Private and Secure Smart Meter Billing

Mohammed Ababneh
New Mexico State University
Las Cruces, NM, USA
mababneh@nmsu.edu

Kartick Kolachala
New Mexico State University
Las Cruces, NM, USA
kart1712@nmsu.edu

Roopa Vishwanathan
New Mexico State University
Las Cruces, NM, USA
roopav@nmsu.edu

ABSTRACT

The introduction of smart grids has changed how electric power is distributed and how power companies measure electricity usage by consumers and generate bills. When smart grids and smart meters calculate and report the power usage of a customer to a utility provider in a user friendly way, we need to ensure that privacy of the customers is not violated and no sensitive data such as their energy consumption habits are revealed either to the utility provider or to third parties, trusted or otherwise. To this end, we propose a novel scheme to compute electricity usage of customers and report it to a utility provider in a secure manner, using cryptographic primitives such as secret sharing, aggregate signatures, and a distributed file system for data storage and retrieval. Using our proposed system, a utility provider can accurately bill a customer for their energy consumption, without having to know the customer's energy usage habits, and without having to rely on trusted hardware or a trusted third party.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**; **Software and application security**;

KEYWORDS

Smart grid, Aggregate signatures, Distributed file systems

ACM Reference Format:
Mohammed Ababneh, Kartick Kolachala, and Roopa Vishwanathan. 2022. Private and Secure Smart Meter Billing. In *Proceedings of the 8th ACM Cyber-Physical System Security Workshop (CPSS '22), May 30, 2022, Nagasaki, Japan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3494107.3522775

1 INTRODUCTION

In the last two decades, the traditional power grid has faced increasing scrutiny from industry as well as the academic community due to emerging technologies such as renewable energy, rising demand for electricity, the ever-increasing complexity of electrical power systems, and the need for highly reliable, efficient, and secure power supply, which has spurred research in, and lead to the development of *smart grids*. A smart grid is the next generation of the power grid which, broadly, involves four sub-systems: Advanced Metering Infrastructure, Advance Distribution Operations, Advanced Transmission Operations, and Advanced Management Assets. These sub-systems support the function of data integration, information processing, monitoring and analysis of system equipment, and data communication to enhance the availability, reliability and confidentiality of the smart grid.

The most crucial sub-system in the smart grid is the Advanced Metering Infrastructure (AMI), whose components include smart meters, communication networks, and information management systems. The AMI's communication network establishes bidirectional communication between the utility provider and the customer. The AMI's smart meter records a customer's energy consumption data within certain intervals of time, e.g., every fifteen minutes, and sends the customer's energy consumption data through insecure, commonly available fixed networks, such as broadband over power line, power line communications, as well as landline and cellular public networks. The energy consumption readings are sent to the AMI's information management system which handles the data storage and analysis, and provides information in an appropriate format to the utility provider [23]. A utility provider can use the energy consumption data in several data analytics applications such as forecasting energy demand, billing automation, and confronting energy theft. It also helps customers monitor their energy consumption.

Unfortunately, the customers' energy consumption data collected by smart meters include private data such as a homeowner's presence (or absence) at a home, the kind of appliances that exist at the home, and to what extent the homeowners use each of their appliances. Molina-Markham *et al.* [24] show that even without prior knowledge of household activities or prior training, it is possible to extract complex usage patterns from smart meter data using off-the-shelf statistical methods such as clustering and pattern recognition techniques.

Researchers have proposed solutions for privacy-preserving smart meter billing using compute-intensive cryptographic protocols such as non-interactive zero knowledge proofs [1, 18, 24, 29]. There are also non-privacy preserving solutions for secure billing that use a blockchain [14], which require a customer to interact with the utility provider in the billing period, thus requiring them to be online all the time. There have also been solutions proposed for private billing that use secret sharing schemes [6, 16, 22, 30, 36], but require trusted third parties.

Our goal in this paper is to design a billing mechanism that protects the privacy of the customers, hides their fine-grained electricity consumption data (e.g., fifteen-minute interval readings) from the utility provider, only gives the utility provider the accurate aggregate readings at the end of the month to generate a bill,

CPSS '22, May 30, 2022, Nagasaki, Japan

ACM ISBN 978-1-4503-9176-4/22/05…$15.00
https://doi.org/10.1145/3494107.3522775

accomplishes this without using trusted third parties, is scalable, and does not use heavy cryptographic machinery.

Our Contributions

- We propose a novel scheme for secure billing of smart meters while safeguarding the privacy of customer electricity consumption data. Unlike the prior works published in this area that makes use of trusted aggregators and/or computationally expensive cryptographic schemes, we do not require any entity in our system to be trusted and use cryptographic mechanisms with minimal overhead.
- Our system records and reports the consumption readings in an accurate way, with data verified at every step, is parallelizable, and uses inexpensive cryptographic primitives. We experimentally evaluate our scheme to demonstrate scalability.

Outline: The remainder of this paper is organized as follows. In Section 2, we review relevant related work, in Section 3 and Section 4 we describe our system and adversary models. In Section 5 we describe our construction and its constituent algorithms, and in Section 6 we present our experimental evaluations. In Section 7, we describe the security analysis of our model. In Section 8 we discuss our design choices and alternatives, and in Section 9, we conclude the paper.

2 RELATED WORK

Privacy-preserving billing in smart meters has been an active area of research in the past decade, with researchers exploring various cryptographic protocols, and proposing solutions with varying levels of efficiency and trust requirements. The relevant related work can be broadly divided into four categories: privacy-preserving billing schemes that use compute-intensive cryptographic protocols, schemes that use a trusted aggregator or a blockchain, schemes based on secret sharing, and schemes based on masking. We now discuss the work in the categories.

Compute-intensive cryptographic protocols: Jawurek *et al.* [18] proposed a scheme that allows a utility provider to compute customer bills using energy consumption data without compromising customer privacy using a trusted *privacy component*, which can do non-trivial cryptographic operations such as generating and verifying homomorphic commitments and signatures. The authors envision the privacy component to be implemented as a software application on the smart meter, which unfortunately, leaves it open to tampering. Diao *et al.* [10] proposed a smart metering scheme with the aim of protecting customers' privacy (i.e., identity hiding) using anonymous signature schemes, but in the process, reveal the fine-grained energy consumption data to the utility provider. On a similar note, Hu *et al.* [17] proposed a technique to protect the identity of smart meters by having the meters prove their identity in zero knowledge to the utility provider, but do not consider customer's fine-grained energy consumption privacy. Saxena *et al.* [31] introduced a secure and privacy-preserving data aggregation approach based on the Paillier cryptosystem's additive homomorphic encryption and proxy re-encryption operations. Their approach can securely aggregate metering data without disclosing personal information (such as identity of the individual customers or energy usage) to intermediaries or third parties. In their system, they assume the existance of two trusted entities, *billing centers* and *data control units*, and honest-but-curious data aggregators. In our system model, we do not assume the existance of any trusted entity; all entities in our system can be potentially malicious and can deviate from the described protocols at any point. Furthermore, they employ homomorphic encryption, which is computationally expensive and adds to the smart meter's overhead. We use comparatively inexpensive cryptographic schemes such as secret sharing and aggregate signatures. We provide a comparison of our scheme with Saxena *et al.* [31] in Section 6.

