encrypted results are then serialized to a byte string.

Listing 1: Computing $g^m \cdot r^n \pmod{n^2}$ using GMP.

```
mpz_powm(term1, key->g, m, key->n2);
mpz_powm(term2, r, key->n, key->n2);
mpz_mul(c, term1, term2);
mpz_mod(c, c, key->n2);
```

Following the NIST recommendations for RSA [2], we select our n to be 2048 bits long by multiplying two 1024-bit primes.

6.3 Probabilistic Secret Sharing

This system is the simplest of the three, especially since limiting the size of the inputs allows us to use native subtraction to compute the final share. However, there are still some subtleties to this implementation. One important one is the random selection of aggregators to receive shares each round. The security of the scheme does not rely on these numbers being unpredictable, only uniformly distributed, so we feel comfortable depending on the `rand_r` function for our randomness. However, using the modulus operator to get a random number between 0 and N_a would introduce bias into the distribution unless N_a evenly divides RAND_MAX +1 (the number of possible outputs from `rand_r`).

To avoid this bias, we carve out a subset of the possible outputs that is divisible by N_a, and select random numbers until we get one in this subset. Then it is safe to use modulus to return the final answer. A slightly simplified implementation of this algorithm is shown in Listing 2.

Listing 2: Computing uniformly distributed random numbers.

```
max = RAND_MAX/n;
max *= n;
do {
  val = rand_r(seedp);
} while (val >= max);
return val % n;
```

We use integer division by n to truncate off any fractional part of the quotient, then multiply by n again, resulting in the largest multiple of n less than RAND_MAX. Then we select random numbers, retrying if the value is not in $[0, \texttt{max})$.

6.4 Shamir's Scheme

The implementation of Shamir's secret sharing scheme used for this work was adapted from libgfshare [37]. Libgfshare is designed to speed computation by performing all operations in $GF(2^8)$ instead of a large prime field as suggested by Shamir's paper. It does this by sharing each byte of a secret individually rather than considering it a unified integer. This approach does not work in our case, because the aggregation functionality depends on the secret sharing scheme having a homomorphic property, and the homomorphism must apply to the integer represented by the entire secret, not the individual bytes it contains. So, the code was modified to use integer arithmetic modulo the prime $P = 2^{32} - 5$, the largest prime that fits in 32 bits.

Limiting the secrets and shares to 32 bits made it possible to implement these algorithms without using a multi-precision arithmetic library like GMP. However, it was still necessary to be mindful of the size of the operands when performing polynomial interpolation to reconstruct the secret. Computing each term of Equation 1 requires multiplying several elements in the modular field. During this step, 64-bit variables were used as accumulators, and modular reduction performed on the result after each step, to ensure that the result would not overflow the variable.

Division of field elements is accomplished by multiplying by the denominator's multiplicative inverse. The extended Euclidean algorithm is used to compute these modular multiplicative inverses, again using 64-bit variables to avoid overflow.

6.5 Test Harness

In addition to the three algorithms, our implementation includes `runtests`, a tool to execute the smart meter's computational tasks within the system and measure the performance of the algorithms. This is a single program that includes the code for all three algorithms, and runs whichever one is specified by its command line flags. The program operates as follows:

1. Parse command-line flags (see Listing 3 for available options)
2. Generate random 16-bit message
3. Allocate memory and initialize algorithm
4. Begin timing
5. Repeat the following the specified number of iterations:
    (a) Supply plaintext message to algorithm
    (b) Serialize each share or ciphertext to a byte string
6. Measure elapsed time
7. Clean up algorithm

Running the algorithm multiple times allows more accurate timing of the operation, which is too fast to produce accurate measurements individually.

The command-line options accepted by the program are given in Listing 3.

Listing 3: Command line options for `runtests`.

```
-N: Test the probabilistic n-out-of-n secret sharing algorithm
-S: Test the threshold (Shamir's) secret sharing algorithm
-P: Test the Paillier encryption algorithm
-i <iterations>: Specify how many times to run the algorithm (default 10)
-a <shares>: Specify the total number of aggregators (-N and -S only)
-n <shares>: Specify the number of shares to compute (-N only)
```

```
-k <shares>: Specify the threshold for secret reconstruction (-S only)

-p <key path>: Specify the path to a Paillier public key (-P only)
```

7 Results

7.1 Privacy Protection

All of the systems protect privacy when the aggregators are honest. Thus, to understand how well these systems protect privacy under our threat model, we must understand the behavior under increasing numbers of compromised aggregators.

The simplest case to analyze is the baseline system, which has only one aggregator. Thus any compromise of the aggregator compromises the privacy of all the users.

The analysis of the threshold system is similar, though the results are more encouraging thanks to the distribution of trust. When $N_c < k$, fewer than k shares are available to the attacker, so the secret sharing scheme guarantees that no information about the secret can be derived. However, if $N_c \geq k$, those N_c aggregators each have a share of every user's data. Together, by combining k or more shares from each user they can reconstruct the measurement submitted by every user.

