Fully Privacy-preserving Billing Models for Peer-to-Peer Electricity Trading Markets

Akash Madhusudan[a,b], Mustafa A. Mustafa[d,a,*], Hilder V.L. Pereira[c], Erik Takke[b]

[a] *COSIC, KU Leuven, Belgium*
[b] *3MI Labs, Belgium*
[c] *Universidade Estadual de Campinas (Unicamp), Brazil*
[d] *The University of Manchester, UK*

Abstract

Peer-to-peer energy trading markets enable users to exchange electricity, directly offering them increased financial benefits. However, discrepancies often arise between the electricity volumes committed to in trading auctions and the volumes actually consumed or injected. Solutions designed to address this issue often require access to sensitive information that should be kept private.

This paper presents a novel, fully privacy-preserving billing protocol designed to protect users' sensitive consumption and production data in the context of billing protocols for energy trading. Leveraging advanced cryptographic techniques, including fully homomorphic encryption (FHE) and pseudorandom zero sharing (PRZS), our protocol ensures robust security and confidentiality while addressing the critical issue of managing discrepancies between promised and actual electricity volumes. The proposed protocol guarantees that users' sensitive information remains inaccessible to external parties, including the trading platform and billing server. By utilizing FHE, the protocol allows computations on encrypted data without compromising privacy, while PRZS ensures secure aggregation of individual discrepancies of each household. This combination of cryptographic primitives maintains data privacy and enhances billing accuracy, even when fluctuations in energy supply and demand occur.

*Corresponding author
Email address: `mustafa.mustafa@manchester.ac.uk` (Mustafa A. Mustafa)

We analyze real-time consumption and production data from 100 households to experimentally validate the effectiveness and efficiency of our billing model. By implementing a flexible framework compatible with any billing method, we demonstrate that our protocol can accurately compute individual bills for 100 households in approximately 0.17 seconds.

Keywords: Privacy, Security, Homomorphic Encryption, P2P electricity market, Billing model, Imbalance cost

1. Introduction

Peer-to-peer (P2P) electricity trading markets allow households to trade with each other through auction mechanisms, giving them more flexibility as opposed to trading only with their contracted supplier, often on fixed prices [1]. This flexibility allows households to be more selective about (1) when they trade their surplus electricity and (2) the selected trading price when they devise their auction bids. The rational strategy while making these selections is to reduce their costs (when buying) or increase their revenues (when selling) electricity [2]. Such markets also encourage local energy production, consumption, and balancing, bringing environmental benefits to local communities [3].

However, in practice, there are often discrepancies between trading volumes committed at an auction (bid volumes) and the volumes consumed/injected by the participating households (actual smart meter readings). This happens because of the following factors: (1) volatility of the consumption/production patterns of the households [4], (2) uncertainty in the outputs of local renewable sources [5], and (3) inaccuracy of the prediction algorithms used by households to devise their auction bids [6]. To maximize the volumes of energy trading at the P2P markets, new billing models that minimize these discrepancies by sharing them among participating households have been proposed [7].

These billing models (and P2P markets in general), however, require more fine-grain data (such as bid volume, meter readings, and price per trading slot) to be shared with other households or the market/auction operator [1]. Such fine-grain data pose significant privacy risks [8]. Potentially, a malicious actor can deploy non-intrusive load monitoring techniques [9] and can easily infer sensitive information about the household occupants such as presence, activities, religion, health conditions, etc. [10].

There have already been attempts to protect households' data by designing privacy-preserving billing protocols for P2P energy trading markets. However, they have limitations. Solutions such as [11] do not consider the aforementioned discrepancies and assume perfect fulfillment of promised bids. The ones that provide privacy-preserving billings while taking the discrepancies into account either use partially homomorphic encryption, thus having limited expressibility in their privacy-preserving computations [12, 13, 14], or deploy multiparty computation techniques which come with high communication overhead [15, 16]]. In general, the control over the user's data is given to the supplier due to the nature of these protocols. Furthermore, none of these solutions evaluate their protocols with real-time consumption/production data.

To address these limitations, this paper proposes a novel privacy-preserving billing protocol that utilizes FHE to calculate encrypted bills that can only be decrypted by the respective household. More specifically, the novel contributions of this paper are:

- We design a fully privacy-preserving billing protocol for P2P energy markets which accommodates discrepancies between electricity volumes committed and actually consumed/injected by households. We utilize FHE to encrypt every household's data with their own key. This enables us to ensure that the control of a household's data is never given to any third party.
- We use real-time consumption data of 100 households, of which 25 are prosumers and 75 are consumers. The prosumers are a mixture of households with wind turbines, solar panels, or both. We simulate the production of these renewable sources based on real-time weather conditions.
- We provide a framework implementation that enables fully private billings

using FHE. Our implementation can be adapted to fit any billing model and is experimentally evaluated as efficient. We demonstrate that our implementation can perform bill calculations for 100 households in approximately 3 seconds.

The rest of the paper is organized as follows: in Sect. 2 we discuss research that has already been done in the field of billings. Then, in Sect. 3 we discuss our system model, threat model, design requirements, the non-privacy-preserving billing model we adapt, and the cryptographic primitives we use to make these models privacy-preserving. Section 4 details our fully privacy-preserving billing model with universal cost split (UCS), followed by its evaluation in Sect. 5. Finally, Sect. 6 concludes this paper and lists some future work directions.

2. Related Work

Privacy risks in P2P electricity trading markets (and in smart metering [17] in general) have previously been highlighted [8]. There have already been attempts to reduce these risks. As follows, we review the privacy-preserving billing solutions for smart metering before focusing on the ones for P2P electricity trading markets.

Rial and Danezis [18] proposed a privacy-preserving protocol for time-of-use billing using fine-grained meter readings, which never leave the user households. Their protocol allows users to compute their bill locally and prove its correctness without disclosing any fine-grained consumption with the help of zero-knowledge proofs. Realizing that even the final bill may leak users' private information, Danezis et al. [19] have proposed to add some small amount of noise to the final bill so that they can offer strong privacy guarantees. This is achieved by using differential privacy. One disadvantage of this method is that 'noise' corresponds to real money, which users must pay to preserve their privacy. To address this limitation, the authors proposed a complementary billing protocol that relies on a cryptographic oblivious mechanism to support rebates. With the help of this additional protocol, a user can deposit an initial monetary value that can support positive and negative noise added to their bills while ensuring that the funds they pay always cover the cost of their consumption.

Erkin and Tsudik [20] proposed privacy-preserving spatial and temporal aggregation of smart metering data. Their solution for temporary aggregation is suitable for private billing. Each smart meter generates random numbers equal to the number of time slots in a billing period. These random numbers are used to encrypt each fine-grained metering data with the smart meter manufacturer's homomorphic public key. The aggregation is either computed on the supplier side or on the smart meter itself. Although none of the individual smart meter readings are leaked to the supplier or the manufacturer, the manufacturer also has access to users' monthly bills. Jawurek et al. [21] have taken a similar approach. They proposed a privacy-preserving billing protocol that relies on zero-knowledge proofs based on Pedersen commitments performed by a plug-in privacy component placed into the communication link between the smart meter and supplier.

Danezis et al. [22] proposed complex non-linear billing protocols for smart meter readings that deploy multi-party computation (based on secret sharing techniques). The limitation of these protocols is that they rely on several semi-honest authorities to maintain the privacy of user's inputs. Similarly, Ababneh et al. [23] proposed a scheme for privacy-preserving electricity billing that deploys secret sharing and blinding techniques. In their proposed scheme, each smart meter blinds each of its meter readings and secretly shares them with its peers, along with aggregating all the blinded readings for all time slots per billing period. This aggregated value is then shared with a lead smart meter. The lead smart meter then verifies that both the blinded aggregate values are the same and shares this value with the supplier. The supplier then unblinds the aggregate value and uses it to calculate the final user bill. The limitation of this scheme is its high communication cost, as it requires many-to-many communication links among the smart meters themselves.