Mustafa *et al.* [25] developed a secure and confidential data aggregation method for gathering operational metering data, which is later used for computing distribution, transmission, and imbalance fees. Secure multiparty computation (a cryptosystem which is known to have significant computational overhead) is their core cryptographic protocol, and it supports three different privacy-friendly data aggregation algorithms. They also presume that at least one server from the *data communications company*, which is an entity that aggregates and delivers users' data to suppliers, is trustworthy. A billing mechanism for smart meters' energy usage is not supported by their system; our focus is on providing a secure and privacy-preserving billing system, hence the work of [25] is orthogonal to ours.

Tonyali *et al.* [35] propose new protocols for adapting expensive cryptographic protocols such as fully homomorphic encryption and secure multiparty computation for use in a smart grid's advanced metering infrastructure. To disguise the meter reading data from smart meters, the proposed solution encrypts it using fully homomorphic encryption, or computes and distributes its shares using secure multiparty computation. Without revealing the true value of the readings, the encrypted data/computed shares are aggregated in a hierarchical fashion at certain aggregator smart meters up until the network's gateway. However, a secure billing system for smart meters is not supported, besides using fully homomorphic encryption and secure multiparty computation is very expensive. We focus on designing a secure billing system, avoid expensive cryptographic protocols in our design, and rely on lightweight secret sharing schemes. This design choice is reflected in the computational time, e.g., in the scheme of [35], for 100 smart meters, the average data collection time for obtaining all smart metering data from all aggregators at the gateway in a *single* round, is roughly 40 seconds. Our scheme has much less computational overhead where the execution time of most our protocols is a fraction of a second, and the maximum overhead is 13 seconds. We discuss our experiments in Section 6.

There have been other privacy-preserving billing schemes proposed that make use of compute-intensive protocols such as zero knowledge proofs, homomorphic encryption, and/or secure multiparty computation [6, 11, 19, 22, 24, 29]. Since smart meters are resource-constrained devices, our goal is to avoid expensive cryptographic protocols and investigate what can be achieved with relatively inexpensive but secure protocols and primitives, such as secret sharing schemes, hash functions and digital signatures. Wagh *et al.* [37, 38] proposed a framework that uses a combination of secret sharing and compute-intesive secure multiparty computation. However, their frameworks only partially protect user's privacy.

Trusted aggregator/blockchain based schemes: There have been blockchain-based solutions proposed, such as the work by Guan *et al.* [14], where the customers are divided into groups and each group has its own private blockchain. At every time interval, a certain customer is selected in a group as a mining node. The mining node aggregates the energy consumption of the customers and writes the aggregate reading on its group's private blockchain, from where the utility provider can read the aggregate reading. Besides the impracticality of groups of users having separate private blockchains, in this approach, the mining nodes will be able to read customers' fine-grained energy consumption data in plaintext. Our focus is on preserving the privacy of the customer's fine-grained energy consumption data.

Chen *et al.* [8] proposed a scheme for aggregation of electricity consumption data (which is recorded at pre-defined time intervals) by a smart meter. In this model, there is a trusted authority that sets up the system, keys all customers, and a control center which is composed of a set of servers. The control center is run by the utility provider, and is assumed to be trusted. In our work, we do not use a trusted authority or a set of computationally powerful parties. There have been smart meter data aggregation schemes proposed, which aggregate the data generated by several smart meters in certain pre-defined time intervals, and report them to the utility provider, such as [2, 20]. Knirsch *et al.* [20] proposed an approach for privacy-preserving data aggregation, where all smart meters are connected in a ring-shaped topology, with a designated aggregating entity called *data aggregator*. However, the proposed scheme is vulnerable to collusion between the data aggregator and malicious smart meters. Bao *et al.* [2] proposed a privacy preserving data aggregation scheme similar to the one by Knirsch *et al*; in their model, there is a trusted authority and a *control center* to which the customer's data is reported, and a gateway that acts as the link between the customers and the control center. They make use of the Boneh-Goh-Nissim cryptosystem [5] for encrypting the customer's consumption data over a given time period and report it to the control center in a secure manner. The trusted authority is responsible for generating the public and private keys used for encryption and decryption. The consumption data of multiple customers is aggregated and encrypted over the given time period and is sent to the control center. Though we make use of a data aggregator in our work, this entity need not be trusted and we also make sure that the data aggregator has no access to any sensitive data, thereby protecting the privacy of the customer.

Secret sharing-based schemes: Wagh et al.[36] proposed a framework based on secret sharing, but need a trusted third party that can collect plaintext readings from different smart meters, aggregate them and pass them along to the utility provider, thus revealing customers fine-grained readings to the third party. Similar ideas have been explored by Rottondi *et al.* [30], who introduce *privacy preserving nodes*, that collect a share of measurements from each smart meter in a neighborhood, and transmit the shares to the utility provider. The utility provider then reconstructs the secret back to get the aggregated reading for all the smart meters. However, the utility provider can learn about the exact energy consumption habits of customers.

Masking: Gope *et al.* [13] proposed a privacy-preserving aggregation scheme for billing and load balancing management using masking. The authors use random values to mask the individual reading of smart meters. However, in their work, the service provider learns the fine-grained readings of the users. Liu *et al.* [21] proposed a data aggregation scheme without any TTP. Their model masks the metering data of individual users. However, they use homomorphic encryption for performing addition operations on the ciphertexts, which is computationally expensive to perform on resource constrained devices such as smart meters.

In contrast to the prior work in this area, we aim to avoid compute-intensive cryptographic protocols in smart meters, hide a customer's fine-grained energy consumption data from a utility provider and any third parties, yet enable the utility provider to bill the customer accurately.

3 SYSTEM MODEL

In this section, we describe our system and adversary models. Our proposed system consists of the following parties:

(1) **Smart Meter (SM)**: A smart meter is an electronic device installed at a customer household that can monitor and record energy consumption readings at pre-determined time intervals (e.g., fifteen minutes, hourly, daily, etc.). Additionally, a smart meter is responsible for reporting the energy consumption readings of a customer to a utility provider. In our work, we assume standard smart meters equipped with a 32-bit CPU with a maximum operating frequency of 50 MHz, that support multiply/divide/multiply-and-accumulate instructions, and have an on-chip flash memory of up to 512 KB [33]. Moreover, we assume the smart meter can interact with system entities through a distributed file system. We refer to a user in whose home a smart meter is installed as a *customer*, and the smart meter itself as the *customer smart meter*.
(2) **Leader:** A leader is a smart meter that is chosen by the utility provider to verify and report the aggregate energy consumption readings of the other smart meters in the system to the utility provider, in exchange for a possible energy rebate or a service fee. During system initialization, the utility provider broadcasts a message for the election of the leader in the system; any smart meter can volunteer to become the leader.
(3) **Peers**: In our proposed system, each smart meter belonging to a unique customer constructs a polynomial to generate shares of its energy consumption readings, blinds them, and distributes the blinded shares to its peer smart meters. Peers are responsible for reconstructing the aggregate blinded energy consumption readings, and reporting the blinded aggregate energy consumption reading to the leader. We assume that the number of peers is equal to the number of energy consumption recorded readings. The number of readings recorded depends on the threshold time period set by the utility provider. This is a system parameter that the utility provider chooses depending on the current work load in the system.