Finally, we consider the system using probabilistic secret sharing. In this system, whether a meter's data is compromised depends on which aggregators were sent its shares. As with the threshold system, when $N_c < k$, not enough shares are available to compromise any user. However for larger values of N_c, there is a chance that a user's data is exposed. With N_a aggregators to choose from, there are $\binom{N_a}{k}$ possible sets a particular meter may have chosen. The meter's data is not compromised unless all k chosen aggregators fall within the N_c that are compromised. So, $\binom{N_c}{k}$ of the possible sets result in compromise. Thus the probability of each meter's data being compromised is

$$P(\text{compromise}) = \frac{\binom{N_c}{k}}{\binom{N_a}{k}}.$$

Figure 3 shows the probability of a meter's data being compromised for increasing numbers of compromised aggregators in a system containing 10 total aggregators. The threshold systems behave similarly to the baseline system, with a sharp line dividing protection from compromise. Increasing k increases the protection provided by moving that line. The probabilistic systems, on the other hand, retain a low probability of compromise until nearly all of the aggregators are compromised. A higher n does provide more protection, but even with $n = 3$, 8 of 10 aggregators must be compromised to steal about 50% of the data in the system.

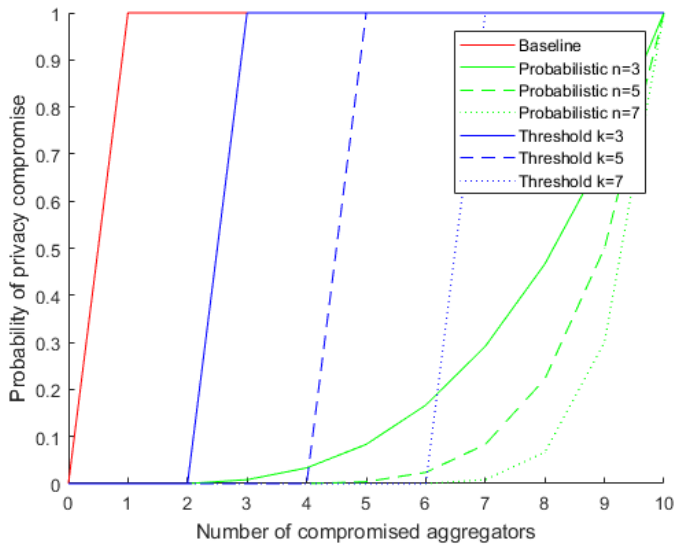


Figure 3: Relative privacy protection of the different schemes.

7.2 Integrity Protection

The only type of integrity attack allowed by our threat model is a dropped packet, so in this section we explore how many dropped packets are tolerated by a system, and the effect on the system output when that limit is exceeded. To analyze these effects more concretely, we assume that the range of possible readings from any smart meter is $[R_{min}, R_{max}]$, with a symmetrical distribution within that range. Thus the expected value of a reading is the center point of the range, $\frac{R_{max}+R_{min}}{2}$.

In the baseline system, if a message from a meter is lost, that meter's entire contribution to the sum is lost, meaning that the sum differs from the correct value by an expected $\frac{R_{max}+R_{min}}{2}$. Each lost message simply adds this same amount of expected error to the computation.

In the probabilistic system, each share is a uniformly-chosen element of a large group with elements $[0, A_{max}]$. A_{max} must be significantly larger than R_{max} so that multiple readings in the high end of the range do not cause overflow. Because the shares are randomly selected, when one share is lost the remaining shares of that meter reading are equally likely to add up to any value in $[0, A_{max}]$. Therefore, the total computed by the utility is also equally likely to be any value in $[0, A_{max}]$. Losing any share therefore prevents reconstruction of *any* information about the correct sum! This is a significant downside of the probabilistic system.

In the threshold system, the secret sharing scheme protects against error when some of the shares are lost. As long as k of the N_a shares of the final aggregate are intact, the final value is computed without error. On the other hand, if fewer than k shares of the same secret are available (that is, if fewer than k of the aggregators receive readings from exactly the same set of meters) then the secret sharing scheme ensures that in this case, too, no information about the correct sum can be derived.

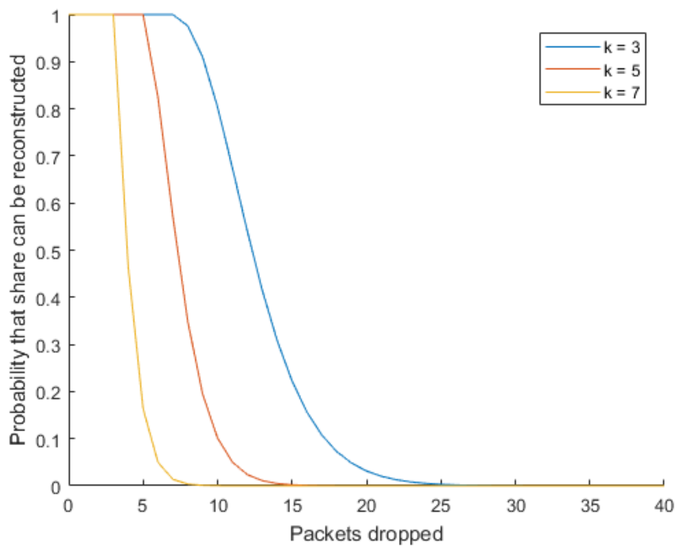


Figure 4: Probability of threshold system correctly reconstructing aggregate with dropped packets.

