Outsourceable Privacy-Preserving Power Usage Control in a Smart Grid

Hu Chun¹, Kui Ren², and Wei Jiang¹(✉)

¹ Department of Computer Science, Missouri University of Science and Technology, Rolla, MO, USA
{wjiang,chwrc}@mst.edu
² Department of Computer Science and Engineering, SUNY Buffalo, Buffalo, NY, USA
kuiren@buffalo.edu

Abstract. The smart grid systems, in replace of the traditional power grid systems, have been widely used among some well-known telecommunication, IT and power industries. These systems have multiple advantages such as energy efficiency, reliability and self-monitoring. To prevent power outage, threshold based power usage control (PUC) in a smart grid considers a situation where the utility company sets a threshold to control the total power usage of a neighborhood. If the total power usage exceeds the threshold, certain households need to reduce their power consumption. In PUC, the utility company needs to frequently collect power usage data from smart meters. It has been well documented that these power usage data can reveal a person’s daily activity and violate personal privacy. To avoid the privacy concern, privacy-preserving power usage control (P-PUC) protocols have been introduced. However, the existing P-PUC protocols are not very efficient and the computations cannot be outsourced to a cloud server. Thus, the utility company cannot take advantage of the cloud computing paradigm to potentially reduce its operational cost. The goal of this paper is to develop a P-PUC protocol whose computation/execution is outsourceable to a cloud. In addition, the proposed protocol is much more efficient than the existing P-PUC protocols. We will provide extensive empirical study to show the practicality of our proposed protocol.

Keywords: Smart grid · Privacy-preserving · Power usage control

1 Introduction

A smart grid can improve the efficiency, reliability, economics, and sustainability of a utility company to produce and distribute electricity. In a smart grid, smart meters can collect the power usage data of each household in a neighborhood to help a utility company self-monitor the power supply of the neighborhood to prevent power outage. For example, when the total power usage is extremely high, physical components in a smart grid could be overloaded. In order to
prevent the failures of these physical components (consequently the power outage of the entire system), the power consumption of some households needs to be reduced (e.g., by setting the temperature of an AC a little bit higher without affecting a person’s well-being).

In general, a utility company can set a power usage threshold beyond which the physical components of a smart grid may work dangerously above their expected capacities. Then the threshold can be compared with the total power usage of a neighborhood at a particular time that can be computed based on the power usage readings from the smart meters of individual households. When the total power usage exceeds the threshold, the energy consumptions of certain households need to be reduced. To achieve this kind of threshold based power usage control (PUC), a utility company needs to collect power usage data frequently from the smart meters of individual households. However, it has been shown that by analyzing 15-minute interval household energy consumption data (even in an aggregated form), the usage patterns of most major home appliances can be determined [8, 21].

From these usage patterns, a malicious party can infer activities of daily living (ADL) information [1] and can potentially initiate any malicious acts toward a particular household. Therefore, it is in the best interest of the utility company not to collect each household’s power usage data. In addition, the threshold set by the utility company can reveal its operational capacity and the number of its customers in a neighborhood. To preserve competitive advantage, any information regarding the threshold values should not be disclosed to the public. Now the question becomes: can a utility company perform any PUC tasks without the company disclosing its threshold values and individual households disclosing their power usage data? Such a problem is termed as privacy-preserving power usage control (P-PUC).

Secure protocols have been proposed to solve the P-PUC problem [11, 26] under different power adjusting strategies. However, those protocols are executed directly between a utility company and its household customers. It is hard to see how to implement the existing protocols effectively in practice since they require each household to actively participate in online computations. To summarize, the exiting work has at least one of the following limitations:

- Not very efficient when the threshold values are from a large domain.
- Leak certain intermediate information that can be used to infer knowledge about the private power usage data of individual households and the threshold values set by the utility companies.
- Incur heavy computations between the households and the utility company.

To eliminate the above problems, in this paper, we develop a novel P-PUC protocol which allows computations to be completely outsourced to cloud servers. Recently, cloud computing has emerged as cost efficiency and operational flexibility approach for entities to outsource their data and computations for on-demand services. Because the power usage data can be very large in quantity (especially when these data are collected with high frequency), it is beneficial for
a utility company to outsource the data and the computations related to P-PUC protocols to a cloud.