(4) **Utility provider (UP)**: This entity is responsible for delivering electricity to the customers and generating the customers' bill based on the received aggregate energy consumption reading. The utility provider sets up the system parameters such as time intervals at which energy readings need to be recorded, and time period for reporting aggregate readings, among others. In our proposed system, we assume that the utility provider interacts with system entities through a distributed file system.

(5) **Distributed file system**: We assume the existence of a peer-to-peer distributed file system, e.g., Interplanetary File System (IPFS), using which all entities can share information. The distributed file system provides a high throughput content-addressed block storage model, with content-addressed hyperlinks [3]. In this work, we do not use a blockchain due to its high gas fees per KB of storage cost. We compare costs of using a distributed file system vs. using a blockchain in Section 8. Once data is written to the IPFS, it returns hash digests that can be used to access the data. These hash digests are fixed in size (32 bytes), hence writing data to the IPFS and storing these hashes on a blockchain is more economical than storing the data directly on the blockchain. We briefly analyze the costs of writing to IPFS vs. blockchain in Figure 4.

We now describe the workflow of our system as depicted in Figure 1. Let us consider a group of customer smart meters $SM_1, \ldots, SM_n$, a leader SM_L, a utility provider, and a distributed file system. In Step 1, the utility provider broadcasts the billing time period, e.g., a month, and the time intervals to record the energy consumption reading by customer smart meters, e.g, every fifteen minutes. The utility provider also selects a leader SM_L; in Figure 1, SM_1 has been chosen as SM_L. In Step 2, each smart meter records its energy consumption reading for the time intervals, blinds them using a suitable blinding factor, and computes the aggregate blinded reading. Each smart meter then creates shares of the blinded reading using a k-of-n secret sharing scheme, e.g., Shamir's secret sharing scheme. For example, SM_2 creates n shares $Share_{2,1}$,...., $Share_{2,n}$, distributes each share to SM_1, SM_3,...., SM_n respectively. The peers come together to reconstruct the signed aggregate blinded reading and send it to SM_L. SM_L verifies the broadcasted aggregate blinded reading and computes the aggregate signature of all the peers. In Step 3, SM_L writes the aggregate blinded reading and aggregate signature to a distributed file system, and broadcasts the returned digests from the file system, which is used to reference and retrieve the data in future. In Step 4, the utility provider queries the file system using the broadcasted digest values in Step 3 to retrieve the aggregate reading and aggregate signature. Finally, the utility provider verifies the aggregate signature and aggregate blinded reading and computes the energy bill for the customer smart meters.

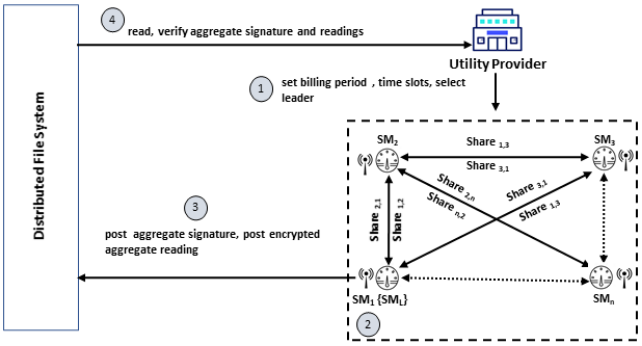

Figure 1: System model

4 ADVERSARY MODEL

We assume that all parties have access to a distributed file system that is always available, e.g., IPFS [3]. We now outline the trust assumptions on the parties:

- We assume that customer smart meters are tamper-evident, and have the capability to perform simple arithmetic operations as described in [12].
- The energy consumption data, once recorded by the customer smart meter, should not be modifiable by anyone in the system. In our system we expect the customer smart meter to record the readings in an honest and fair manner.
- The leader smart meter can be potentially malicious, in that it might deviate from the protocol steps at any point. The leader can potentially collude with other malicious parties in our system, such as peers and the utility provider: we discuss different collusion scenarios between the leader and other malicious parties, and how our system handles them in Section 7.
- The peer smart meters can turn malicious at any point. They may arbitrarily deviate from described protocol steps, try to inject false information into the system, contribute wrong shares, or might altogether refuse to participate in secret sharing protocols. In the worst case, our system can tolerate up to $(n-1)$ corruptions among n peers. Although, in practical deployment scenarios, the number of corrupted peers will be likely far less.

4.1 Security & Privacy goals

We identify the following security and privacy goals for our system:

- A customer's fine-grained energy consumption data should only be readable by the customer themselves, and should be kept private from all other parties. It should be infeasible for the utility provider, leader, and peers to obtain information about any customer's fine-grained electricity usage habits.
- All data exchanged needs to be authenticated as originating from a verified source.

5 OUR CONSTRUCTION

In this section, we first describe cryptographic preliminaries, followed by a brief technical overview of our system, and then describe our construction and its component algorithms.

4

5.1 Cryptographic Preliminaries

We use Shamir secret sharing [32], which is a threshold scheme used to split a secret value into n pieces called *shares*, such that any $k \leq n$ parties can reconstruct the secret. We also use aggregate signatures [4]. An aggregate signature scheme compresses multiple signatures by different parties into a single signature, thus reducing the overhead of signature verification.

5.2 Technical Overview

At a high level, in our system, each customer smart meter records and stores the electricity consumption readings for pre-defined time intervals of a billing period, e.g., fifteen minutes for a monthly billing period. Following this, the customer smart meter blinds each time interval reading by multiplying it with a blinding factor chosen from $\mathbb{R}^+$. Note that adding the readings for all time intervals will give us the total electricity consumption for the entire billing period. Then the customer smart meter encrypts the aggregate blinded reading with a key that is shared between the leader smart meter, SM_L, UP, and the customer smart meter. The blinded energy consumption readings per time interval are secret shared by the customer smart meter by individually sending them to the peers. The peers come together to reconstruct the aggregate blinded consumption reading for the billing period, which is the sum of all the individual blinded readings generated by the customer smart meter, i.e., it is the constant term of the reconstructed polynomial. The peers then encrypt the reconstructed aggregate blinded reading with the public keys of the leader and the utility provider, respectively, sign the corresponding ciphertexts generated, and broadcast them in the system. The leader retrieves this value, verifies the signatures of all the peers that took part in the interpolation process, checks if the aggregate blinded reading matches with the one reported by the customer smart meter to the leader. Upon successful verification, the leader gathers all the signatures of the peers into an aggregate signature. This aggregate signature along with the all the ciphertexts by all the entities until this point are collected by the leader and are written to a distributed file system, such as IPFS. The file system returns the digests for the cotent written. These digests are collected into a set by the leader and it is broadcasted in the system. The utility provider then uses this set of digests and reads these values from the distributed file system and calculates the aggregate unblinded consumption reading which is used to send a bill to the customer smart meter for that billing time period.