So, if fewer than $N_a - k$ packets are dropped, there is no error. With a larger number of dropped packets, the secret can still be reconstructed with some probability, depending on whether the lost packets went to the same aggregators or to different ones. Figure 4 shows how the probability of successfully reconstructing the correct aggregate drops off with larger numbers of dropped packets. The values pictured reflect a system with 10 smart meters and 10 aggregators. The graph shows that the probability of successful reconstruction is 1 until $N_a - k$ packets have been dropped, though it drops off fairly steeply after that. For example, when $k = 5$, reconstruction is always successful for 5 dropped packets, but for 8 dropped packets the probability is less than 0.35. Theoretically, there is still a chance for reconstruction with as many as $N_m(N_a - k)$ dropped shares (50 shares for $k = 5$), but the graph shows that the probability becomes negligible much sooner than that.

The values for this analysis were computed numerically using an algorithm developed as part of this thesis and presented in Appendix A.

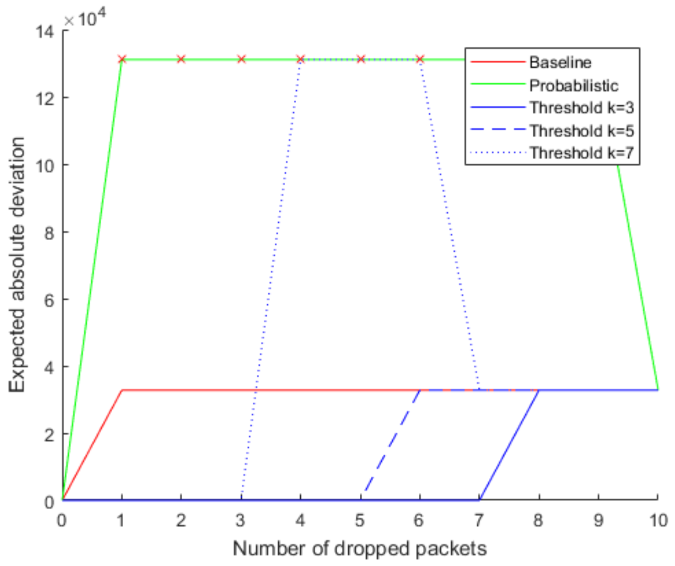


Figure 5: Expected value of error in final aggregate as number of packets dropped (all from the same smart meter) increases. Red X's represent cases where all information is lost.

Both the threshold and probabilistic systems perform differently against integrity threats when all the missing shares come from the same meter. Figure 5 shows the expected error resulting from this special case, modeling error as the absolute deviation from the correct value of the computed aggregate. The system is modeled with $N_a = 10$, $R_{min} = 0$, and $R_{max} = 2^{16}$. Points marked with red X's in the graph represent cases where no information about the correct aggregate value can be reconstructed. The y-coordinates of those points are arbitrarily chosen and do not represent actual amounts of error, because the the system would be designed to detect those situations and raise an error anyway.

The figure shows that in the probabilistic system, any number of shares lost from a meter cause the sum to be effectively meaningless, *except if all of the meter's shares are lost.* In this case the effect is much smaller, like losing one measurement in the baseline system.

In the threshold system, as long as the threshold $k \leq N_a/2$, even if more then $N_a - k$ of the shares from a single smart meter go missing, those aggregators would still be able to compute shares of a meaningful answer: an aggregate that excludes the results from the missing smart meter. Since $k \leq N_a - k$, any k of those aggregators should be able to compute an aggregate, and the resulting error would again be the same as a missing share in the baseline system.

7.3 Performance

A critical question for these systems is whether they can successfully be run on smart meters, which have low computational power. If the algorithm makes the meter draw too much power, or takes too much time to run, it could miss measurements or increase the power cost of metering to unsustainable levels. To understand the performance of the algorithms, we ran our implementation on a Raspberry Pi 1 Model B, which has a 700MHz ARMv6 CPU and 512 MB of RAM, as well as an 8 GB SD card for storage. This device is probably unrealistically powerful for a smart meter; however it is more representative than a desktop CPU.

Figure 6 shows our performance measurements of the computation done by the smart meter, in a simulated system with 10 aggregators. Paillier encryption is clearly more expensive than the secret sharing schemes. However, all systems were able to run several times per minute, which should be sufficiently fast to enable a 5- or 15-minute measurement interval.