Inspired by the existing solutions for privacy-preserving billing for smart metering, there have already been attempts to provide billing solutions for P2P electricity markets that protect users' privacy.

Abidin et al. [24] proposed a simple billing protocol for the P2P electricity trading market. Each smart meter calculates a fine-grained bill per time slot and then masks it before sending the masked bill to a supplier. When summed up in a billing period, the masked values are selected to cancel each other. Once the supplier gets all the masked values from a smart meter for the billing period, it aggregates them, resulting in the user's monthly bill (as the random numbers cancel each other). Their work, however, does not consider individual deviations of households in the bill calculation and assumes perfect fulfillment. In addition, since the bills are computed at the smart meters, changes to the billing model would require changes to all smart meters, which, in some cases, might not be possible/practical.

Thandi and Mustafa [12] proposed the first privacy-preserving billing solution for P2P trading markets that considers discrepancies between what volumes households commit and what volumes they finally consume/deliver during a trading period. Their idea enables these discrepancies to be traded in the retail market. They deploy a partial homomorphic encryption scheme to calculate what volumes are traded at the retail and the P2P market, respectively. Hutu and Mustafa [13] improved the solutions proposed in [12] by proposing privacy-preserving billing protocols that can accommodate several (more complex) billing models for reducing users' bills by (more) fairly splitting the cost amongst all users through the deployment of weighting in the calculations of each user's individual contribution towards the overall discrepancy. All the calculations are computed using a partial homomorphic encryption scheme.

Erdayandi et al. [14] deployed a blockchain to store hashes of the committed volumes by the smart meters and the real meter readings. These readings are later used to verify the user bill. The final bill of consumers is calculated by prosumers, and vice versa, using a partial homomorphic encryption scheme. For example, for each billing period, a consumer is matched with several prosumers, who receive encrypted data from the consumer and calculate the consumer's encrypted bill. The bill is then sent to the supplier, who can decrypt it and obtain the user's bill. Alqahtani and Mustafa [15] further improved the billing models by incorporating different zones, encouraging in-zone trades, and reducing user bills. In [16], the same authors further improve [15] by involving a collaboration of different entities performing bill computation with verification of correctness. Their solution also mitigates the potential impact on individuals' privacy resulting from internal collusion. They propose three different approaches, resulting in different levels of privacy protection and performance. Both of these solutions deploy multi-party computation techniques to ensure that the bills are calculated in a privacy-preserving manner.

In summary, existing solutions either assume perfect fulfillment of promised bids [11] or use partially homomorphic encryption schemes [12, 13, 14]. Thus, they have limited expressibility in their privacy-preserving computations, and do not enable full control of the data by the users. Some existing solutions [15, 16] also use multi-party computation techniques, thus introducing too much communication overhead.

To address these limitations, we propose a novel privacy-preserving billing protocol for P2P electricity markets that utilizes fully homomorphic encryption to calculate encrypted bills, which can only be decrypted by a participating user.

3. Preliminaries

This section describes our solution's system model, specifically what entities participate in making it work. We then proceed to our threat model and the assumptions we make. Finally, we conclude by detailing our solution's design requirements.

3.1. System Model

The system model used in this work consists of the following entities:

- Users: entities that use electricity in their households. Users who can only consume electricity are known as *consumers*, and ones who consume and produce electricity (e.g., by using renewable energy sources) are called *prosumers.*

- Suppliers: entities that provide electricity to users in retail markets. They also purchase electricity from users at a feed-in tariff. Additionally, suppliers act as a fallback option in case of discrepancies in the P2P market.
- Grid operator: entities that manage the injection and output of electricity to/from the grid are known as grid operators. They ensure that the grid remains balanced to avoid outages. It is assumed that the grid operators have an additional role of aggregating the household input data required for our billing models. Thus, we use the terms grid operator and aggregator interchangeably.
- Market operator: entities responsible for executing the auction mechanism that matches prosumers and consumers for P2P trading.
- Smart meter: used to measure the volume of electricity consumed/injected by the users from/into the grid. They facilitate P2P trading by performing functions and providing information necessary for our proposed billing models to work.

3.2. Threat Model and Assumptions

We assume the existence of honest-but-curious adversaries: the grid operator, users, and suppliers act according to the protocol but are curious about the information they receive. The threat model for the market operators is borrowed from the work done by Botelho Da Gama et al. [25] and assumes an active adversary running the P2P auction mechanism. We assume that the SMs are tamper-proof and that the prosumers and consumers act rationally. All entities in our billing models are assumed to communicate over secure and authenticated communication channels. Finally, we assume that a household can only sell or buy electricity from P2P market at a given time; it cannot do both.

3.3. Design Requirements

We split the design requirements for our billing models into functional, security, and privacy requirements.

3.3.1. Functional Requirements

The functional requirements of our billing models are as follows:

- **Accomodation of deviations**: The billing models should handle positive or negative deviations from all households and should be able to function in the presence of any arbitrary number of deviations for each time period.
- **Fairness**: The incurred penalties due to deviations should be split amongst all the deviating entities.

3.3.2. Security and Privacy Requirements

The security and privacy requirements of our billing models are as follows:

- **Authenticity of consumption/production data**: The aggregator should validate the integrity and origin of all production-/consumption-specific data.
- **Non-repudiation of calculated bills**: The bills calculated by the aggregator should be authenticated as such.
- **Unlinkability of prosumers and consumers**: Any third party should not be able to link a pair of prosumers and consumers who trade.
- **Confidentiality of billing data**: Any data that can classify consumption/production patterns of a household should not be accessible by any third party in plain. Only the encrypted version of this data should be available.

Algorithm 1 Billing Model for Retail Markets for each user per time slot [7]

```
Input: Individual consumer demand per time slot ($C^{dem}_{i,t}$), Individual prosumer supply per time slot ($P^{sup}_{i,t}$), RP, FiT
Output: Consumer bill per time slot ($C_{bill}$), Prosumer reward per time slot ($P_{rew}$).
1: $C_{bill} = C^{dem}_{i,t} \times \text{RP}$
2: $P_{rew} = P^{sup}_{i,t} \times \text{FiT}$
```

3.4. Billing Models

We briefly revisit the billing model with the UCS proposed by Madhusudan et al. in [7]. We start with the billing model for retail markets only (the status quo). We then describe a P2P market with RM (the status quo) acting as a fall-back option. Finally, we present a billing model with UCS as listed in [7].

3.4.1. Billing Model for Retail Markets – the status quo

Algorithm 1 presents a billing model that is used in current retail markets. In this billing model, customers buy/sell their electricity from/to suppliers they have a contract with. The model inputs a customer's consumption and production, in addition to the retail and feed-in tariff set by the entities in retail markets. Note that Algorithm 1 describes a billing model for an individual user per time slot. A consumer (C) is charged a retail price (RP) per kWh for the electricity they consume in each timeslot; similarly, a prosumer (P) gets paid a feed-in tariff (FiT) per kWh of electricity they inject to the grid for each timeslot. The supplier's balance (S) is derived from the difference between their earnings through billing consumers and expenditure through paying prosumers.