As discussed before, the power usage data and the threshold values are sensitive information, so these data should not be disclosed to the cloud. Thus, before outsourcing, the data need to be encrypted, and the cloud only stores and processes the encrypted data. When the data are encrypted with fully homomorphic encryption schemes, the cloud can perform any arbitrary computations over the encrypted data without ever decrypting them. Nevertheless, fully homomorphic encryption schemes have yet to be practical due to their extremely high computational cost. As a result, we adopt a multi-server framework to securely and efficiently implement the proposed protocol.

1.1 Problem Definition

Suppose a neighborhood has $n$ households $P_1, \ldots, P_n$. For $1 \leq i \leq n$, let $a_i$ denote the average power consumption of $P_i$ at a specific time interval and $t$ be a threshold designated by the utility company for the neighborhood. We use $a$ to denote the power usage aggregation of the neighborhood where $a = \sum_{i=1}^{n} a_i$. If $a > t$, each $P_i$ is required to reduce its power consumptions by $\delta_i$ to prevent the possibility of power outage in the neighborhood. The value $\delta_i$ is determined by the following equation:

$$\delta_i = \frac{a_i}{a} \times (a - t) = a_i \times (1 - \frac{t}{a})$$ (1)

Here $\delta_i$ is a lower bound on the amount of power usage the user $P_i$ should cut. After each round of power reduction, the total average power usage of the neighborhood will at a safe level (e.g., $a < t$). This is the common strategy adopted by the existing P-PUC protocols [11].

In our problem setting, the input values $a_1, \ldots, a_n$ and $t$ should be hidden from the cloud servers. That is, before outsourcing, these values need to be either encrypted or secretly shared. In the proposed protocols, we adopt additive secret sharing scheme to hide the original values. The proposed outsourceable privacy-preserving power usage control (OP-PUC) protocol can be formulated as follows:

$$\langle P_1, \delta_1 \rangle, \ldots, \langle P_n, \delta_n \rangle \leftarrow \text{OP-PUC}(\langle P_1, a_1 \rangle, \ldots, \langle P_n, a_n \rangle, S_1, S_2, \langle U, t \rangle)$$ (2)

According to the above formulation, there are three types of participating entities: $n$ households, two cloud service providers $S_1$ and $S_2$, and a utility company $U$. The input for each household or customer $P_i$ is its average power consumption $a_i$ within a specific period of time, and the input of $U$ is a threshold $t$. The two cloud servers perform the necessary computations, and there are no explicit inputs for the two servers. After the execution of the OP-PUC protocol, each $P_i$ receives a value denoted by $\delta_i$, the minimum amount of the energy consumption that needs to be reduced by $P_i$. The other participating entities do not receive any outputs. (A max-usage based control strategy is discussed in Sect. 3).
Privacy and Security Guarantee. During the execution of the OP-PUC protocol, $a_i$ is private to $P_i$, and it should not be disclosed to the other households. In addition, $a_i$ should not be known to the two cloud servers and the utility company. Since $t$ is private to the utility company $U$, $t$ should not be known to the other participating entities.

- $a_i$ is only known to $P_i$, for $1 \leq i \leq n$, and
- $t$ is only known to $U$.

Threat Model. In the paper, we adopt the commonly accepted security definition of secure multiparty computation (SMC). More specifically, we assume the participating entities are semi-honest; that is, the entities follow the prescribed procedures of the protocol. Under the semi-honest model, it is implicit that the participating entities do not collude. Another adversary model of SMC is the malicious model. Under the malicious model, the entities can behave arbitrarily. Most efficient SMC-protocols are secure under the semi-honest model since less number of steps are needed to enforce honest behaviors. We have the following motivations to adopt the semi-honest model:

- The OP-PUC protocol needs to be sufficiently efficient. Between the semi-honest model and the malicious model, the semi-honest model always leads to much more efficient protocol.
- Smart meters can be made temper proof, so we can assume the households cannot modify the readings from smart meters and the messages sent from the smart meters to the two cloud servers. Thus, the semi-honest model fits our problem domain well regarding the households.
- The cloud service providers and the utility company are legitimate business. It is hard to see they collude and initiate any malicious act to discover the private smart meter readings. For well-known and reputable cloud servers (e.g., Amazon and Google), it makes sense to assume they follow the protocol and behave semi-honestly.