5.3 Algorithms

Our system is made up of five algorithms that handle different functions. The first algorithm is Setup, which configures the various system parameters and generates the smart meters' signature and verification keys. The second algorithm is the Blinded Reading Generation algorithm, and it involves the customer smart meter recording energy consumption data, computing the aggregate blinded consumption reading, encrypting and signing the aggregated blinded energy consumption readings, and broadcasting the result. The third algorithm is Interpolation, which deals with the distribution of blinded individual consumption readings as shares, the interpolation procedure to reconstruct the aggregate blinded reading, encryption of the aggregated blinded reading, and broadcasting the ciphertext,

Algorithm 1: Setup

Input : λ
Output: T,τ, $\mathrm{SK}_1 \ldots \mathrm{SK}_n$, $\mathrm{VK}_1 \ldots \mathrm{VK}_n$, $\mathrm{K}_1 \ldots \mathrm{K}_n$,$\mathcal{T}$
Parties: UP, $\mathbb{SM}$, SM_L

begin
  for $i = 1; i \leq |\mathbb{SM}|$; i++ **do**
    $\mathsf{Key}(1^\lambda) \rightarrow K_i$
  end
  /* Steps 5 - 9 are run by all the smart meters */
  for $i = 1; i \leq |\mathbb{SM}|$; i++ **do**
    $\mathsf{Agg.KeyGen}(1^\lambda) \rightarrow (SK_i, VK_i)$
  end
  UP picks T and τ, broadcasts in the system and picks $SM_L \in \mathbb{SM}$ as the leader, SM_L picks a value of $\mathcal{T}$ for the system
end

which contain the encryption of the interpolated value by the peers. This encryption is done twice, once with the public key of the SM_L and also with the public key of the utility provider. In the fourth algorithm, the SM_L verifies the signatures created by the peers on the interpolated value and also verifies the equality between the aggregate blinded consumption reading generated by a customer in Algorithm 2. The SM_L is also responsible for constructing the aggregate signature from the individual signatures of the peers on the interpolated value, and writing all ciphertexts to the distributed file system.

The UP uses the hash digests which were generated by the distributed filesystem, to retrieve the ciphertexts containing the aggregate unblinded consumption reading, which was generated by both the peers and the customer smart meter, and the aggregate signature generated by the SM_L. The UP then proceeds to verify the equality between the aggregate unblinded consumption reading generated by the customer smart meter and the peers, and if they are found to be equal, UP proceeds to calculate the bill for the customer. For ease of reference, we give a table describing the notations used in our algorithms in Table 1.

We now describe the details of our algorithms.

1) Algorithm 1, Setup: During system initialization, all customer smart meters call the $\mathsf{KeyGen}(1^\lambda)$ and $\mathsf{Agg.KeyGen}(1^\lambda)$ functions, which take a security parameter λ as input to obtain the shared keys K_i , and a signing/verification keypair (SK_i, VK_i) respectively. The customer smart meter and the utility provider both utilize the shared key K_i. The shared key K_i is used to encrypt the customer's aggregate blinded consumption reading as well as the blinding factor the customer smart meter applied. The signing and verification keys generated as a part of this algorithm are used for signing and verification processes in Algorithms 2,4,5. The UP picks a smart meter to be the leader, SM_L. Any node can volunteer to be a SM_L. In the case of multiple nominations, the customer which has had the longest association will be selected as the SM_L. Then the UP sets the billing time period, T, e.g., a month, and time intervals, τ, at which readings need to be recorded, e.g., fifteen minutes, a few hours, or

Table 1: Notations Used

Notation	Description
SM_i	An arbitrary smart meter, i
n	Total number of smart meters in the system
$\mathbb{SM}$	The set of all smart meters in the system
SM_L	Leader smart meter.
UP	Utility Provider
$\text{Sign}_{\text{SK}_i}$	Signing function
$\text{Verify}_{\text{VK}_i}$	Verification function
SK	Signing key
VK	Verification key
BF	Blinding factor
τ	Threshold for reporting
T	Billing time period
t_s	Starting timestamp
t_e	Ending Timestamp
IPFS.Read()	IPFS read operation
IPFS.Write()	IPFS write operation
δ	t_e - t_s
$\mathcal{T}$	Timeout for reporting the readings to the utility provider

days. As a part of this algorithm, the SM_L also sets a timeout value $\mathcal{T}$, which is the maximum time allowed for all entities (except the customer) in the system to finish the protocol execution. This value helps us prevent any intentional DoS attacks in our model. This timeout value is a system parameter and is subject to change based on the current workload of the system.

2) Algorithm 2, Blinded Reading Generation: All smart meters begin recording their energy consumption readings using the interval, τ, that UP selected in Algorithm 1. To ensure that readings are not recorded too frequently and the system is not overwhelmed, each customer smart meter selects a time interval value δ such that $\delta \geq \tau$. Each customer smart meter selects an initial timestamp value t_s and begins recording the consumption data. Each reading is recorded for a duration of δ. t_e represents the ending timestamp for each reading being recorded. Hence δ is the difference between the values of t_s and t_e. To authenticate the readings, each customer smart meter signs the readings recorded with their signing keys generated from Algorithm 1, with the aggregate reading denoted as a_0. The readings are then blinded by multiplying them with a blinding factor BF. Once these blinded readings are generated, they are recorded in a set $\mathbb{SH}$. The ending timestamps, t_e of these blinded readings are hashed using a collision-resistant hash function to produce unique digest values for each time stamp. This is done for two reasons: to protect the privacy of the customer's individual consumption habits, and to make the process of interpolation easier. An ordered pair consisting of the hashed time stamps and the blinded readings are recorded in a set $\mathbb{SH}$. The reason for the digests of timestamps being recorded in this set is to make use of these values in the interpolation process. These blinded readings are signed by the customer's smart meter using the signing key generated in Algorithm 1 to preserve the integrity of the readings recorded, and the signatures are recorded in a set $\mathbb{G}$. The signatures from this set $\mathbb{G}$ are later on verified by the peers in the system during the interpolation process. Finally, the a_0 and the BF are encrypted with K_i, which is the shared key generated in Algorithm 1, and the corresponding ciphertext C is signed by the smart meter. A tuple consisting of C and the signature on it is broadcasted in the system.

Algorithm 2: Blinded Reading Generation

Input : $T, \tau, H : \{0,1\}^* \rightarrow Z^+$
Output: $\mathbb{G}$, a_0, $\mathbb{S}'$, $\mathbb{S}$, $\mathbb{SH}$,C
Parties : $\mathbb{SM}$

Every SM $\in \mathbb{SM}$ runs this algorithm in parallel
begin
  Pick $\delta \geq \tau$, BF
  /* τ = threshold for reporting */
  $\mathbb{G} = \varnothing$, $\mathbb{S} = \varnothing$, $\mathbb{S}' = \varnothing$ and $\mathbb{SH} = \varnothing$
  $a_0 = 0$, $t_s = 1$
    /* t_s and t_e are the starting and the ending time stamps. */
  for $i = t_s; i \leq T; i$++ **do**
    $t_e = t_s + \delta$, Record $\mathbb{S} = \mathbb{S} \cup (t_e, Y_i)$
    $t_s = t_e + 1$
      /* Y_i' is the blinded reading. BF = blinding factor */
    compute $Y_i' = Y_i * BF$, Record $a_0 = a_0 + Y_i'$, Record $\mathbb{SH} = \mathbb{SH} \cup (H(t_e), Y_i')$, Agg.$\text{AggSign}_{\text{SK}_i}(Y_i') \rightarrow \sigma_i$, $\mathbb{G} = \mathbb{G} \cup (\sigma_i)$
  end
  $\text{Encrypt}_{\text{K}_i}((a_0 || BF)) \rightarrow C$, $\text{Sign}_{\text{SK}_{\text{SM}}}(C) \rightarrow \sigma_C$, Broadcast $(C, \sigma_C, \mathbb{G}, \mathbb{SH})$ in the system
end