3.4.2. P2P market with RM as Back-up

This P2P market uses the retail market as a backup in case there is a deviation in the volume of electricity that was committed to be consumed/supplied and the actual volume delivered. Each customer has an individual deviation ($InDev_x$) where x=p for a prosumer and x=c for a consumer. Figure 1 depicts how such an interaction would look like. Briefly, P2P trading markets run as follows:

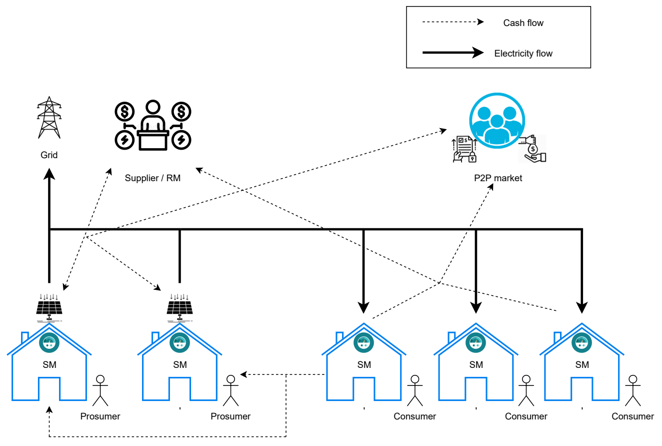

Figure 1: P2P market with RM as a fall-back option.

- Consumers/Prosumers submit bids/offers for electricity;
- A market operator (e.g., trading platform) clears the P2P market. It determines the total cleared volume of electricity at the market, the market clearance price and the consumers/prosumers whose bids/offers have been accepted. Consumers/Prosumers whose bids have not been accepted use the RM to meet their needs.
- Finally, meter readings from the smart meters of consumers/prosumers are used to calculate their bills/rewards. Suppose the data from the smart meters of prosumers/consumers whose bids have been accepted is not the same as their respective committed value in the bid/offer. In that case, the RM compensates for the difference (deviation). This could come with an extra cost for some of the prosumers/consumers.

In all future billing models, we do not explicitly list the supplier balance as that is not the main focus of this work and it is straightforward to calculate it. The balance of a supplier (S_{bal}) is always the difference between their income S_{inc} = demand x RP and their expenditure S_{exp} = supply x FiT.

The cost associated with compensating the deviations of the individual prosumers/consumers will have to be distributed amongst all (or some) of the

Table 1: Abbreviations and Notations

Symbol	Description
TD_i	Total deviation per time slot
S	Number of timeslots
InDev_i, t	Individual deviation of a customer per timeslot t
TP, RP	Trading price, Retail price
FiT	Feed-in tariff
$P2P_n^c$	Total no. of P2P consumers with accepted bids and individual deviations opposite to the total deviation
$C_{i,t}^{dem}$	Individual demand of consumer per time slot
$P2P_n^p$	Total no. of P2P prosumers with accepted bids and individual deviations opposite the total deviation
$P_{i,t}^{sup}$	Individual supply of prosumer per time slot
C_{bill}, P_{rew}	Consumer bill, Prosumer reward
$\mathbf{c}, \mathbf{s}, \mathbf{d}, \mathbf{md}, \mathbf{b}$	Vector of consumptions, supplies, individual deviations, masked individual deviations, signs of deviations
$\mathbf{u}, \mathbf{v}, \mathbf{y}, \mathbf{w}$	Vector of trading prices, retail prices, feed-in tariffs, total deviations, per time slot

prosumers/consumers to reduce the dependency on the retail market as a fallback. There could be different mechanisms to share these costs amongst prosumers/consumers. Madhusudan et al. [7] propose three different methods of sharing these costs: individual, social, and universal cost split. In this paper, we focus on UCS.

Note that Table 1 lists the notations used throughout the paper.

3.4.3. Billing model with UCS

This billing model aggregates all the individual deviations of P2P market users (prosumers and consumers), and the cost of this resulting total deviation is split amongst the customers whose individual deviation is in the same direction as the total deviation. This is described below and shown in Algorithm 2.

Interested readers are directed to [7] for a detailed explanation of UCS.

Algorithm 2 Billing Model with Universal Cost Split UCS (adapted from [7])

Input: Individual consumer/prosumer demand/supply per time slot ($C_{i,t}^{dem}$ and $P_{i,t}^{sup}$), Individual costumer deviation per time slot (InDev$_{i,t}$), total deviation per time slot (TD_t), TP, RP, FiT, number of consumers/prosumers whose individual deviation is opposite to/aligned with the total deviation ($P2P_n^c$ / $P2P_n^p$)

Output: Customer bill (C_{bill}) and Prosumer reward (P_{rew}) per user per time slot

```
1:  for each time slot $0 \leq t < S$ do
2:      if bid Accepted then
3:          if TD = 0 then
4:              $C_{bill} = C_{i,t}^{dem} \times TP$
5:              $P_{rew} = P_{i,t}^{sup} \times TP$
6:          if TD < 0 then
7:              $P_{rew} = P_{i,t}^{sup} \times TP$
8:              if InDev$_{i,t} = 0$ or InDev$_{i,t} < 0$ then
9:                  $C_{bill} = C_{i,t}^{dem} \times TP$
10:             if InDev$_{i,t} > 0$ then
11:                 $C_{bill} = (C_{i,t}^{dem} - |\frac{\mathsf{TD}_t}{P2P_n^c}|) \times TP + |\frac{\mathsf{TD}_t}{P2P_n^c}| \times RP$
12:         if TD > 0 then
13:             $C_{bill} = C_{i,t}^{dem} \times TP$
14:             if InDev$_{i,t} = 0$ or InDev$_{i,t} < 0$ then
15:                 $P_{rew} = P_{i,t}^{sup} \times TP$
16:             if InDev$_{i,t} > 0$ then
17:                 $P_{rew} = (P_{i,t}^{sup} - |\frac{\mathsf{TD}_t}{P2P_n^p}|) \times TP + |\frac{\mathsf{TD}_t}{P2P_n^p}| \times FiT$
18:     if bid not accepted then
19:         goto Algorithm 1
```

3.5. Cryptographic Primitives

3.5.1. Fully Homomorphic Encryption

Fully homomorphic encryption (FHE) is a powerful tool that allows operations to be performed on encrypted data. One can encrypt the data and then send it to a third party, which can still process it without decrypting it. Once the third party finalizes its operations, it returns the encrypted data back to the sender, who can then perform its decryption to reveal the resulting output.

In more detail, a client generates a ciphertext c encrypting some data m under a public key pk (to which it knows the corresponding private key sk). Then, a server holding c and pk, but not the corresponding secret key sk, can then construct a new ciphertext $c^{'}$ that encrypts $f(m)$ for any function f. We call this step the "homomorphic computation" and say that "f was evaluated homomorphically". Finally, the client can download c', decrypt it using sk, and obtain $f(m)$. Since sk remains with the client, FHE allows one to securely outsource computation to an untrusted party.

Now, we present at a high level the operations available in a generic FHE scheme allowing plaintext slots.

- $\mathsf{HE.paramGen}(\lambda, p)$: given the security parameter λ and the plaintext precision p, generates parameters param, which include the *ring dimension* N, the *ciphertext modulus* Q, and the number of slots S.

- $\mathsf{HE.keyGen}(\mathsf{param})$: generates the encryption key pk, the decryption key sk, and also the following special keys, which are also public: relinearization key rlk; rotation keys rk_i for $0 \leq i < N$; and bootstrapping key bk.

- $\mathsf{HE.enc}(\mathbf{m})$: outputs a ciphertext $\mathbf{c}$ encrypting an S-dimensional vector $\mathbf{m}$ In this case, we write $\mathbf{c} = \mathsf{Enc}(m_1, ..., m_S)$.