1.2 Our Contribution

In this paper, we develop an efficient OP-PUC protocol that incurs almost no computations on the households since the computations are completely outsourced to the cloud servers. The proposed protocol is secure under the semi-honest model and satisfies all the security requirements discussed in Sect. 3. Due to the fact that all computations are outsourced to the cloud servers and the computations are only performed on encrypted data, the existing P-PUC protocols cannot be applied to our problem setting. Plus, our proposed protocol is more efficient because it takes advantage of both secret sharing based secure computation and Yao’s Garbled Circuit [29].

The proposed protocol consists of three stages: (1) data collection and integration, (2) comparing $a$ and $t$, and (3) computing the $\delta_i$ values. At the first stage, the two cloud servers collect the average power consumption data $a_i$ from
each household $P_i$, and the threshold value $t$ from the utility company. This stage utilizes additive secret sharing which is extremely efficient to securely combine the data together to generate secret shares of the total power consumption $a$ of the neighborhood. The second stage determines the comparison result between $a$ and $t$. The third stage computes the $\delta$ values using the garbled circuit. The key functionality involved in this stage is secure division. The existing secure division protocols are very inefficient, and our work provides a new and more efficient implementation of secure division. Details regarding our proposed protocol is given in Sect. 3.

The rest of the paper is organized as follows: Sect. 2 discusses the work most related to the proposed problem domain. Section 3 provides the detailed implementation of the proposed OP-PUC protocol. Section 4 presents empirical results to show the practicality of OP-PUC. Section 5 summarizes the paper and points out some important future research directions.

2 Related Work and Background

In this section, we provide an overview of the existing work related to our proposed problem including privacy issues related to energy consumption data in a smart grid and the current P-PUC protocols. In addition, we present some background information on secure division and Yao’s garbled circuit.

2.1 Privacy Issues in Smart Grids

The use of smart grid infrastructure is growing rapidly; however, there exist potential privacy and security risks during the process of collecting power usage of data [1,13,16]. It is shown in [22] that data collected over a reasonable time interval (e.g., 15 or 30-minute) can be used to identify most household appliances. Another work [21] also shows that power consumption data collected in every 15-minute time interval can be used to uniquely identify home appliances with 90% accuracy. From these data, various information about a person’s daily activities can be inferred such as how many people are home, sleeping schedule, laundry and cooking routines [15,19,24]. If these data are in the wrong hand, the safety of a household will be at a very high risk. Therefore, power consumption data are considered private, and it is necessary to build privacy-preserving protocols to preserve user’s privacy in smart grids.

2.2 Privacy-Preserving Protocols in Smart Grids

Most privacy-preserving protocols in smart grids [14,17,23,25,27] are not focusing on the P-PUC problem. To our knowledge, the protocols presented in [11,26] are the only existing work closely related to the proposed problem. The first two P-PUC protocols are proposed in [11] built based on two strategies. The protocols leak the total energy consumption to one of the households, and the maximum
Table 1. Common notations

| Notation | Description |
|----------|-------------|
| SMC      | Secure Multi-Party Computation |
| P-PUC    | Threshold-Based Power Usage Control |
| OP-PUC   | Outsourceable P-PUC |
| $a_i$    | Power consumption of user $P_i$ within specific period |
| $a_i'$ and $a_i''$ | Secret shares of $a_i$ between two parties |
| $t$      | Threshold value provided by a utility company |
| $t'$ and $t''$ | Secret shares of $t$ between two parties |
| $S_1$ and $S_2$ | Two independent cloud servers |
| $U$      | The utility company |
| $P_i$    | One user in a neighborhood, $i = 1, \ldots, n$ |

energy consumption among the households is also revealed. In addition, they utilize a secure division protocol among several proposed protocols in [2]. Its secret sharing based secure division protocol requires at least three parties, and it is not applicable in our problem setting where we assume two independent cloud servers perform all necessary secure computations. Although there is an efficient two-party secure division protocol [2], one party needs to perform division operations between randomized values to obtain the final division result. However, the division result is known to the party which is not allowed in our problem domain. Also, we are not certain how to modify it to hide the final division result securely and efficiently.

In [26], another P-PUC protocol is developed to address the security issues of the earlier P-PUC protocols. However, the protocol is still not very efficient in that the individual households and the utility company involve in heavy computations. More importantly, all the exiting P-PUC protocols are not applicable in our problem domain where the computations are completely outsourced to the cloud which results in a more practical P-PUC protocol from the perspectives of individual households and utility companies.