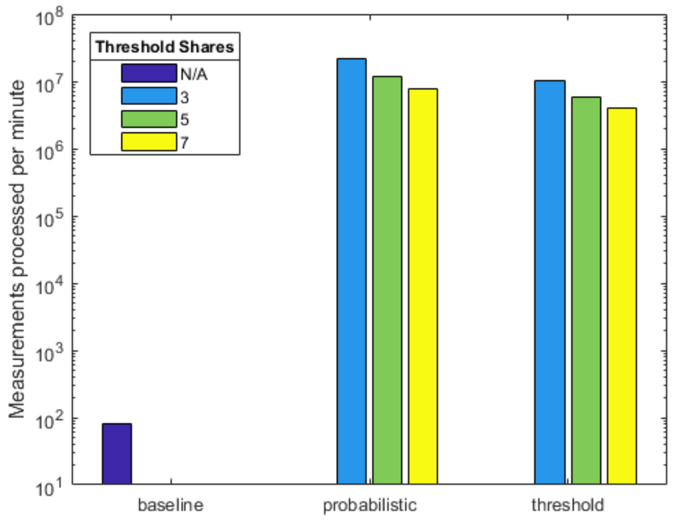


Figure 6: Processing throughput of privacy-preserving schemes.

7.4 Data Transmission Cost

In addition to processing time, another cost these systems must consider is the transmission of data. This is especially relevant for smart meters that transmit data wirelessly, in which case transmission can use several orders of magnitude more energy than processing. In Table 3 we evaluate the data transmission required for a single smart meter to send a single measurement. We present both the theoretical number of bits required, and the number used in our implementation of the protocols. In the analysis below, l_k is the number of bits in the Paillier public key.

Table 3: Comparison of data transmission required by different schemes.

Scheme	Theoretical	Our Implementation
Baseline	$2l_k$	4096 bits
Probabilistic	$n \log A_{max}$	$32n$ bits
Threshold	$N_a \log A_{max}$	$32N_a$ bits

It is worth noting that the overall data transmission rate is relatively low in any case since only one measurement is sent every few minutes.

8 Discussion

The above analysis and experimentation shows that each system provides a different set of tradeoffs. Depending on the goals of system implementors, a different selection of systems or even a combination of the systems might be the best choice.

If robustness to dropped packets is the primary concern, the threshold scheme is a clear winner, supporting up to $N_a - k$ dropped packets, and probabilistically even more, with no error whatsoever. However, the scheme does not degrade gracefully beyond that point; if the system is unable to compute k exactly correct aggregated shares, the computed value is effectively nonsense. The most successful system for graceful degredation is actually the baseline one, due to its having only one share per measurement. Losing a single measurement is tolerable because the value lost is small relative to the total aggregate. On the other hand, losing any share in the probabilistic system or too many shares in the threshold system adds enough noise to completely overwhelm the real signal.

It is worth considering what steps can be taken to prevent that noise from being added. If the aggregators could communicate with one another that a share was missing, they could compute their aggregates excluding shares from the affected smart meter (or shares from the affected time interval, if performing temporal aggregation). This would reduce the impact of the missing share down to just a missing measurement—the same as in the baseline system. The protocol between the aggregators to omit a smart meter would need to be carefully designed, though, otherwise this could be a useful tool for compromised aggregators. An aggregator could request others to omit all but one smart meter, revealing that meter's unaggregated data.

If the implementor's primary concern is privacy, the probabilistic scheme is preferable. With a given threshold value, the threshold scheme preserves privacy exactly until the threshold, then abandons it completely. In contrast, even if more than the threshold number of aggregators have been compromised, the probabilistic scheme only reveals a random subset of the measurements. In this situation, although a user may be compromised occasionally, it is not possible to get a full record of his or her usage. The baseline system, because it only uses a single, trusted aggregator, provides no privacy protection against compromised aggregators.

The two ideas tested in these systems don't need to be mutually exclusive; the best solution for a given deployment might actually be a combination of theshold secret sharing and probabilistic assignment of shares to aggregators. Use of threshold secret sharing would provide some resistance to errors, making a lost share less devastating than in the probabilistic scheme. However, randomly selecting aggregators rather than distributing shares to all aggregators would prevent unilateral privacy compromise once the threshold number of aggregators are compromised.

To consider this proposal in more detail, suppose we begin with the threshold scheme with a threshold of k. Then there are $N_a - k$ redundant shares, which can be damaged (for example, by missing packets) without affecting the result. We can sacrifice some of this resilience in exchange for privacy by decreasing the number of redundant shares. If we only send out N_s shares, $N_s < N_a$, then only $N_s - k$ damaged shares can be tolerated. But if we randomly select the aggregators to receive shares each round, k compromised aggregators will no longer be able to reconstruct the unaggregated measurements every round. Instead, the attacker must compromise $k + N_a - N_s$ to guarantee the measurements can be read. Thus, the combined system would have both some error resistance and also some probabilistic protection against compromise by misbehaving aggregators.