- $\mathsf{HE.dec}(\mathbf{c})$: decrypts $\mathbf{c}$ and returns the message $\mathbf{m}$.

- $\mathsf{HE.add}(\mathbf{c}_0, \mathbf{c}_1)$: consider that $\mathbf{c}_0$ and $\mathbf{c}_1$ encrypt vectors $\mathbf{u}$ and $\mathbf{v}$. This homomorphic operation outputs $\mathbf{c} = \mathsf{Enc}(\mathbf{w})$ where $\mathbf{w}[i] = \mathbf{u}[i] + \mathbf{v}[i]$.

- $\mathsf{HE.mult}(\mathbf{c}_0, \mathbf{c}_1, \mathsf{rlk})$: consider that $\mathbf{c}_0$ and $\mathbf{c}_1$ encrypt vectors $\mathbf{u}$ and $\mathbf{v}$. This homomorphic operation outputs $\mathbf{c} = \mathsf{Enc}(\mathbf{w})$ where $\mathbf{w}[i] = \mathbf{u}[i]{\cdot}\mathbf{v}[i]$. Notice that it requires the relinearization key.

3.5.2. Pseudo-Random Zero Sharing

We recall a simplified version of the Pseudo-random zero sharing (PRZS) protocol from Appendix A of [26]. Here, we do not need commitment schemes in the setup phase since we assume the semi-honest model, i.e., all parties follow the protocol during its execution, but can retain all the exchanged messages to be analyzed later aiming to discover secret values. This protocol allows k parties to generate random-looking shares that add up to zero. Those shares can then be used to mask values that the users want to send to other users. This is done by adding the derived shares to the user's input, akin to a one-time pad. The PRZS is defined as follows:

- $\mathsf{Setup}(\lambda, k)$: each user $1 \leq u \leq k$ generates PRF keys $K_{u,v}$, for $1 \leq v \leq k$ and $u \neq v$ and sends $K_{u,v}$ to user v. The keys are long enough so that the PRF achieves λ bits of security.
- $\mathsf{GenShare}(u, \mathsf{count})$: define $a_i := \mathsf{PRF}_{K_{u,i}}(\mathsf{count})$ and $b_i := \mathsf{PRF}_{K_{i,u}}(\mathsf{count})$ for $i \neq u$. Then, the user u's share is defined as $s_u = \sum_{i=1}^{k} a_i - \sum_{j=1}^{k} b_j$.

Note that $\sum_{u=1}^{k} s_u = 0$. Also, each s_u is pseudorandom, i.e., indistinguishable from a random value.

4. Privacy-Enhanced Billing Models

We introduce a privacy-preserving variant of the universal cost-split billing model that collectively enables customers to settle their electricity bills fully privacy-preserving while reducing the dependency on retail markets.

4.1. Overview

Before going into the details, we review how this model works. Figure 2 provides a schematic representation of this model.

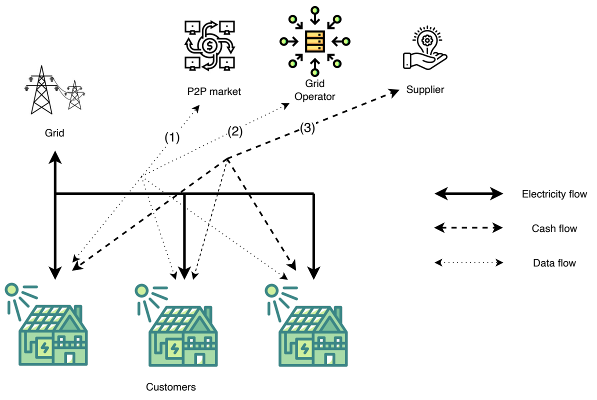

Figure 2: Overview of our privacy-preserving billing model.

For each time slot, each user participates in the auction held on the trading platform [as shown in step (1)]. The customers selected for P2P trading send their consumption, production, and deviation data encrypted to the grid operator [as shown in step (2)]. The grid operator then runs the privacy-preserving billing model, i.e., for each customer, the grid operator calculates the encrypted bill/reward using only this encrypted data. Consequently, they output the encrypted bill/reward. Upon receiving the encrypted bill/reward, the customer decrypts it with their secret key to reveal how much they owe/are owed to/by the supplier. The customer then performs or receives the payment [as shown in step (3)].

4.2. Setup Phase

Before utilizing the billing models, we need to perform the steps necessary to generate the required cryptographic keys for the utilized cryptographic primitives, i.e., PRZS [26] and FHE [27].

As discussed in Sec. 3.5, each of these primitives has a setup phase that has to be executed *once* for a set of users utilizing our billing models to receive their keys. After generating these keys, users can use them in P2P electricity trading.

Specifically, for PRZS, the participant needs to execute the $\mathsf{Setup}(\lambda, k)$ where λ is the desired security parameter and k is the number of participants. The output of this setup is the PRF keys required to generate masking shares for the individual deviations.

For FHE, the users must execute $\mathsf{HE.paramGen}(\lambda, p)$ and $\mathsf{HE.keyGen}(\mathsf{param})$ to generate the necessary key pair for encrypting and decrypting their deviations, consumptions, and productions.

By the end of this setup phase, all participants must have the required cryptographic keys.

4.3. Client Setup for Billing Model with Universal Cost Split

Our billing model is divided into client setup and server-side execution. The client setup details each household's steps to send encrypted consumption, production, and deviation data to the grid operator's server.

Upon receiving the required data, the grid operator executes the billing algorithm to derive the bills and rewards (in encrypted form) for each client, i.e., the users.

These steps are depicted in Alg. 3. During the client setup, the SMs on the client's side collect the energy consumption and/or production for each time slot. The SM computes deviations compared to their committed values in the P2P market using this data. Naively, the SMs can encrypt each of these measurements in a different ciphertext and send it to the server (grid operator) to be processed. However, that would sub-utilize the FHE ciphertexts since they can encrypt a vector. Hence, for efficiency, the SMs first collect several measurements, populate a vector, encrypt it, and then send the ciphertext to the grid operator.

Moreover, the SM encrypts each deviation's sign to simplify the homomorphic computation performed on the server side. More specifically, the grid operator has to compute values that depend on whether the deviation is negative or positive. Thus, given a ciphertext encrypting the deviation, the grid operator would have to compute a homomorphic comparison, which is rather expensive. However, if the grid operator already has a ciphertext encrypting the sign, the homomorphic comparison is replaced by two homomorphic multiplications, which is more computationally efficient.

Algorithm 3 Client Data Collection

Input: Vector of consumptions ($\mathbf{c}$), Vector of supplies ($\mathbf{s}$), Vector of individual deviations ($\mathbf{d}$), Vector of masked deviations ($\mathbf{md}$), Vector of deviation signs ($\mathbf{b}$), Number of slots (S), Client's secret key (sk), Global Count ($count$), Individual consumption/supply of each consumer/prosumer per time slot ($\mathsf{C}_{i,t}^{dem}$, $\mathsf{P}_{i,t}^{sup}$), Individual deviation per customer per time slot ($\mathsf{InDev}_{i,t}$).