2.3 Secure Division and Yao’s Garbled Circuit

In [11], the authors utilize a secure division protocol based on homomorphic encryption, but the protocol is not secure and efficient. Although the secure division protocol in [26] is secure, the protocol is efficient for very small domains. Also, one of the inputs of this secure division protocol is not encrypted. In our case, all input values are encrypted. To implement an efficient secure division protocol under the proposed problem domain. We adopt the garbled circuit approach introduced by Yao [29]. Recently, an intermediate language for describing and executing garbled circuits - the GCParser [18] has been proposed. This framework can implement any optimizations at both the high level and the low level, and it has already been applied to optimizing free XOR-gates and pipelin-
Algorithm 1. Secure_Split(α) → (α’, α’’)

Require: P has α and N where α < N

1: P:
   (a) α’ ← α + r mod N, where r ∈ R Z
   (b) α” ← N − r
   (c) Send α’ to S1 and send α” to S2

2: S1 and S2:
   (a) Receive α’ and α” respectively

ing circuit generation and execution. We adopt this framework to build a garbled circuit for secure division.

3 The Proposed OP-PUC Protocol

In this section, we adopt the same notations from the previous sections summarized in Table 1. The same as the existing work, the proposed OP-PUC protocol adopts the following two power usage control strategies when α > t: (1) reducing the power usage for the user who used the maximum amount of energy among all users, and (2) providing the specific power reduction amount each individual users in a particular neighborhood.

3.1 The First Stage of OP-PUC

In our problem settings, the cloud servers first need to gather the needed data from power customers and the utility company, then compare the total power consumption of those customers α with the threshold given by utility company t. If α > t, users need to reduce their power usage for the next period. The first stage of OP-PUC is data collection.

During the first stage, we emphasize that data must be hidden before outsourced. In the previous P-PUC protocols, homomorphic encryptions are utilized to encrypt the power usage data. However, if we extend the homomorphic encryption approach in this outsourced environment, huge computations would be incurred on the cloud servers. Therefore, to have a more efficient protocol, we adopt secret sharing approach for the data collection stage.

To get shared inputs, we use the Secure_Split protocol presented in Algorithm 1 where we assume N is a large number. In this protocol, P split its input value α to two random values α’ and α” so that α’ + α” = α mod N, and send them to S1 and S2 respectively. At the end, S1 holds α’, and S2 holds α”. They do not know anything about α except for N.

Algorithm 2 gives the main steps for the data collection stage of OP-PUC. For each user Pi, the power consumption value ai is split into ai’ and ai” and sent to servers S1 and S2 by using Secure_Split. At the same time, the utility company
Algorithm 2. Data_Collection → \{((a'_1, a''_1), \ldots, (a'_n, a''_n), (a', a''), (t', t''))\}

Require: \( P \) has \( a_i \) where \( 1 \leq i \leq n \), \( U \) has \( t \), and \( N \) is publicly known

1: \( P_i \): Secure_Split\((a_i)\)
2: \( U \): Secure_Split\((t)\)
3: \( S_1 \): \( a' = \sum_{i=1}^{n} a'_i \)
4: \( S_2 \): \( a'' = \sum_{i=1}^{n} a''_i \)

\( U \) also uses Secure_Split to send the secret shares of \( t \) to the two cloud servers. At the end, \( S_1 \) and \( S_2 \) compute \( a' = \sum_{i=1}^{n} a'_i \) and \( a'' = \sum_{i=1}^{n} a''_i \) separately. It is easy to see \( a = a' + a'' \mod N \) is the total power usage at a specific period. Since each server has one secret share of each value, they do not know anything about the original values.

3.2 The Second Stage of OP-PUC

The main task for the second stage of the proposed protocol is to securely determine whether \( a > t \) or not; thus, we need a secure comparison protocol to compare \( a \) and \( t \) with secret shares of each value as inputs. We consider to use garbled circuit to securely perform the comparison task because the existing secure comparison protocols [3,6,7,9,20] are not directly applicable in our problem domain. These protocols require that the inputs need to be the actual values. In addition, garbled circuit is known for its efficiency to securely evaluate simple functionalities such as secure comparison. A garbled circuit has only one round of communication. Details about constructing and evaluating a garbled circuit are given in [12]. In this paper, we assume that the secure comparison protocol built by a garbled circuit is denoted by Secure_Comparison\((a', a'', t', t'')\) → \( b \). The protocol is performed by \( S_1 \) and \( S_2 \), where \( a' \) and \( t' \) are the inputs of \( S_1 \), and \( a'' \) and \( t'' \) are the inputs of \( S_2 \). The protocol returns a bit \( b \) to the servers. If \( b = 1 \), the total power usage exceeds the threshold, and the OP-PUC protocol will proceed to the next stage.