3) Algorithm 3, Interpolation: The peers and the customer smart meters run this algorithm. Each customer smart meter constructs the secret-sharing polynomial, in which the smart meter selects random coefficients $a_1, ...a_{n-1}$, and sets the value of a_0 to be equal to the sum of aggregate of the blinded consumption readings. The customer smart meter splits the aggregate blinded consumption reading into $n-1$ shares using the constructed secret-sharing polynomial and sends each share to a corresponding peer smart meter. Initially the customer smart meter constructs a mapping polynomial that maps the digests of the ending time stamps produced in Algorithm 1 to the corresponding blinded consumption reading. The value of the term with no coefficient is set to the aggregate unblinded consumption reading recorded by the customer. The shares produced by the customer are recorded in the set $\mathbb{SH}$ in Algorithm 1. This set is used by the peers to perform the interpolation process. Since Shamir secret sharing is a threshold secret sharing scheme, only k out of the total n shares will be needed for successful interpolation of the secret value. To guarantee accurate delivery of shares, e.g, preventing shares from getting lost/dropped along the way, *publically verifiable secret sharing* proposed by Stadler [34]

can be used. Although this was part of our initial design prototype, we ultimately decided against using it due to efficiency reasons. The peers initially verify the signature of the customer smart meter on these shares. Upon successful verification, the interpolation is performed on the shares present in the set $\mathbb{SH}$. Once the interpolation process is completed with the calculation of the reconstructed secret value which is the aggregate unblinded consumption reading generated by the customer, this secret value is signed by all the peers that took part in the interpolation process to preserve the integrity of the interpolated value. This data in then encrypted with the public key of the SM_L and the UP. This encryption is done to prevent un-authorized modification to the value generated by the peers. The resulting cipher texts C_1 for the leader and C_2 for the UP are broadcasted in the system.

Algorithm 3: Interpolation

Input : $\mathbb{S}'$,$\mathbb{G}$,$\mathbb{SH}$
Output: $C_1.C_2$
Parties : $\mathbb{SM}$

begin
  /* The customer smart meter runs the steps 2-5 */
  a'_0= 0 , $reading$ = 0 , $\mathbb{IG}$ = $\varnothing$
  for *i=1;i* $\leq$ $|\mathbb{SH}|$ *; i++* **do**
    n = $|\mathbb{SH}|$-1 construct $p\langle x\rangle = a_0 + a_1x + a_2x^2 \ldots a_{n-1}x^{n-1}$ such that $p\langle x\rangle = Y'_i$
  end
  /* steps 6-19 are run by SM_i; i $\in$1 ... $|\mathbb{SH}|$ */
  for *i = 1;i*$\leq$ $|\mathbb{SH}|$ *;i++* **do**
    Retrieve σ_i from $\mathbb{G}$, Retrieve Y'_i from $\mathbb{SH}$
    if (Agg.AggVerify$_{\mathsf{VK_i}}$(Y'_i,σ_i) $\rightarrow$ *1*) **then**
      Interpolate Y'_i to compute $a'_0 = p\langle 0\rangle$
    end
    calculate a'_0
  end
  if a'_0=$\sum_{i=1}^{T} Y'_i$ **then**
    for *i=1;i* $\leq$ $|\mathbb{SH}|$ *; i++* **do**
      Agg.AggSign$_{\mathsf{SK_i}}$(i || a'_0) $\rightarrow$ σ_i , $\mathbb{IG}$ = $\mathbb{IG} \cup (a'_0, \sigma_i)$
    end
    Encrypt$_{\mathsf{PK_{UP}}}$($\mathbb{IG}$) $\rightarrow$ C_1, Encrypt$_{\mathsf{PK_{SM_L}}}$ ($\mathbb{IG}$) $\rightarrow$ C_2 , Broadcast C_1, C_2 in the system
  end
end

4) Algorithm 4, Final Reading Calculation: This algorithm is run by the SM_L in the system. The leader first decrypts the cipher text C_1 with his secret key and verifies the signatures on the interpolated value produced by the peers using their signing keys. If the signatures are not successfully verified, it means that one of the peers deviated from the protocol steps and acted in a malicious manner. Upon successful verification, the leader verifies the equality between the aggregate unblinded consumption reading generated by the customer smart meter and the value that is generated by the peers during the interpolation process. This is done to ensure that there is consistency in the values reported by the customer and the values interpolated by the peers. Upon successful verification of this equality, the leader generates the aggregate signature from all the individual signatures of the peers. This is denoted by α. The use of aggregate signatures is to conserve the bandwidth and reduce the time of verification of the signatures of the peers for the utility provider. The aggregate signature and all the cipher texts generated and broadcasted by all the entities in the system until this point $C, C_1, C_2, C_{a_0'}$ are written to the distributed file system, which gives back the digest as a content address. SM_L then broadcasts the digest in the system, which can be used by UP to retrieve data stored in the set $\mathbb{H}$. This set is then broadcasted to all the parties in the system.

5) Algorithm 5: The steps of Algorithm 5 are run by the utility provider. The provider reads the ciphertexts $C_2, C_{a'_0}, C$ from the distributed file system using the set of digests broadcasted in the system by the leader. C is then decrypted with the shared key K_i to retrieve the values of a_0 and BF. The signature on a_0 by the customer smart meter, is then verified. Upon successful verification, the aggregate signature α is read and the ciphertext $C_{a'_0}$ is decrypted using the shared key to reveal the value of a_0. Finally, the values of a_0 and a'_0 are compared. If they are found to be equal, the aggregate unblinded consumption reading is calculated by dividing a_0 with BF and the corresponding bill is generated for the customer.

Algorithm 4: Final Reading Calculation

Input : C_1, C_2
Output: $C_{a'_0}$, α,$\mathbb{H}$
Parties : SM_L

begin
  /* SM_L runs the following steps */
  $\mathbb{H} = \emptyset$
  Decrypt$_{\mathsf{SK_{SM_L}}}$ (C_1) $\rightarrow$ $\mathbb{IG}$
  for *i=1;i* $\leq$ $|\mathbb{IG}|$*;i++* **do**
    if Agg.AggVerify$_{\mathsf{VK_i}}$(a'_0,σ_i) $\rightarrow$ *0* **then**
      return $\perp$
    end
  end
  if Agg.AggVerify$_{\mathsf{VK_i}}$(a'_0,σ_i) $\rightarrow$ *1* **then**
    if a'_0= a_0 **then**
      Agg.Aggregation($\sigma_1, M_1 = 1||a'_0 \ldots$ ($\sigma_{|\mathbb{IG}|}, M_{|IG|} = |IG|||a'_0$) $\rightarrow$ α
      Encrypt$_{\mathsf{K_i}}$ (a'_0) $\rightarrow$ $C_{a_0'}$, IPFS.Write ($C, \sigma_C, C_1, C_2, C_{a'_0}, \alpha$) $\rightarrow$ $H_1, H_2, H_3, H_4, H_5, H_6$
      $\mathbb{H} \cup \{H_1, H_2, H_3, H_4, H_5, H_6\} \rightarrow \mathbb{H}$
      Broadcast $\mathbb{H}$ in the system
    end
  end
end