Output: Ciphertexts $\mathbf{c}_c$, $\mathbf{c}_d$, $\mathbf{c}_s$, and $\mathbf{c}_b$ encrypting, respectively, vectors of consumptions, individual devitations, supplies, and bits.

```
1:  $\mathbf{c} = (0, ..., 0) \in \mathbb{Z}^S$    $\rightarrow$ Vector of consumptions
2:  $\mathbf{s} = (0, ..., 0) \in \mathbb{Z}^S$    $\rightarrow$ Vector of supplies
3:  $\mathbf{d} = (0, ..., 0) \in \mathbb{Z}^S$    $\rightarrow$ Individual deviations
4:  $\mathbf{md} = (0, ..., 0) \in \mathbb{Z}^S$    $\rightarrow$ Masked individual deviations
5:  $\mathbf{b} = (0, ..., 0) \in \mathbb{Z}^S$    $\rightarrow$ Signs of deviations
6:  for each time slot $0 \leq t < S$ do
7:      $\mathbf{c}[t] = \mathsf{C}_{i,t}^{dem}$
8:      $\mathbf{s}[t] = \mathsf{P}_{i,t}^{sup}$
9:      $\mathbf{d}[t] = \mathsf{InDev}_{i,t}$
10:     if $\mathsf{InDev}_{i,t} \leq 0$ then
11:         $\mathbf{b}[t] = 1$
12:     $s_t$ = GenShare(id, $count||t$) $\in \mathbb{Z}_{2^\ell}$
13:     $\mathbf{md}[t] = s_t + \mathbf{d}[t] \bmod 2^\ell$
14: $\mathbf{c}_c = \mathsf{HE.enc}_{\mathsf{sk}}(\mathbf{c})$
15: $\mathbf{c}_s = \mathsf{HE.enc}_{\mathsf{sk}}(\mathbf{s})$
16: $\mathbf{c}_d = \mathsf{HE.enc}_{\mathsf{sk}}(\mathbf{d})$
17: $\mathbf{c}_b = \mathsf{HE.enc}_{\mathsf{sk}}(\mathbf{b})$
18: Send $\mathbf{c}_c, \mathbf{c}_s, \mathbf{c}_d, \mathbf{c}_b, \mathbf{md}$ to the server
```

The way Alg. 3 proceeds is as follows:

- Lines 6-9: The SM collects the measurements of consumption, production, individual deviations, and signs of deviations for the required number of time slots.

- Lines 10-13: The SM collects consumption and production data and then computes the individual deviations and their signs. Additionally, each household derives its masking value s_t and calculates the masked individual deviation $\mathbf{md}[t]$ per time slot t.

- Lines 14-18: After the billing cycle (i.e., the required number of timeslots), the SM homomorphically encrypts these vectors and sends them to the grid operator.

4.4. Server's Billing Model with Universal Cost Split

Upon receiving the encrypted data from the SMs, the server (grid operator) executes a homomorphic version of Algorithm 2, which we describe in detail in Alg. 4. As each ciphertext encrypts a vector of dimension S, where S denotes the total measurements (billing cycle length) in the vector, the homomorphic version is equivalent to running Alg. 2 in parallel for S different inputs in a privacy-preserving way.

All the branches, i.e., if/else statements of the original algorithm are computed homomorphically based on the fact that the code "`if B then f(x) else g(x)`", where `B` is a Boolean value, is equivalent to "`B*f(x) + (1 - B)*g(x)`". Hence, if `B` equals 1, we end up with $f(x)$. Otherwise, $f(x)$ is multiplied by zero and we have $g(x)$.

For the branches that depend on the individual deviations, note that the SM uploads encryptions of bits that indicate whether $\text{InDev}_x \leq 0$ or not. Thus, we homomorphically compute the bill and the reward corresponding to both cases, i.e., $\text{InDev}_x \leq 0$ or $\text{InDev}_x > 0$.

Finally, we multiply those bills and rewards by the ciphertexts $\mathbf{c}_b$ and $1-\mathbf{c}_b$, with the bits corresponding to signs of the individual deviations, so that only the correct bills and rewards are multiplied by one while the others are nullified.

Algorithm 4 Client Bill Calculation (by server)

Input: Number of slots S. Ciphertexts $\mathbf{c}_c$, $\mathbf{c}_d$, $\mathbf{c}_s$, and $\mathbf{c}_b$ encrypting, respectively, vectors of consumptions, individual devitations, supplies, and bits. Vectors $\mathbf{u}, \mathbf{v}, \mathbf{y}, \mathbf{w}$ consisting of trading prices, retail prices, feed-in tariffs, and total deviations per timeslot, respectively. Vectors $\mathbf{md}_u$, for $1 \le u \le k$, containing the masked individual deviations of each user u.

Output: Ciphertexts $\mathbf{c}_{bill}/\mathbf{p}_{rew}$ encrypting the bills/rewards for each time slot.

```
 1: $\mathbf{u} = (\mathsf{TP}_0, ..., \mathsf{TP}_{S-1}) \in \mathbb{Z}^S$            ▷ Trading price per time slot
 2: $\mathbf{v} = (\mathsf{RP}_0, ..., \mathsf{RP}_{S-1}) \in \mathbb{Z}^S$            ▷ Retail price per time slot
 3: $\mathbf{y} = (\mathsf{FiT}_0, ..., \mathsf{FiT}_{S-1}) \in \mathbb{Z}^S$          ▷ Feed-in tariff per time slot
 4: $\mathbf{w} = \sum_{u=1}^{k} \mathbf{md}_u \in \mathbb{Z}^S$                     ▷ Total deviation per time slot
 5: Let $\mathbf{w}^{(0)} \in \{0,1\}^S$ such that $\mathbf{w}^{(0)}[i] = 1$ iff $\mathbf{w}[i] = 0$
 6: Let $\mathbf{w}^{(<0)} \in \{0,1\}^S$ such that $\mathbf{w}^{(<0)}[i] = 1$ iff $\mathbf{w}[i] < 0$
 7: Let $\mathbf{w}^{(>0)} \in \{0,1\}^S$ such that $\mathbf{w}^{(>0)}[i] = 1$ iff $\mathbf{w}[i] > 0$
 8: $\mathbf{c}'_b = \mathsf{HE.NOT}(\mathbf{c}_b)$
    ▷ Block corresponding to $\mathsf{TD}_i = 0$
 9: $\mathbf{c}_{bill}^{(0)} = \mathbf{c}_c \cdot \mathbf{u}$
10: $\mathbf{p}_{rew}^{(0)} = \mathbf{c}_s \cdot \mathbf{u}$
    ▷ Block corresponding to $\mathsf{TD}_i < 0$
11: $\mathbf{p}_{rew}^{(<0)} = \mathbf{c}_s \cdot \mathbf{u}$
    ▷ If $\mathsf{InDev}_{i,t} \le 0$
12: $\mathbf{c}_{bill}^{(<0)} = \mathbf{c}_b \cdot \mathbf{c}_c \cdot \mathbf{u}$
    ▷ If $\mathsf{InDev}_{i,t} > 0$
13: $\mathbf{c}_{bill}^{(<0)} \mathrel{+}= \mathbf{c}'_b \cdot (\mathbf{c}_c \cdot \mathbf{u} + \frac{\mathbf{w}}{\mathsf{P2P}_n^c} \cdot (\mathbf{v} - \mathbf{u}))$
    ▷ Block corresponding to $\mathsf{TD}_i > 0$
14: $\mathbf{c}_{bill}^{(>0)} = \mathbf{c}_c \cdot \mathbf{u}$
    ▷ If $\mathsf{InDev}_{i,t} \le 0$
15: $\mathbf{p}_{rew}^{(>0)} = \mathbf{c}_b \cdot \mathbf{c}_s \cdot \mathbf{u}$
    ▷ If $\mathsf{InDev}_{i,t} > 0$
16: $\mathbf{p}_{rew}^{(>0)} \mathrel{+}= \mathbf{c}'_b \cdot (\mathbf{c}_s \cdot \mathbf{u} + \frac{\mathbf{w}}{\mathsf{P2P}_n^c} \odot (\mathbf{y} - \mathbf{u}))$
    ▷ Combine bills and rewards for all possible $\mathsf{TD}_i$
17: $\mathbf{c}_{bill} = \mathbf{c}_{bill}^{(0)} \cdot \mathbf{w}^{(0)} + \mathbf{c}_{bill}^{(<0)} \cdot \mathbf{w}^{(<0)} + \mathbf{c}_{bill}^{(>0)} \cdot \mathbf{w}^{(>0)}$
18: $\mathbf{p}_{rew} = \mathbf{c}_{rew}^{(0)} \cdot \mathbf{w}^{(0)} + \mathbf{c}_{rew}^{(<0)} \cdot \mathbf{w}^{(<0)} + \mathbf{c}_{rew}^{(>0)} \cdot \mathbf{w}^{(>0)}$
```