3.3 The Third Stage of OP-PUC Based on Strategy 1

For strategy 1, the user with the most power consumption is selected and ordered to reduce his power usage. During the process, the cloud servers are not allowed to know which user is chosen. Basically, in this stage, a Secure_Maximum protocol is used to securely pick out the maximum value among \( n \) shared values. We utilize garbled circuit approach to implement Secure_Maximum. The key steps can be found in [11]. At the end, the maximum value will be known to each user. The user had the maximum energy consumption will reduce its power consumption. As stated in the existing work, how much energy consumption needs to be reduced is hard to decide. Thus, the second strategy is more practical.
Algorithm 3. Division\((t, a) \rightarrow q\)

Require: Bit representation of \(t\) is \(t_0, \ldots, t_{l-1}\) and bit representation of \(a\) is \(a_0, \ldots, a_{m-1}\) from the least to the most significant bits. Expand dividend \(t\) with \(m\) bits and set \(t_i = 0\) where \(l \leq i < l + m\) and expand another bit \(t_{l+m} = 0\) as sign bit of dividend.

1: for \(1 \leq i \leq m\):

(a) Shift left \(t\) for 1 bit
(b) if \(t_{l+m} = 0\) subtract \(t_{l+m-1} \ldots t_l\) with \(a\)
(c) else add \(t_{l+m-1} \ldots t_l\) with \(a\)

2: \(q \leftarrow t_{l-1} \ldots t_0\)

3.4 The Third Stage of OP-PUC Based on Strategy 2

When the total energy consumption exceeds the threshold \(t\), each user needs to reduce his or her power usage. How to decide a reasonable power reduction for everybody is really important. Here we adopt the function from the prior work which has been shown in Eq. 1. By using this equation, every user \(P_i\) will reduce at least \(\delta_i\) power which is decided by \(a_i\) weighted in \(a\). Since party \(P_i\) has his or her power consumption value \(a_i\) the, \(\frac{t}{a}\) needs to be calculated at the servers.

To securely compute \(\frac{t}{a}\), two secure division protocols using additive homomorphic encryption schemes were introduced in [4, 26]. However, we believe that a garbled circuit approach should be more efficient. The reason is that in the outsourced environment, inputs to the secure protocols are hidden from the cloud servers. Thus, it is not easy to extend the prior solutions to fit our problem domain. In particular, when both \(t\) and \(a\) are hidden, to compute division between two encrypted values under homomorphic encryption is very expensive. Whereas in the garbled circuit approach, we can directly use the secret shares of \(t\) and \(a\) as inputs to the circuits.

We build the division circuit based on the “shift and subtract” non-restoring method. Algorithm 3 gives a detail of this method. In general, if we want to calculate the quotient of \(l\)-bit number \(t\) and \(m\)-bit number \(a\), first we need to expand \(t\) with \(m + 1\) bits and perform an iterative algorithm. In each loop, \(t\) makes a left shift and subtract or add \(a\) from the \(l^{th}\) bit to the \((l + m - 1)^{th}\) bit based on the value of \(t_{l+m}\): if \(t_{l+m} = 0\) then subtraction, otherwise addition. After \(m\) rounds, the latest \(t_0\) to \(t_{l-1}\) store the quotient \(q\).

Here we provide an example of how this method works. If we want to calculate the quotient of 11 (e.g., 1011 in binary format) divided by 3 (i.e., 0011 in binary format), first we expand 1011 to 000001011 and shift left of this number. Then we get 00001011x1, using the left most \(l + m - 1 = 5\) bits to subtract 0011, we have 11110011x1. Now the first bit is 1, so we set \(x_1 = 0\) and shift left again. We then use the most 5 bits of 11100110x2 to add 0011 since \(x_1 = 0\). We get 11111110x2, and set \(x_2 = 0\). Shift and add again for \(x_2 = 0\), this round we get 00010100x3, \(x_3 = 1\) because the most significant is 0. For next and last round
Algorithm 4. OP-PUC-Stage-3(a_i, t', t'', a', a'') → δ_i

Require: S_1 has a' and t', S_2 has a'' and t'', P_i has a_i for 1 ≤ i ≤ n, N is public
1: S_1 and S_2:
   (a) do Secure_Division(t', t'', a', a'') → (q', q'')
   (b) Send q' and q'' to every power users
2: P_i:
   (a) Calculate δ_i = a_i * (1 - \frac{t}{a}) and reduce at least δ_i power usages

we need to shift and subtract, finally we get result of 00010001. Thus
the quotient of this example is 3 (i.e., 0011 in binary format).