6 IMPLEMENTATION

In this section, we describe the environment used to test our system and provide the experimental results for the performance of

Algorithm 5: Bill Generation

Input : $C, C_1, C_{a_0'}, K_i, \alpha, \mathbb{H}$
Output : Bill for the customer
Parties : UP

begin
  IPFS.Read($\mathbb{H}$) $\rightarrow C_1, C_{a_0'}$
  $\text{Decrypt}_{\text{K}_\text{i}}$ $(C) \rightarrow (a_0 || \ BF)$
  if $\text{Verify}_{\text{VK}_{\text{SM}_\text{i}}}$ $(a_0, \sigma_i) \rightarrow 1$ **then**
    IPFS.Read(α), $\text{Decrypt}_{\text{K}_\text{i}}$ $(C_{a_0'}) \rightarrow a_0'$, $\text{Decrypt}_{\text{SK}_{\text{UP}}}$ $(C_1) \rightarrow \mathbb{IG}$
    if Agg.Verification $(\alpha, (M_1, VK_1), \ldots, (M_{|\mathbb{IG}|}, VK_{|\mathbb{IG}|})) \rightarrow 1$ **then**
      if $a_0' = a_0$ **then**
        $reading = \frac{a_0}{BF}$
      end
    end
  end
end

different system entities: utility provider, customer smart meter, peers, and leader. Our algorithms were implemented in Python and evaluated on a platform equipped with Intel(R) Xeon(R) CPU clocked at a 2.00GHz processor, 13 GB RAM.

Table 2: Number of operations performed per billing period per user for all our algorithms. T is the billing time period and δ is the time interval for recording the readings.

Operations	At SM	At Peers	At SM_L	At UP
Polynomial construction	1	–	–	–
Polynomial interpolation	–	1	–	–
Encryptions	1	2	1	–
Exponentiations	–	–	–	–
Signatures	(T/δ) +1	(T/δ)	1	–
Products	(T/δ)	–	–	–
Pairings	–	–	–	–
Hashes	(T/δ)			

Table 3: Comparison of our scheme and Saxena *et al.* [31]. Here, n is the total number of smart meters in the system.

Operations	Our Scheme	Saxena *et al.* [31]
Random number generation	–	2n+5
Polynomial construction	1	–
Polynomial interpolation	1	–
Exponentiations	–	7n+13
Encryptions	4	–
Products	(T/δ)	4n
Pairings	–	n+5
Hashes	(T/δ)	2n+6
XOR	–	2n
L function	–	n+1
Signatures	2(T/δ)+1	–

We implement cryptographic primitives using a cryptography library in Python [28]. The aggregate signature scheme implementation is built on the top of the *Petlib* library in Python [26], and also uses the *Pblib* library which supports bilinear pairings [9]. In our implementation, the value of the billing period, T, and the time intervals within each billing period, τ, are established by the utility provider. This value serves as the minimum threshold time period for the smart meters to record readings, e.g., if the value of τ is set to one day, it means that the time interval in which the smart meters record their consumption readings, δ, should be greater than or equal to one day. To model a realistic scenario, we have set the values T and τ and δ to be one month, three days, and one day, respectively. All smart meters records their energy consumption data every three days and send the data to the utility provider at the end of the month, at which point the bill is generated.

Table 2 depicts the type and the number of operations performed by each party in our system over the course of the entire billing period. Table 3 summarizes the number of operations performed by our proposed scheme and the one proposed in [31]. While maintaining better operational overhead, our scheme does not require any entity to be trusted.

As shown in Table 4, the peers have the highest execution time, which increases linearly with the number of peers because the peers execute several secret-sharing related functions such as interpolation, computation of the blinded secret value, signing, encryption, and writing to the IPFS. The degree of the secret-sharing polynomial is one less than the number of smart meters. The main contributing functions of the utility provider are Verify, AggVerify, and computing the aggregate reading. To make the system more efficient, the degree of the polynomial, the number of peers, and the execution time of entities can be reduced by increasing the value of τ.

Table 4: Execution Time for for Algorithms 4 & 5

# of Smart Meters	Execution time (in sec) for steps run by parties			
	Smart meters	Peers	SM_L	UP
10	0.42664	1.73602	0.83343	0.8838
20	0.90201	2.49307	0.16605	0.18146
50	2.29235	6.36160	0.43392	0.48573
100	4.74029	13.17859	0.95056	0.98683

The main contributing factors to the execution time of the leader are Verify, AggVerify, Encrypt functions. The execution time of Algorithm 1 and Algorithm 2 are shown in Table 5. As shown in Table 5, the AggVerify function has a higher execution time because it is a compute-intensive function (including several pairing functions), which grows linearly in the number of smart meters in the system. Algorithm 2 has a higher execution time because each customer smart meter records the energy consumption data in each time interval, calculates the blinded reading value, hashes the time intervals values, encrypts, and signs the aggregate reading. Algorithm 1 has a lower running time since it just generates the shared key between the entities, and the secret keys and verification keys of entities.

Table 5: Execution Time of Algorithm 1, Algorithm 2, AggSign **and** AggVerify

# of Smart meters	Execution time (in sec)			
	Algorithm 1	Algorithm 2	AggSign	AggVerify
10	0.01972	0.51779	0.00725	0.07606
20	0.03639	1.12124	0.01739	0.15168
50	0.10365	2.8868	0.06904	0.38649
100	0.27932	6.08485	0.22432	0.73501

7 SECURITY ANALYSIS

In this section, we describe the security analysis of our system. Specifically, how the different parties in the system can turn malicious and the mechanisms we have in place to handle the malicious behavior. As we have stated in our assumptions in Section 3, only the utility provider is honest-but-curious. The other parties in our system can turn malicious as shown in Figure 2 and Figure 3

Passive attacks:

(1) **(n-2) Peers are compromised:** The adversary can compromise (n-2) peers and obtain their shares. In this case, the adversary can reconstruct the secret value, but the adversary cannot learn anything about the real individual energy consumption of a smart meter because each individual smart meter masks their energy consumption reading with a blinding factor.

(2) SM_L **is compromised :** In our system model, the adversary can compromise SM_L and can obtain the individual energy consumption readings of smart meters. But the adversary cannot learn anything regarding the individual consumption habits of the customer smart meter because all of the individual energy consumption smart meters are blinded values.

Active attacks:

(1) **Malicious behavior by peers:**
The peers in our system can turn malicious and misreport their shares. If this were to happen, the interpolated secret value which also happens to be the aggregate blinded consumption reading, a_0' will be corrupted. This is depicted in the Figure 2. It can be inferred from Figure 2 that if this were to happen, the interpolated secret value would be incorrect (step 3) This will eventually be caught by the UP and the SM_L when they compare the value of a_0 , which is sent by the customer smart meter with that of the incorrect a_0' , which is calculated by the peers in steps 5 and 6 respectively.