The same strategy is used for the if/else of Alg. 2 that depends on the total deviation. Since this value is clear for the server (because it is obtained via an additive secret sharing protocol), the server can produce, on its own, the bits corresponding to the following three cases: total deviation is less than, equal to, or greater than zero. Then, we homomorphically compute the bills and rewards corresponding to each of these three cases, and multiply by those bits so that we only keep the correct bills and rewards.

For example, if, for the fourth time slot the total deviation ($\mathbf{w}$) is zero, then we have $\mathbf{w}^{(0)}[4] = 1$ while $\mathbf{w}^{(<0)}[4] = \mathbf{w}^{(>0)}[4] = 0$. Then, as shown in lines 17 and 18 of Alg. 4, we compute the bills/rewards using the formulas of the three possible cases, i.e.:
$\mathbf{c}_{bill}^{(0)}$ has in its S slots the bills computed as if all the total deviations were 0, whereas $\mathbf{c}_{bill}^{(<0)}$ encrypts bills considering negative total deviations, and $\mathbf{c}_{bill}^{(>0)}$ encrypts bills considering positive total deviations. Hence, at the end of the algorithm, when we compute

$$\mathbf{c}_{bill} = \mathbf{c}_{bill}^{(0)} \cdot \mathbf{w}^{(0)} + \mathbf{c}_{bill}^{(<0)} \cdot \mathbf{w}^{(<0)} + \mathbf{c}_{bill}^{(>0)} \cdot \mathbf{w}^{(>0)}$$

the fourth slot of the message encrypted by $\mathbf{c}_{bill}$ will encrypt the bill corresponding to the fourth timeslot where $\mathsf{TD}_4 = 0$. The same argument applies to all the other slots and the rewards.

4.5. Bill settlement

After the successful execution of Alg. 4, the client receives either $\mathbf{c}_{bill}$ or $\mathbf{c}_{reward}$. Upon receiving these encrypted bills/rewards, they can decrypt them using the private key generated during the setup phase (see Sec 4.2). Once decrypted, the client can pay the required amount to the supplier or request a reward from the supplier.

5. Evaluation

5.1. Security Analysis

The security and privacy requirements of our billing models (as listed in Sec. 3.2) are (1) **Authenticity of consumption/production data**, (2) **Non-repudiation of calculated bills**, (3) **Unlinkability of prosumers and consumers**, and (4) **Confidentiality of billing data.**

Authenticity of consumption/production data is guaranteed by our assumption that SM's are tamper-proof. They collect data about the volume of electricity being consumed/produced without any alteration, perform the necessary operations and communicate this to the grid operator. Additionally, the provided data is also signed by the SM for non-repudiation and authenticity.

Non-repudiation of calculated bills is guaranteed due to the requirement for grid operators to sign all computed bills/rewards before sending them to the respective households. We assume that the signing algorithm guarantees the properties of *unforgeability* and *verifiability*; thus, we can guarantee the fulfillment of this property.

Unlinkability of prosumers and consumers is guaranteed by the protocol used by the market operator for running the privacy-preserving P2P auction to match trading bids and offers from prosumers and consumers and derive a trading price per timeslot. Protocols such as the one proposed by Botelho Da Gama [25] guarantee unlinkability of prosumers and consumers who trade.

Confidentiality of billing data is guaranteed by the primitives we use for our billing model. The utilized pseudo-random zero sharing [26] is formally proven to provide *pseudo-randomness*, i.e., the masked individual deviations calculated in Alg. 3 are indistinguishable from truly random values. Since we assume passive adversaries, they cannot gain any information about the individual deviations of the households from the masked values. Additionally, our utilized FHE algorithms for calculating the bills is formally proven to be IND-CPA secure [27], i.e., the encrypted volumes and deviations do not leak any information about each household's actual volumes and deviations.

Hence, using these two primitives to calculate the bills/rewards for households does not leak any private information and guarantees the fulfillment of this property.

5.2. Theoretical analysis

Our protocol composes two different primitives, namely, PRZS and FHE. Since PRZS only uses symmetric encryption, its time complexity is negligible compared to the cost of FHE. The only important point to notice about PRZS is that the communication in the setup phase is quadratic in the number of users, i.e., if we use groups of k users, each user has to send about k^2 keys. Given that each key typically has λ bits, the communication cost of this setup step is $\Theta(k^2 \cdot \lambda)$ bits per user.

An important metric in FHE schemes is the depth of the homomorphic computation to be evaluated, which is the number of ciphertext-ciphertexts multiplications that must be performed in sequence. By inspecting Alg. 4, we can see that the depth is only 1. Indeed, one can see that the variable $\mathbf{c}_{bill}^{(<0)}$ is the one with the longest chain of multiplications, but it turns out that it is of the form

$$\mathbf{c}_{bill}^{(<0)} = \mathbf{c}_b \cdot \mathbf{c}_c \cdot \mathbf{u} + \mathbf{c}'_b \cdot \mathbf{c}_c \cdot \mathbf{u} + \mathbf{c}'_b \cdot \mathbf{p}$$

where $\mathbf{u}$ and $\mathbf{p}$ are plaintexts and $\mathbf{c}_b, \mathbf{c}_c$, and $\mathbf{c}'_b$ are ciphertexts.

Since the multiplicative depth is only 1, the client can instantiate the FHE scheme with a very small ciphertext modulus, for example, $Q = 2^{64}$. For generality, we consider then $Q = O(1)$, i.e., a constant. For such small modulus, we can obtain λ bits of security by setting $N = \Theta(\lambda)$ with very small hidden constant. For instance, for $\lambda = 128$, it is possible to choose $N = 1024$. Hence, the size of each ciphertext is $2N \log Q = O(\lambda)$.

By analyzing Alg. 3, we see that the only overhead the client has compared to the original setup algorithm with no privacy guarantee is the sequence of 4 encryptions at the end, which just cost time $O(\lambda \log \lambda)$. Also, the communication cost is just $O(\lambda)$, since the client just has to upload those 4 ciphertexts to the server. However, notice that since each ciphertext encrypts $S = \Theta(\lambda)$ messages, the overhead compared to the plaintext version is just $O(1)$.

To analyze the cost of Alg. 4, i.e., how the server processes the queries, we can focus on the homomorphic multiplications, as they are far more costly than the other operations. We can see that the number of multiplications is constant and, actually, very close to the number of multiplications performed by the original algorithm, which runs in clear. Thus, we can measure the overhead by estimating the cost of one homomorphic multiplication divided by the cost of one multiplication in clear. The latter just costs one instruction. However, one homomorphic multiplication is performed with two or three degree-N polynomial multiplications, which can be performed in time $O(N \log N)$ using Fast Fourier Transform. Again, remember that each homomorphic operation acts on $S = \Theta(N)$ slots in parallel, thus, the amortized cost is $O(\log N)$. Because we can set $N = O(\lambda)$, the overhead of Alg. 4 is then just $O(\log \lambda)$.