Our garbled division circuit follows the basic rules of “shift and subtract”
non-restoring method, and it is denoted by Secure_Division(t', t'', a', a'') →
(q', q''). The inputs of the circuit are secret shares of t and a from S_1 and
S_2. The outputs are secret shares of q so that S_1 and S_2 cannot infer anything
about a and t. At the end, every power user will get q = q' + q'' mod N so as to
compute δ_i by Eq. 1. Algorithm 4 summarizes the main steps of the third stage
of the OP-PUC protocol.

3.5 Complexity Analysis

In this section we analyze both computation and communication complexities of
proposed OP-PUC protocol. First, we analyze the computation complexity for
different sub-protocols at each stage. At the first stage, each user P_i and P_per-
form the Secure_Split protocol, which just has two addition operations. Servers
S_1 and S_2 perform summations of n values, so the computation complexity of
the first stage is bounded by O(n) summations.

For the second stage, we need to consider the secure comparison protocol.
For the garbled circuit approach, the inputs are two random shares with size
bounded by N, so O(log N) gates are needed in the initial phase of the garbled
circuit to add the shares. This step will result in much smaller values than N,
so the total number of gates for the comparison circuit is bounded by O(log N).

Protocols under two strategies become different at the third stage. For strat-
egy 1, the maximum value among the n values needs to be found. This is achieved
by a number of secure comparison circuits. Thus, there are at least O(n log N)
gates in the initial stage. Since the numbers involved are much less than N, the
total number of gates is bounded by O(n log N). Each gate of the garbled circuit
is encrypted by AES encryption. Therefore, the computation complexity of stage
2 and stage 3 under strategy 1 is bounded by O(n log N) AES encryptions. The
total computation complexity of OP-PUC under strategy 1 is bounded by O(n)summations plus O(n log N) AES encryptions.

Under strategy 2 of stage 3, the secure division circuit needs to be built and
evaluated. As before, the computation complexity of initial stage is also bound
by \( O(\log N) \). Since the bit lengths of dividend and divisor are much less than \( N \), the computation complexity of the division circuit is also bounded by \( O(\log N) \). Each gate of the garbled circuit is encrypted by AES encryption. Therefore, the computation complexity of stage 2 and stage 3 is bounded by \( O(\log N) \) AES encryptions. The total computation complexity of OP-PUC under strategy 2 is bounded by \( O(n) \) summations plus \( O(\log N) \) AES encryptions.

To analyze the communication complexity, we need to know the size of the secret shares. Since the value of each share is bounded by \( N \), we need \( \log N \) bits to represent each share. Thus, at the first stage, the total communication complexity is bounded by \( O(n \cdot \log N) \) bits. Because the AES key size is a constant value varying from 128 to 256, the communication complexity for both stage 2 and stage 3 is bounded by \( O(\log N) \) bits and \( O(n \cdot \log N) \) bits under strategy 1 and strategy 2 respectively. Therefore, regardless of the strategies, the total communication complexity of OP-PUC is bounded by \( O(n \cdot \log N) \) bits.

3.6 Security Analysis

The security proof of the proposed protocols is straightforward. Here we only provide a high level discussion. In the second stage and third stage, comparison and division garbled circuit are used which is proved secure under the semi-honest model [12]. Since all the intermediate outputs of these protocols are random shares, Based on the sequential Composition Theorem [10], the OP-PUC protocols are also secure under the semi-honest model.

4 Experimental Results

In this section, we discuss the performance of the OP-PUC protocols in details under different parameter settings. Then, we evaluate the computation costs of the existing methods [11] and compare them with our proposed protocols.

In the OP-PUC protocols, each \( P_i \) and \( U \) only interact with the cloud servers for one round: sending their inputs and receiving the final outputs. Between the two cloud servers \( S_1 \) and \( S_2 \), a garbled circuit can be evaluated in about two rounds of communication. Therefore, regardless of different strategies and stages, the total number of interactions between the two cloud servers are constant or just several rounds.