(2) **Malicious behavior by the leader:**
   (a) If the leader claims that the AggVerify function (Algorithm 4, line 5) does not pass as in Figure 3, this will be caught by the interpolating smart meters since they have access to their verification keys and can eventually verify the signatures by themselves.
   (b) If the leader calculates a wrong aggregate signature in Algorithm 4 line number 13, or in Algorithm 5 line number 5, in step-1 of the figure. The UP eventually will handle it by verifying the aggregate signature, as shown in step 4 of Figure 3.
   (c) The leader can write wrong values on the distributed file system and broadcast the corresponding digests in the system in Algorithm 4 line number 15 as shown in step 1 of Figure 3. But, since the digests are publicly available, any entity eventually in the system can verify them or the utility provider verifies their correctness in step 4. The leader can also modify the ciphertexts created by the peers during transmission to the system in step 1. If this were to happen, the adversary would get caught in Algorithm 5 line number 3. When the utility provider decrypts them to generate the bill for the customer. The mangled cipher text will not be equal to the interpolated value calculated by the peers in step 5 in figure 3
   (d) The adversary can also mangle the cipher text $C_{a_0'}$ during its construction. This would be caught during its decryption in Algorithm 5 line number 5. The leader can collude with the utility provider and can report an incorrect bill to the customer. This can be eventually discovered and disputed by the customer, who knows the amount of electricity consumed, and the per unit cost.
   (e) The leader can collude with one or more peers and can cause the peers to produce invalid signatures on the interpolated secret value (Algorithm 3, line 15). If this were to happen, the malicious peers and the leader would be eventually caught, because the peers sign the interpolated secret value by concatenating it with their unique identifier. This would help us in identifying exactly which peers were compromised. Finally, a compromised leader can potentially write incorrect values of cipher texts to the IPFS file system; since the hashes are publicly broadcasted, the

values written are publicly auditable. This is depicted in Figure 3, step 4.

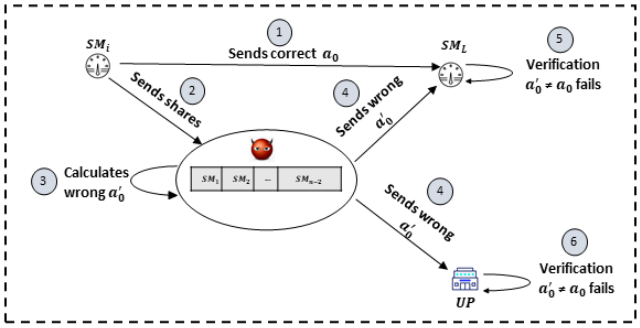

Figure 2: Potential Malicious Activity by the Peers

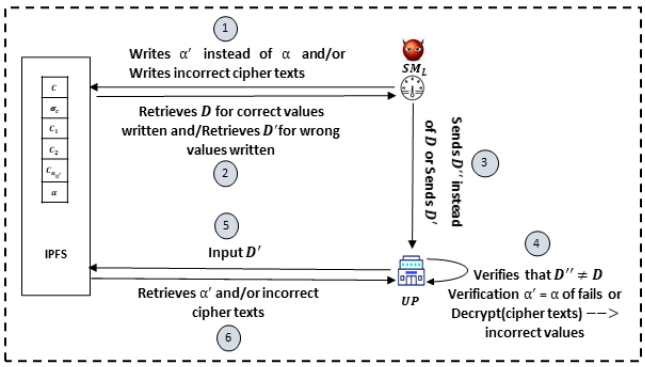

Figure 3: Potential malicious Activity by the SM_L

8 DISCUSSION

In this section, we discuss the limitations of our system and a few feasible methods to address them.

1) Using a unique blinding factor for every recorded reading: The blinding factor, *BF* chosen in Algorithm 2 is common to all readings recorded by a particular smart meter over a given billing time period. Even though the readings are blinded, the peers can still try to guess a pattern from the blinded readings reported to them in the same range for a given time period. In order to minimize this, the customer smart meters can use a unique blinding factor for each time interval reading they create and the tuple containing the blinding factors can be shared with the utility provider. In practice, the blinding factor can be generated using a pseudo-random number generator.

2) Paid IPFS versus vanilla IPFS: The choice of what version of a distributed file system to use is implementation-dependent. For example, the free version of IPFS is susceptible to garbage collection after twenty-four hours, specifically, IPFS nodes can delete data for which there is not enough demand if the data is not periodically updated. In our system, since the energy consumption data once recorded cannot be changed, updates to the IPFS are unlikely in our application and hence the data could get deleted.

Figure 4: Storage costs for IPFS and blockchain

(a) Storage cost for 1 KB of data on Ethereum blockchain

1 KB = 8000 bits = 625,000 gas
Gas price = 175 gwei
Total cost = Gas used × price
Total cost = 625,000 × 175 = 109,375,000 Gwei
Total cost in USD = $\frac{price(ETH) \times Total\ gas\ cost}{1{,}000{,}000{,}000}$

Total cost in USD = $\frac{4346.17 \times 64{,}375{,}000}{1{,}000{,}000{,}000}$

Total cost in USD = $475.36

(b) Storage cost for one IPFS hash on Ethereum blockchain

1 KB = 8000 bits
32 bytes = 256 bits
Total gas needed = 20,000
Gas price = 164 Gwei
Total Cost = 20,000 × 0.00000018 ETH = 0.0036 ETH
Total cost in USD for 1 digest = $16.92
Total cost in USD for writing 6 digest = 16.92 × 6 = $101.52

A solution to this is to use a paid version of IPFS, e.g., Pinata [27]. Since there is a cost involved, Pinata provides permanent storage of data on the IPFS system. The price for the service needs to be paid upfront and users can get storage of 1 GB for a price of 15 cents. This solution could be leveraged if permanent data storage is a requirement. If one uses this, the utility provider can pay the storage cost upfront and later retrieve it from the customers as a part of their monthly energy bills.

3) **Distributed file system vs. blockchain:** For our proposed model, we used a distributed storage system, e.g., IPFS, to store the data. Using such a system achieves high throughput, high storage capacity, and low latency by establishing content service and provides a storage model representation based on the block with the content address, low cost, no single point of failure, and is immutable, hence it cannot be altered by an adversary. This is better in many ways compared to using a centralized authority to store data such as a cloud server. An alternative to using a distributed file system is to store all data on a blockchain that all parties have access to. In our system model, we prefer to use a distributed IPFS system to store the data instead of a blockchain since prior studies have shown that using blockchain technology to store the data is expensive [15].

9 CONCLUSION

In this paper, we have presented a novel construction for secure and private billing of energy consumption data from smart meters using distributed file systems without the need for a trusted third

party. We show that apart from the aggregate consumption data which is used for generating the bill, no other information about the user is known to the utility provider. In the future, our system could be extended to incorporate energy profiling techniques that might be of use in data analytics applications As a part of our future work, we plan to prove the security of our proposed system in the universal composability framework [7].

ACKNOWLEDGMENTS

Research supported by NSF award #1800088 and the Federal Aviation Administration (FAA). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF and FAA.

REFERENCES

[1] M. R. Asghar, G. Dán, D. Miorandi, and I. Chlamtac. 2017. Smart Meter Data Privacy: A Survey. *IEEE Communications Surveys Tutorials* 19, 4 (2017), 2820–2835. https://doi.org/10.1109/COMST.2017.2720195

[2] H. Bao and R. Lu. 2015. A New Differentially Private Data Aggregation With Fault Tolerance for Smart Grid Communications. *IEEE Internet of Things Journal* 2, 3 (2015), 248–258. https://doi.org/10.1109/JIOT.2015.2412552

[3] Juan Benet. 2014. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561* (2014).