Summarizing, to add privacy guarantees to the algorithms, we have almost no overhead in the communication, i.e., $O(1)$, and a very small overhead for the server running time, i.e., $O(\log \lambda)$, where λ is the security parameter which is typically set as 128. In Sect. 5.3, we show concrete numbers obtained with some experiments to measure the overhead in practice.

5.3. Simulations

5.3.1. Use-case

To empirically evaluate our proposed solution, we simulate and analyse the following use-case.

There is a local community of 100 households, of which 75 are consumers and 25 prosumers. The RES owned by the prosumers are as follows: 15 prosumers have photovoltaic (PV), 5 consumers have wind turbines (WTs), and the remaining five prosumers have both.

Each household submits a bid/offer to a P2P trading market for each time slot. The market runs a double auction to select the winning bids/offers and to determine the trading price. The households whose bids are accepted are expected to fulfill their obligation (committed via their bids) by demanding/supplying the corresponding volumes of electricity. After the trading period, real data from the SMs are used to calculate individual bills. Households whose bids are rejected trade on the retail market, while households whose bids are accepted trade on the P2P market (and the retail market if their SM data deviates from their committed volume of electricity in their bids/offers).

The grid operator calculates households' bills. Households can collectively choose to be billed daily, weekly, or monthly.

5.3.2. Data Generation

We use real and simulated data in our experiments. The type of data we use are consumption data for 100 households, PV generation data for 20 households, and WT generation data for 10 households; committed volumes of electricity at the P2P market by the households whose bids/offers are accepted; and trading prices at the P2P market as well as the prices at the retail market.

Using one-hour trading slots, we gather/simulate data for the Brussels region from 1 to 7 July 2020.

Real consumption data collected from 61 Belgian households as part of the Linear project [28] are used as example data and a base for generating the consumption data for 39 households. For this data generation, we follow the following approach for each household. We randomly select one of the 61 households and multiply their consumption data with a randomly selected coefficient from the range [0.9 – 1.1]. In summary, for the consumption data of households, we use real data from 61 real households and synthetic data for 39 households derived from real data.

For the PV generation data, we randomly choose a PV capacity from the set [2.3, 3.6, 4.7] KWp for each household with PVs. Related to the strength of the solar radiation at a particular time slot, we set a PV output equal to the percentage of its capacity for each PV. For example, on 7 July 2023 at 13:15 CEST in Belgium, PVs with total capacity of 7.677,34 MWp generated 5.658,64 MW energy, resulting in a conversion coefficient of 0.737 [29]. To incorporate further deviations in their outcome, we multiplied each PV outcome by a coefficient randomly chosen from the range [0.9 – 1.1].

Similarly, for the WT generation data, for each of the households with WTs, we randomly chose a WT capacity from the following set [1.0, 1.5, 2.0] KWp. Related to the strength of the wind at a particular time slot, we set an output equal to the percentage of its capacity for each WT. For example, on 7 July 2023 at 13:15 CEST in Belgium, WTs with total capacity of 2.525,9 MWp generated only 180,06 MW energy, resulting in a conversion coefficient of 0.0712 [30]. To incorporate further deviations in their outcome, we multiplied each WT outcome by a coefficient randomly chosen from the range [0.9 – 1.1].

The real demand/supply data of prosumers is then calculated by subtracting their PV/WTs generation data from their consumption/supply data per each time slot.

We generate the predicted volume of each household from their consumption/supply data. We assume that 10% of households have a perfect prediction algorithm; 60% – good, 20% – not so good, and 10% – quite bad. We randomly select 10% of the households and make their predicted volume equal to their actual consumption/supply data. We randomly select 60% of the remaining households whose predicted volume deviates from their actual consumption/generation profile with [1–10%]. We randomly select 20% of the remaining households whose predicted volume deviates from their actual consumption/generation profile with [11-20%]. The predictions of the remaining 10% of the households deviate greatly from their actual consumption/supply data [21-100%].

The code to transform the weather data into PV/WT data and generate the household production/consumption data is available here: `https://github.com/3MI-Labs/private-billings-data-generation`.

5.3.3. Simulation Framework

The private-billing library was created to demonstrate the performance of a privacy-preserving billing model. This Python library contains two primary modules:

1. a core module focused on executing the steps in the billing model, and
2. server and client implementations that can be used to (automatically) exchange data between the parties involved in the billing algorithm.

The library provides the Universal Cost Split (UCS) billing model [7] as a default implementation. It has been organized such that future work can expand it, to implement and test other innovative privacy-preserving billing models. To our knowledge, this is the first such library in existence. It is publicly available at `https://github.com/3MI-Labs/private-billings`.

To execute the experiment set and measure the desired data, the library was slightly modified. These modifications do not impact the performance of the library. The source of the experiment library can be found here: `https://github.com/3MI-Labs/private-billings-experiment`.

5.3.4. Simulation Results

We have also performed simulations to demonstrate the feasibility of our proposed solution in a real-world P2P market setting.

Computational cost. We split the computational cost into two parts: cost per trading slot and cost per billing period.

Our experiments were run one at a time on a near-idle PC. The machine we ran our simulations on had the following specifications: 13th Gen Intel® Core™ i7-13700H @ 5 GHz and 16 GB RAM.

We split our experiments based on the number of participating households, i.e., 12, 25, 50, or 100. As depicted in Fig. 3, we obtained the following results:

- The time taken to perform encryption (hide data) of all data increases sub-linearly with the number of households. For 12 households, it takes ~ 1 second compared to ~ 3 seconds for 100 households.
- The average time for bill computation is ~ 0.17 seconds for all our experiments, i.e, irrespective of the number of households.

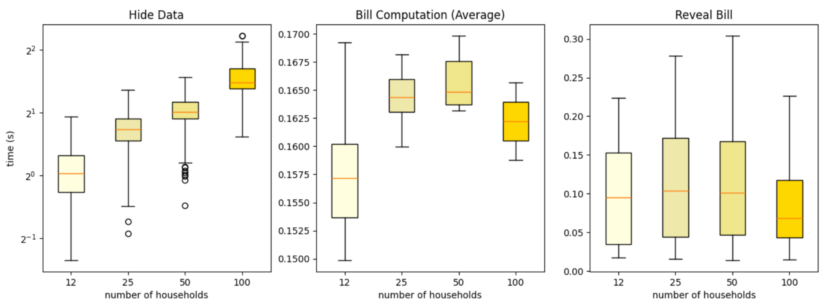

Figure 3: Average computation time for all households and billing periods

- The time taken to reveal the encrypted bills is nearly constant ~ 0.10 seconds per household.

6. Conclusions and Future Work

This paper presented a fully privacy-preserving billing model based on universal cost-split that handles discrepancies between committed and consumed/injected volumes of electricity. Our billing model can calculate bills/rewards for various participating households within ~ 0.17 seconds, showing its feasibility when deployed in a real-world setting. In addition, we also implemented a framework to enable private billings using FHE that can be adapted to support any billing model of choice.

In future work, we plan to execute the simulations of our billing model in a completely distributed fashion, i.e., the server and clients are situated in geographically different locations. Additionally, we assume that our adversary is honest but curious (or passive). While this fits our use case, as future work, we plan to formally prove our billing protocol to be secure in the presence of an actively malicious adversary that can take any arbitrary action to sabotage the execution of the protocol.