Because the threshold scheme requires all shares of the aggregate to contain shares of exactly the same set of readings, this option is slightly more delicate to implement than our original probabilistic proposal. Each round, all meters must select the same set of aggregators to receive their shares, otherwise the effect on the shares is similar to that of dropped packets. This could be achieved without too much difficulty using a deterministic pseudorandom number generator and a shared seed, but keeping the seed synchronized may require some extra communication. Still, this combination allows implementors to make a direct tradeoff between privacy protection and integrity protection.

The random selection of aggregators could be used to improve the baseline system as well. If the system contained multiple aggregators, and each meter randomly selected one to process its data each round, the data would be probabilistically protected against aggregators compromised by the privacy attacker. This system still provides less privacy than the probabilistic system based on secret sharing, however, since that system allows the minimum threshold for data compromise to be set arbitrarily high by increasing the number of shares.

Another important consideration when selecting systems is performance. Most significant is the complexity of the algorithms executed on the smart meters themselves, as those are under the most severe performance constraints. Improving the computational power of the aggregators might require more servers, but improving the power of the smart meters might mean upgrading thousands or millions of devices. Fortunately, secret sharing turns out to be inexpensive relative to public-key cryptography. The two secret sharing schemes were fairly similar in performance; the need to randomly select a set of aggregators in the probabilistic scheme seems to have counteracted the reduced complexity of subtraction as opposed to polynomial evaluation as the secret sharing mechanism. Further performance tuning of the implementations could certainly cause these results to shift, as well.

Regardless of which of those algorithms is the fastest, it seems clear that Paillier encryption is several orders of magnitude slower than either. In a general sense, this difference is attributable to the fact that the security of secret sharing is information theoretic, while Paillier's security is computational. That is, when too few shares of a shared secret are available, the reconstructed secret could be any value within the algorithm's input range, with equal probability. No amount of computation can reveal the secret, so the size of the values in the system doesn't matter as long as it is big enough to provide uncertainty about the secret. In encryption, on the other hand, a public key is a math problem that can be solved with sufficient computation. The security of the system depends on that problem being very slow to solve, and key sizes are chosen to ensure that. Thus, the requirement that it be slow to decrypt without the key drives up the cost of encryption as well.

Ultimately, the most interesting question about performance is whether these schemes are feasible on embedded devices such as smart meters. We believe these results show that they are. Even the Paillier scheme, which is the most expensive, takes less than a second per encryption. We must also take into account the performance differential between our Raspberry Pi and a smart meter's microcontroller. If we suppose that meters are powered by chips like the STCOMET by STMicroelectronics or the Silergy MAX71315S, we can expect them to have clock frequencies of

50-100MHz, in contrast to the 700MHz CPU in the Raspberry Pi. Even if we suppose operations take 10 times as long on the smart meter, we can still perform about 8 Paillier encryptions per minute, which should be plenty for a measurement interval that is multiple minutes long.

8.1 Extended Threat Models

The threat model proposed in this thesis is somewhat optimistic. Although a dropped packet between smart meter and aggregator is the most probable type of error, there are other components that can fail, and in ways no protocol can prevent. For example, we could consider the possibility that an aggregator could go offline in the middle of a round. Assuming that the meters could still choose this aggregator as a target for their messages, this would interfere with the computed measurements in significant ways. In the threshold system it would cause that aggregator's share to be unavailable, reducing the redundancy in the system, which is much the same effect as a dropped smart meter share. In the probabilistic system, on the other hand, losing an aggregator means losing one share from multiple different smart meters. This is almost certainly enough to make the result meaningless. And in the baseline system, losing the single aggregator brings the system to a halt.

We could further explore potential risks to the system by considering the compromise of meters as well. Smart meters usually have very good physical security, but researchers have shown several practical network-based attacks against existing smart meters [10, 13], so a meter under the control of an attacker is far from impossible. A stronger threat model allowing compromised meters could, for example, allow the integrity attacker to gain full control of αN_m of the smart meters, for some fraction $0 \leq \alpha < 1$. To make the model even stronger, we could accept the threat model used by P4P [15], which merges the powers and knowledge of our two attackers into one. Or, the model of Prio [11], which would allow the aggregators to actively disrupt the protocol, could be used.

In models where the meters may be malicious, we must accept a certain amount of error, since the value transmitted by a meter under the control of an attacker is not necessarily related to the actual electricity usage. However, we can try to limit the damage these compromised meters can do. Both P4P [15] and Prio [11] present zero-knowledge proof systems that can put restrictions on the values being distributed, even though the values themselves are obscured via secret sharing. For instance, the magnitude of the secret value could be bounded to what the utility considers a plausible measurement. If the meter fails to send the aggregator a proof that a share belongs to a small-enough secret, the aggregator just discards the share. Losing the measurement of that meter produces a small amount of error but preserves the usefulness of the overall aggregate.