[4] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. 2003. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Advances in Cryptology — EUROCRYPT 2003*, Eli Biham (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 416–432.

[5] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. 2005. Evaluating 2-DNF formulas on ciphertexts. In *Theory of cryptography conference*. Springer, 325–341.

[6] C. Callegari, S. De Pietro, S. Giordano, M. Pagano, and G. Procissi. 2013. 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*. 1622–1627.

[7] Ran Canetti. 2001. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. IEEE Computer Society, 136–145.

[8] Le Chen, Rongxing Lu, and Zhenfu Cao. 2014. PDAFT: A privacy-preserving data aggregation scheme with fault tolerance for smart grid communications. *Peer-to-Peer Networking and Applications* 8 (03 2014), 1122–1132. https://doi.org/10.1007/s12083-014-0255-5

[9] configuration [n.d.]. Pairing library. https://github.com/dfaranha/OpenPairing.

[10] Feng Diao, Fangguo Zhang, and Xiangguo Cheng. 2015. A Privacy-Preserving Smart Metering Scheme Using Linkable Anonymous Credential. *IEEE Transactions on Smart Grid* 6 (2015), 461–467.

[11] Tassos Dimitriou and Ghassan O Karame. 2015. Enabling anonymous authorization and rewarding in the smart grid. *IEEE Transactions on Dependable and Secure Computing* 14, 5 (2015), 565–572.

[12] Features of smart meter 2019. Features of smart meter. https://www.sciencedirect.com/topics/engineering/smart-metering.

[13] Prosanta Gope and Biplab Sikdar. 2018. An efficient data aggregation scheme for privacy-friendly dynamic pricing-based billing and demand-response management in smart grids. *IEEE Internet of Things Journal* 5, 4 (2018), 3126–3135.

[14] Zhitao Guan, Guanlin Si, Xiaosong Zhang, Longfei Wu, Nadra Guizani, Xiaojiang Du, and Yinglong Ma. 2018. Privacy-preserving and efficient aggregation based on blockchain for power grid communications in smart communities. *IEEE Communications Magazine* 56, 7 (2018), 82–88.

[15] Kyawt May Hlaing and Dim En Nyaung. 2019. Electricity Billing System using Ethereum and Firebase. In *2019 International Conference on Advanced Information Technologies (ICAIT)*. IEEE, 217–221.

[16] Jaap-Henk Hoepman. 2017. Privacy Friendly Aggregation of Smart Meter Readings, Even When Meters Crash. In *Proceedings of the 2nd Workshop on Cyber-Physical Security and Resilience in Smart Grids* (Pittsburgh, PA, USA) *(CPSR-SG'17)*. Association for Computing Machinery, New York, NY, USA, 3–7. https://doi.org/10.1145/3055386.3055389

[17] Hongbo Hu, Xin Zhao, Yalian Wu, Mengbiao Huang, Ziqi Zhu, and Qingyu Yang. 2020. Privacy Preservation of Smart Meters Based on Identity Authentication. *Energy and Power Engineering* 12 (01 2020), 53–62. https://doi.org/10.4236/epe.2020.124B006

[18] Marek Jawurek, Martin Johns, and Florian Kerschbaum. 2011. Plug-in privacy for smart metering billing. In *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 192–210.

[19] Tobias Jeske. 2011. Privacy-preserving smart metering without a trusted-third-party. In *Proceedings of the International Conference on Security and Cryptography*. IEEE, 114–123.

[20] F. Knirsch, G. Eibl, and D. Engel. 2018. Error-Resilient Masking Approaches for Privacy Preserving Data Aggregation. *IEEE Transactions on Smart Grid* 9, 4 (2018), 3351–3361. https://doi.org/10.1109/TSG.2016.2630803

[21] Yining Liu, Wei Guo, Chun-I Fan, Liang Chang, and Chi Cheng. 2018. A practical privacy-preserving data aggregation (3PDA) scheme for smart grid. *IEEE Transactions on Industrial Informatics* 15, 3 (2018), 1767–1774.

[22] Daisuke Mashima and Arnab Roy. 2014. Privacy preserving disclosure of authenticated energy usage data. *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)* (2014), 866–871.

[23] Ramyar Rashed Mohassel, Alan S Fung, Farah Mohammadi, and Kaamran Raahemifar. 2014. A survey on advanced metering infrastructure and its application in smart grids. In *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 1–8.

[24] Andrés Molina-Markham, Prashant Shenoy, Kevin Fu, Emmanuel Cecchet, and David Irwin. 2010. Private Memoirs of a Smart Meter. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building* (Zurich, Switzerland) *(BuildSys '10)*. Association for Computing Machinery, New York, NY, USA, 61–66. https://doi.org/10.1145/1878431.1878446

[25] Mustafa A Mustafa, Sara Cleemput, Abdelrahaman Aly, and Aysajan Abidin. 2019. A secure and privacy-preserving protocol for smart metering operational data collection. *IEEE Transactions on Smart Grid* 10, 6 (2019), 6481–6490.

[26] Petlib Python library 2019. Petlib Python library. https://github.com/gdanezis/petlib.

[27] Pinata IPFS [n.d.]. Pinata IPFS. https://pinata.cloud/

[28] Python library 2019. Python library. https://pypi.org/project/cryptography/.

[29] Alfredo Rial and George Danezis. 2011. Privacy-preserving smart metering. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*. 49–60.

[30] Cristina Rottondi, Giacomo Verticale, and Antonio Capone. 2013. Privacy-preserving smart metering with multiple data Consumers. *Computer Networks* 57 (05 2013), 1699–1713. https://doi.org/10.1016/j.comnet.2013.02.018

[31] Neetesh Saxena, Bong Jun Choi, and Santiago Grijalva. 2017. Secure and privacy-preserving concentration of metering data in AMI networks. In *2017 IEEE International Conference on Communications (ICC)*. IEEE, 1–7.

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

[33] Smart Meter configuration [n.d.]. Smart Meter configuration. https://www.renesas.com/us/en/document/bro/smart-meter-solutions-catalog?language=en

[34] Markus Stadler. 1996. Publicly Verifiable Secret Sharing. In *Advances in Cryptology — EUROCRYPT '96 (Lecture Notes in Computer Science, Vol. 1070)*, Ueli Maurer (Ed.). Springer-Verlag, 190–199.

[35] Samet Tonyali, Kemal Akkaya, Nico Saputro, A Selcuk Uluagac, and Mehrdad Nojoumian. 2018. Privacy-preserving protocols for secure and reliable data aggregation in IoT-enabled smart metering systems. *Future Generation Computer Systems* 78 (2018), 547–557.

[36] Gaurav S Wagh, Sahil Gupta, and Sumita Mishra. 2020. A distributed privacy preserving framework for the Smart Grid. In *2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 1–5.

[37] Gaurav S Wagh and Sumita Mishra. 2020. A cyber-resilient privacy framework for the smart grid with dynamic billing capabilities. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 1–6.

[38] Gaurav S Wagh and Sumita Mishra. 2021. Divide & Conquer: A Privacy Safeguarding Framework for the Smart Grid. In *ICC 2021-IEEE International Conference on Communications*. IEEE, 1–6.