Since the communication complexity of the proposed protocol is very small, and the communications between the individual users and the cloud servers at the first stage are parallelizable. Here we ignore the communication complexity. We simulate the computation complexity on a Linux machine with an Intel® Xeon® Six-Core™ CPU 3.07 GHz processor and 12GB RAM running Ubuntu 12.04 LTS. Since the main part of the protocol is based on the garbled circuits method, we implement the protocol on top of FastGC [12], a Java-based framework that allows users to define Boolean circuits. After the circuits are constructed, the framework encrypts the circuits, performs oblivious transfer, and evaluates the garbled/encrypted circuits. We also fix the size of inputs for
cloud servers to a 1024-bit modulus. In our experiments, we randomly generate the values of $a_i$’s and $t$ such that $a > t$ and $1 \leq a, t \leq 2^m$, where $m$ is an upper bound bit length of the domain size.

### 4.1 Performance of OP_PUC and OP_PUC$^2$

Let OP_PUC$^2$ denote the proposed protocol based on the second strategy. We first compute the computation costs of different parties involved in the OP_PUC protocol for $n = 50$ and varying $m$. That is, the running time of cloud servers $A$ and $B$ (or $S_1$ and $S_2$) is analyzed for one iteration (the same as the existing work). We do not consider the costs of individual users and the power company, since almost all the computations are outsourced to the cloud servers. As shown in Fig. 1, the computation costs of $A$ and $B$ are 6.043 and 7.767 s respectively.
for $m = 10$. Although the computation times of $A$ and $B$ are increasing with $m$, the portion of increasing is very small in comparison with the expansion of the domain size. For example, even when $m = 50$, the computation costs of $A$ and $B$ are 6.123 and 7.854 s respectively, and they are pretty close to what they were when $m = 10$. This is due to the inner structure of maximum circuit: with the domain size expanding, many new xor gates are plotted which are free for evaluation, whereas the number of costly and gates does not increase significantly.

![Fig. 3. Complexity: OP_PUC Vs. OP_PUC^2](image)

In a similar manner, the computation costs of $A$ and $B$ in the OP_PUC^2 protocol is analyzed for varying $m$ and with $n = 50$ and $\theta = 10$, where $\theta$ is the bit length of a scalar factor. Note that the output of the division circuit is an integer, and $a$ and $t$ might be very close, so we need a scalar factor to come up with more accurate quotient. Therefore, the inputs of the division circuit are one $m + \theta$-bit dividend and one $m$-bit divisor. The computation costs of different parties in OP_PUC^2 are shown in Fig. 2. The same as OP_PUC, the computation costs of individual users and the power company are negligible and not counted. On the other hand, for $m = 10$, the computation costs of $A$ and $B$ are 2.497 and 4.195 s respectively. Similarly, the costs of $A$ and $B$ grow slightly with the increasing of $m$. For instance, the computation time of $A$ is 2.688 s when $m = 50$, and it is only increased by 0.191 second with a 40-bit size expansion.

We now compare the total computation costs of cloud providers $A$ and $B$ in OP_PUC (for one iteration) and OP_PUC^2 for $m = 10$ and varying $n$, where $n$ denotes the number of households from a given neighborhood. As shown in Fig. 3, the total running time of OP_PUC varies from 13.81 to 62.635 s when $n$ is changed from 50 to 250. On the other hand, the total running time of OP_PUC^2 remains to be nearly constant at 6.692 s in average since $t$ is independent of $n$. Following from Fig. 3, it is clear that the total run time of OP_PUC (even for one iteration) is always greater than that of OP_PUC^2. According to the above
analyses, we conclude that the proposed protocols are very practical especially for OP_PUC². Besides, there is nearly no computation costs for the individual users and the utility company.