8.2 Implementation Assumptions

We have made some simplifying assumptions when analyzing and implementing these systems. The focus of this thesis is the means of protecting privacy and distributing trust, so the actual network protocol used by the system has been left unspecified. However, it is important to enumerate the requirements for that protocol to understand the implications they have on the system.

1. Handling of varying packet sizes. The secret sharing systems need to send 32-bit shares, while values in the Paillier system are around 2048 bits. We expect our network protocol to transparently handle these packets, fragmenting as necessary.

2. Mutual authentication of the smart meter and aggregator. Aggregators must accept measurements only from smart meters that belong to actual customers of the grid. Further, the aggregator must have accurate knowledge of the origin of a share in order to assign it to the appropriate neighborhood (for spatial aggregation) or customer bill (for temporal aggregation). Meanwhile, the smart meter must only share data with authorized aggregators, to prevent the user's data from being leaked.

3. Encrypted and authenticated data transmission. The baseline system is the only one of the three in which the data are encrypted, and that encryption does not protect the message from tampering. (In fact, tampering is highly effective thanks to the cryptosystem's homomorphic properties.) So, to protect the confidentiality of messages from external eavesdroppers and to protect the utility against data manipulation by a man in the middle, the protocol must encrypt, and verify the integrity of, all packets sent.

Fulfilling all these requirements requires some additional communication and cryptography that is not accounted for in our performance testing. However, as it is required by all the systems it should not affect their relative performance dramatically.

Another component of the system that is not fully demonstrated in our implementation is the functionality of the aggregators. For each system we have built tools to demonstrate that shares of that system can be added homomorphically, and recombined to return the original secret. However, implementing the full system requires more metadata about the received messages to be retained. In order to perform temporal aggregation, the aggregator must keep track of the customer and timestamp of each measurement, and retain measurements for at least a full billing cycle. In addition, in order to perform spatial aggregation of a neighborhood, the aggregator must know the neighborhood to which each measurement belongs. Keeping track of customers and timestamps is especially important in the threshold system, because the exact list of meters or time periods involved in an aggregation must be transmitted to the utility along with the aggregate value. This allows the utility to ensure it never attempts to combine shares from different polynomials.

9 Conclusions

User privacy is a significant challenge for the smart grid, as the high-frequency, per-household electricity usage data that is most useful to the utility also reveals sensitive data about the user's activities. In this thesis, we have detailed three candidate systems that aggregate user data before it reaches the utility, so that only the combined usage of a neighborhood of meters, or the combined monthly usage of a single household, can be reconstructed. The baseline system, based on homomorphic encryption, was compared to two alternate proposals that use homomorphic secret sharing; each system was evaluated based on its resilience to privacy and integrity attacks, its performance, and the data transmission costs it incurs.

All three systems were implemented and tested for performance. These performance results were encouraging; it appears that any of the systems should be feasible on a low-power processor such as a smart meter. However, the two implementations based on secret sharing performed nearly

10,000 times faster, so secret sharing has a clear performance advantage. It also has an advantage in transmission cost. Using the data sizes of our sample implementation, and assuming a moderate number of aggregators such as 10, the baseline system must transmit at least 10 times as much data per measurement. A larger data value or a larger number of aggregators could bring the two costs closer together, however.

When considering resilience to attacks, both of the new systems have an advantage over the baseline system, though they are helpful in different cases. The probabilistic system is useful primarily against a privacy attacker. While the baseline system compromises privacy completely with only one corrupted aggregator, and the threshold system does the same once a few more aggregators are corrupted, the randomized nature of the probabilistic system means at least some data is protected unless all aggregators are misbehaving.

On the other hand, the threshold system really shines against an integrity attack. While a dropped packet in the baseline system means a lost measurement, and a dropped packet in the probabilistic system generally means the loss of the entire aggregate, the threshold system has built in protection from dropped shares. As long as k of the aggregators receive shares from the same set of smart meters, they will be able to reconstruct a consistent aggregate. This means that the first few dropped packets cause no problem at all, and larger numbers of dropped packets have a probabilistic effect—a chance to disrupt the computation or not depending on which aggregators they were meant for.

The ideas tested in these systems allow system implementors to distribute trust among multiple entities. By doing so, they can benfit the user by ensuring that no one aggregator can compromise all users' privacy, and the utility by ensuring that that no one dropped packet can compromise the validity of the final result. Thus these techniques, applied individually or in combination, can make privacy-preserving smart metering more resilient to threats.

Appendices

A Probability of Successful Aggregation

We aim to compute the probability that the system based on threshold secret sharing (presented in Section 5.4) successfully reconstructs the correct aggregate value. That is, we wish to know the probability that at least k of the aggregators receive a share from every smart meter, given that N_d of the total shares in the system have been lost.