7. Acknowledgements

This work was supported by EPSRC through the project EnnCore EP/T026995/1 and by the Flemish Government through the FWO SBO project SNIPPET S007619.

References

[1] T. Capper, A. Gorbatcheva, M. A. Mustafa, M. Bahloul, J. M. Schwidtal, R. Chitchyan, M. Andoni, V. Robu, M. Montakhabi, I. J. Scott, et al., Peer-to-peer, community self-consumption, and transactive energy: A systematic literature review of local energy market models, Renewable and Sustainable Energy Reviews 162 (2022) 112403.

[2] J. M. Schwidtal, P. Piccini, M. Troncia, R. Chitchyan, M. Montakhabi, C. Francis, A. Gorbatcheva, T. Capper, M. A. Mustafa, M. Andoni, et al., Emerging business models in local energy markets: A systematic review of peer-to-peer, community self-consumption, and transactive energy models, Renewable and Sustainable Energy Reviews 179 (2023) 113273.

[3] M. Montakhabi, S. van der Graaf, M. A. Mustafa, Valuing the value: An affordances perspective on new models in the electricity market, Energy Research & Social Science 96 (2023) 102902.

[4] W. Kong, Z. Y. Dong, D. J. Hill, F. Luo, Y. Xu, Short-term residential load forecasting based on resident behaviour learning, IEEE Transactions on power systems 33 (1) (2017) 1087–1088.

[5] K. N. Hasan, R. Preece, J. V. Milanović, Existing approaches and trends in uncertainty modelling and probabilistic stability analysis of power systems with renewable generation, Renewable and Sustainable Energy Reviews 101 (2019) 168–180.

[6] Y. Wang, F. Zobiri, M. A. Mustafa, J. Nightingale, G. Deconinck, Consumption prediction with privacy concern: Application and evaluation of

federated learning, Sustainable Energy, Grids and Networks 38 (2024) 101248.

[7] A. Madhusudan, F. Zobiri, M. A. Mustafa, Billing models for peer-to-peer electricity trading markets with imperfect bid-offer fulfillment, in: 2022 IEEE International Smart Cities Conference (ISC2), 2022, pp. 1–7. doi:10.1109/ISC255366.2022.9922530.

[8] M. A. Mustafa, S. Cleemput, A. Abidin, A local electricity trading market: Security analysis, in: 2016 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), 2016, pp. 1–6. doi:10.1109/ISGTEurope.2016.7856269.

[9] S. Dai, F. Meng, Q. Wang, X. Chen, Dp2-nilm: A distributed and privacy-preserving framework for non-intrusive load monitoring, Renewable and Sustainable Energy Reviews 191 (2024) 114091.

[10] E. McKenna, I. Richardson, M. Thomson, Smart meter data: Balancing consumer privacy concerns with legitimate applications, Energy Policy 41 (2012) 807–814.

[11] A. Abidin, A. Aly, S. Cleemput, M. A. Mustafa, Secure and privacy-friendly local electricity trading and billing in smart grid, arXiv preprint arXiv:1801.08354 (2018).

[12] R. Thandi, M. A. Mustafa, Privacy-enhancing settlements protocol in peer-to-peer energy trading markets, in: 2022 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), IEEE, 2022, pp. 1–5.

[13] A. Hutu, M. A. Mustafa, Privacy preserving billing in local energy markets with imperfect bid-offer fulfillment, in: 2023 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), IEEE, 2023, pp. 1–9.

[14] K. Erdayandi, L. C. Cordeiro, M. A. Mustafa, A privacy-preserving and accountable billing protocol for peer-to-peer energy trading markets, in: 2023 International Conference on Smart Energy Systems and Technologies (SEST), IEEE, 2023, pp. 1–6.

[15] E. Alqahtani, M. A. Mustafa, Zone-based privacy-preserving billing for local energy market based on multiparty computation, in: 2023 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), IEEE, 2023, pp. 1–7.

[16] E. Alqahtani, M. A. Mustafa, Privacy-preserving billing for local energy markets (long version) (2024). `arXiv:2404.15886`.
URL `https://arxiv.org/abs/2404.15886`

[17] G. Kalogridis, M. Sooriyabandara, Z. Fan, M. A. Mustafa, Toward unified security and privacy protection for smart meter networks, IEEE Systems Journal 8 (2) (2013) 641–654.

[18] A. Rial, G. Danezis, Privacy-preserving smart metering, in: Proceedings of the 10th annual ACM workshop on Privacy in the electronic society, 2011, pp. 49–60.

[19] G. Danezis, M. Kohlweiss, A. Rial, Differentially private billing with rebates, in: Information Hiding: 13th International Conference, IH 2011, Prague, Czech Republic, May 18-20, 2011, Revised Selected Papers 13, Springer, 2011, pp. 148–162.

[20] Z. Erkin, G. Tsudik, Private computation of spatial and temporal power consumption with smart meters, in: Applied Cryptography and Network Security: 10th International Conference, ACNS 2012, Singapore, June 26-29, 2012. Proceedings 10, Springer, 2012, pp. 561–577.

[21] M. Jawurek, M. Johns, K. Rieck, Smart metering de-pseudonymization, in: 27th annual conf. on computer security applications, 2011, pp. 227–236.

[22] G. Danezis, C. Fournet, M. Kohlweiss, S. Zanella-Béguelin, Smart meter aggregation via secret-sharing, in: Proceedings of the first ACM workshop on Smart energy grid security, 2013, pp. 75–80.

[23] M. Ababneh, K. Kolachala, R. Vishwanathan, Private and secure smart meter billing, in: Proceedings of the 8th ACM on Cyber-Physical System Security Workshop, 2022, pp. 15–25.

[24] A. Abidin, A. Aly, S. Cleemput, M. A. Mustafa, An mpc-based privacy-preserving protocol for a local electricity trading market, in: Int. Conf. on Cryptology and Network Security (CANS), Springer, 2016, pp. 615–625.

[25] M. B. da Gama, J. Cartlidge, A. Polychroniadou, N. P. Smart, Y. T. Alaoui, Kicking-the-bucket: Fast privacy-preserving trading using buckets, in: Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers, Springer-Verlag, Berlin, Heidelberg, 2022, p. 20–37. `doi:10.1007/978-3-031-18283-9_2`.
URL `https://doi.org/10.1007/978-3-031-18283-9_2`

[26] M. Keller, D. Rotaru, N. P. Smart, T. Wood, Reducing communication channels in mpc, in: D. Catalano, R. De Prisco (Eds.), Security and Cryptography for Networks, Springer International Publishing, Cham, 2018, pp. 181–199.

[27] J. H. Cheon, A. Kim, M. Kim, Y. Song, Homomorphic encryption for arithmetic of approximate numbers, in: T. Takagi, T. Peyrin (Eds.), Advances in Cryptology – ASIACRYPT 2017, Springer International Publishing, Cham, 2017, pp. 409–437.

[28] R. D'hulst, W. Labeeuw, B. Beusen, S. Claessens, G. Deconinck, K. Vanthournout, Demand response flexibility and flexibility potential of residential smart appliances: Experiences from large pilot test in belgium, Applied Energy 155 (2015) 79–90.

doi:https://doi.org/10.1016/j.apenergy.2015.05.101.
URL https://www.sciencedirect.com/science/article/pii/S0306261915007345

[29] elia, Solar power generation, https://www.elia.be/en/grid-data/power-generation/solar-pv-power-generation-data (2024).

[30] elia, Wind power generation, https://www.elia.be/en/grid-data/power-generation/wind-power-generation (2024).