We compute this probability by studying a simpler function, S. $S(N_d, N_t)$ returns the number of possible sets of dropped packets, given that N_d packets are dropped, and each of those packets was going to one of N_t different aggregators (with all N_t aggregators losing at least one packet). Note that the total number of aggregators is still N_a, and the number of packets meant for each aggregator is equal to the number of smart meters, N_m.

We compute values of S iteratively, starting with $N_t = 0$. In this case we have

$$S(N_d, 0) = \begin{cases} 1 & N_d = 0 \\ 0 & otherwise, \end{cases}$$

because there is only one set of 0 dropped packets, and it touches 0 aggregators. Similarly, $S(N_d, N_t) = 0$ for any $N_d < N_t$, since it is impossible for the loss of N_d packets to affect more than N_d aggregators.

In all remaining cases, we compute the values for N_t aggregators recursively from the values for $N_t - 1$ aggregators. Suppose we start with a set of d dropped packets that touches $N_t - 1$ aggregators. From that set we can construct several sets that touch N_t aggregators by choosing which packets destined for the new aggregator are dropped. The number of additional packets to drop is $N_d - d$, selected from the N_m packets going to the aggregator, so $\binom{N_m}{N_d - d}$ new sets can be constructed for each old one.

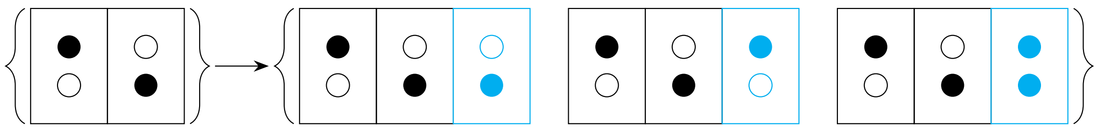


Figure 7: Analyzing the possible arrangements of dropped packets in a system with 2 smart meters sending packets. Starting from one (of 4) ways that 2 dropped packets can affect 2 aggregators, we can construct 2 ways 3 aggregators can be affected by 3 packets, and 1 way they can be affected by 4 packets.

Following this logic, we have the following recurrence relation:

$$S(N_d, N_t) = \sum_{d=\max(N_t-1, N_d-N_m)}^{N_d-1} S(d, N_t-1) \cdot \binom{N_m}{N_d-d}.$$

A note about the bounds: the upper bound for d is $N_d - 1$ because we need at least 1 dropped packet to belong to the newly added aggregator. The lower bound has two parts. The first, $N_t - 1$, is the smallest value of d where the previous round of the S function takes a value. Since $S(N_d, N_t) = 0$ for any $N_d < N_t$, we don't need to try d any smaller than $N_t - 1$. The second part, $N_d - N_m$, reflects the maximum number of packets that can be lost on the way to a single aggregator. If d were smaller than this, it would mean the last aggregator had lost more than N_m packets, which is impossible.

Now that we have a means of computing S, we can use it to compute the probability that the aggregate can be reconstructed given N_d dropped packets. As there are $N_m \cdot N_a$ total packets in a round, there are $\binom{N_m \cdot N_a}{N_d}$ sets of packets that could be dropped. $S(N_d, N_t)$ gives the number of these sets that affect the results from a specific set of N_t aggregators. We want to consider all possible sets of N_t aggregators, where $N_t \leq N_a - k$, the threshold for successful reconstruction. Our final probability of reconstruction, then, is

$$P(N_d) = \frac{\sum_{N_t=0}^{N_a-k} S(N_d, N_t) \cdot \binom{N_a}{N_t}}{\binom{N_m \cdot N_a}{N_d}}.$$

References

[1] C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Transactions on Information Theory*, 29(2), March 1983.

[2] Elaine B. Barker and Allen L. Roginsky. Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths. Technical Report NIST SP 800-131Ar1, National Institute of Standards and Technology, November 2015.

[3] Antonella Barletta, Christian Callegari, Stefano Giordano, Michele Pagano, and Gregorio Procissi. Privacy preserving Smart Grid Communications by Verifiable Secret Key Sharing. In *2015 International Conference on Computing and Network Communications (CoCoNet)*, December 2015.

[4] Josh Cohen Benaloh. Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret (Extended Abstract). In *Advances in Cryptology — CRYPTO' 86*. Springer, Berlin, Heidelberg, August 1986.

[5] Anna Biselli, Elke Franz, and Maurílio Pereira Coutinho. Protection of Consumer Data in the Smart Grid Compliant with the German Smart Metering Guideline. In *Proceedings of the First ACM Workshop on Smart Energy Grid Security*, SEGS '13, New York, NY, 2013. ACM.

[6] G. R. Blakley. Safeguarding cryptographic keys. In *Managing Requirements Knowledge, International Workshop on*, Los Alamitos, CA, 1979. IEEE Computer Society.

[7] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*. Springer, 2011.

[8] Zvika Brakerski and Vinod Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS '11, Washington, DC, 2011. IEEE Computer Society.

[9] Christian Callegari, Sara De Pietro, Stefano Giordano, Michele Pagano, and Gregorio Procissi. A Distributed Privacy-Aware Architecture for Communication in Smart Grids. In *2013 IEEE 10th International Conference on High Performance Computing and Communications 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC)*, November 2013.

[10] CCC. On Smart Cities, Smart Energy, And Dumb Security. URL: `https://media.ccc.de/v/33c3-8272-on_smart_cities_smart_energy_and_dumb_security`.

[11] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, Robust, and Scalable Computation of Aggregate Statistics. *arXiv preprint arXiv:1703.06255*, 2017.

[12] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. A generalization of Paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, 9(6), September 2010.

[13] Don C. Weber. OptiGuard: A Smart Meter Assessment Toolkit. Technical report, InGuardians, Inc., July 2012.

[14] Yitao Duan and John Canny. Practical private computation and zero-knowledge tools for privacy-preserving distributed data mining. In *Proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM, 2008.

[15] 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, Berkeley, CA, 2010. USENIX Association.

[16] C. Efthymiou and G. Kalogridis. Smart Grid Privacy via Anonymization of Smart Metering Data. In *2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, October 2010.

[17] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science (SFCS 1987)*, October 1987.

[18] Vitaly Ford, Ambareen Siraj, and Mohammad Ashiqur Rahman. Secure and efficient protection of consumer privacy in Advanced Metering Infrastructure supporting fine-grained data analysis. *Journal of Computer and System Sciences*, 83(1), February 2017.

[19] Free Software Foundation. The GNU MP Bignum Library. URL: `https://gmplib.org/`.

[20] Flavio D. Garcia and Bart Jacobs. Privacy-Friendly Energy-Metering via Homomorphic Encryption. In Jorge Cuellar, Javier Lopez, Gilles Barthe, and Alexander Pretschner, editors, *Security and Trust Management*, Lecture Notes in Computer Science 6710. Springer Berlin Heidelberg, September 2010.

[21] Shafi Goldwasser, Vipul Goyal, Abhishek Jain, and Amit Sahai. Multi-Input Functional Encryption. Technical Report 727, 2013.

[22] Ulrich Greveler, Peter Glösekötterz, Benjamin Justusy, and Dennis Loehr. Multimedia content identification through smart meter power usage profiles. In *Proceedings of the International Conference on Information and Knowledge Engineering (IKE)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2012.

[23] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient Non-interactive Secure Computation. In *Advances in Cryptology – EUROCRYPT 2011*. Springer Berlin Heidelberg, May 2011.

[24] Christine Jost, Ha Lam, Alexander Maximov, and Ben Smeets. Encryption performance improvements of the Paillier cryptosystem. *IACR Cryptology ePrint Archive*, 2015, 2015.

[25] E. Karnin, J. Greene, and M. Hellman. On secret sharing systems. *IEEE Transactions on Information Theory*, 29(1), January 1983.

[26] Sye Loong Keoh, Yi Han Ang, and Zhaohui Tang. A lightweight privacy-preserved spatial and temporal aggregation of energy data. In *2015 11th International Conference on Information Assurance and Security (IAS)*, December 2015.

[27] S. C. Kothari. Generalized Linear Threshold Scheme. In *Advances in Cryptology*. Springer, Berlin, Heidelberg, August 1984.

[28] Klaus Kursawe, George Danezis, and Markulf Kohlweiss. Privacy-Friendly Aggregation for the Smart-Grid. In Simone Fischer-Hübner and Nicholas Hopper, editors, *Privacy Enhancing Technologies*, Lecture Notes in Computer Science 6794. Springer Berlin Heidelberg, July 2011.

[29] Fengjun Li, Bo Luo, and Peng Liu. Secure Information Aggregation for Smart Grids Using Homomorphic Encryption. In *2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, October 2010.

[30] Benjamin Lipton. Smart Grid Privacy through Distributed Trust — System Implementation. URL: `https://github.com/LiptonB/smart-grid-privacy`.

[31] Andrés Molina-Markham, Prashant Shenoy, Kevin Fu, Emmanuel Cecchet, and David Irwin. Private Memoirs of a Smart Meter. In *Proceedings of the 2Nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, New York, NY, 2010. ACM.

[32] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can Homomorphic Encryption Be Practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, New York, NY, 2011. ACM.

[33] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, Lecture Notes in Computer Science 1594. Springer Berlin Heidelberg, May 1999.

[34] Torben Pryds Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology — CRYPTO '91*. Springer, Berlin, Heidelberg, August 1991.

[35] C. Rottondi, G. Verticale, and A. Capone. A security framework for smart metering with multiple data consumers. In *2012 Proceedings IEEE INFOCOM Workshops*, March 2012.

[36] Adi Shamir. How to Share a Secret. *Commun. ACM*, 22(11), November 1979.

[37] Daniel Silverstone. libgfshare, 2008. URL: `http://www.digital-scurf.org/software/libgfshare`.