4.2 Performance Comparison with the Existing Work

Finally, we compare the computation costs of our protocols with the existing work [11]. For \( n = 50 \) and \( m = 10 \) (note that when \( m = 10 \), the domain size is \( 2^{10} = 1024 \) already slightly bigger than \( l = 1000 \) in previous paper), the performance of OP_PUC is close to the STPUC_{max}, which is roughly 13–15 s. We notice that the running time of OP_PUC increases quickly when number of households increases. However, according to the domain size, OP_PUC is more scalable: the running time is nearly stable e.g., even when domain size is increased by a factor of \( 10^4 \) with the size of the neighborhood fixed to 50, the running time of OP_PUC just increases less than 1 second. Note that in STPUC_{max}, execution time is significantly increased with the increase of the domain size. Experiments showed that when domain size changes from 1024 to 4096, and fix the number of neighborhood to 50 users, the running time of STPUC_{max} increases from 11.02 s to 33.75 s. Also OP_PUC² is more efficient and scalable than STPUC_{div}. For example, when the domain size is 5000, OP_PUC² is faster than STPUC_{div} by a factor of 3 to 4. Besides, although the problem definition of our work is different from the existing work, our protocols achieve the same power usage control in a more efficient way.

5 Conclusion

In this paper, we developed outsourceable, privacy-preserving power usage control (OP-PUC) protocols. Comparing to the existing work, the proposed protocols are more efficient and as secure. More importantly, the computation costs for the users and the utility company are negligible. As a future research direction, we will develop OP-PUC protocols secure under the malicious model and utilize more than two cloud servers to further improve the computation costs.

If there are at least three cloud servers, all secure computations can be performed on secure shares. Secret sharing based secure computations can be more efficient than the garbled circuit. we will investigate if the efficiency of the OP-PUC protocols can be improved under the secret sharing model. To develop OP-PUC protocols secure under the malicious model, we may adopt threshold homomorphic encryption [5] or Shamir secret sharing [28]. We will investigate the pros and cons under each direction.

Acknowledgments. The authors would like to thank the anonymous reviewers for their insightful comments and suggestions. In addition, the first and third authors’ contribution to this work was supported by NSF under award No. CNS-1011984.
References

1. Guidelines for smart grid cyber security the smart grid interoper-ability panel cyber security working group. In: NISTIR 7628, August 2010
2. Atallah, M., Bykova, M., Li, J., Frikken, K., Topkara, M.: Private collaborative forecasting and benchmarking. In: Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, pp. 103–114. ACM (2004)
3. Blake, I.F., Kolesnikov, V.: One-round secure comparison of integers. J. Math. Cryptolol. 3(1), 37–68 (2009)
4. Bunn, P., Ostrovsky, R.: Secure two-party k-means clustering. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 486–497. ACM (2007)
5. Cramer, R., Damgård, I.B., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–299. Springer, Heidelberg (2001)
6. Damgård, I.B., Geisler, M., Kroigaard, M.: Efficient and secure comparison for online auctions. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 416–430. Springer, Heidelberg (2007)
7. Damgard, I., Geisler, M., Kroigard, M.: Homomorphic encryption and secure comparison. Int. J. Appl. Crypt. 1(1), 22–31 (2008)
8. Drenker, S., Kader, A.: Nonintrusive monitoring of electric loads. IEEE Comput. Appl. Power 12(4), 47–51 (1999)
9. Garay, J.A., Schoenmakers, B., Villegas, J.: Practical and secure solutions for integer comparison. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 330–342. Springer, Heidelberg (2007)
10. Goldreich, O.: Foundations of Cryptography: Volume 2, Basic Applications., vol. 2. Cambridge University Press, New York (2009)
11. Chun, H., Jiang, W., McMillin, B.: Privacy-preserving power usage control in the smart grid. In: Butts, J., Shenoi, S. (eds.) Critical Infrastructure Protection VI. IFIP Advances in Information and Communication Technology, vol. 390, pp. 127–137. Springer, Berlin Heidelberg (2012)
12. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: USENIX Security Symposium, vol. 201 (2011)
13. Jokar, N., Arianpoo, P., Leung, V.: A survey on security issues in smart grids. Secur. Comm. Netw. 7(4), 414–424 (2012). doi:10.1002/sec.559.V.C.M
14. Kursawe, K., Danezis, G., Kohlweiss, M.: Privacy-friendly aggregation for the smart-grid. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 175–191. Springer, Heidelberg (2011)
15. Lisovich, M.A., Mulligan, D.K., Wicker, S.B.: Inferring personal information from demand-response systems. IEEE Secur. Privacy 8(1), 11–20 (2010)
16. Liu, J., Xiao, Y., Li, S., Liang, W., Chen, C.L.P.: Cyber security and privacy issues in smart grids. IEEE Commun. Surv. Tutorials 14(4), 981–997 (2012)
17. Rongxing, L., Xiao-hui Liang, X., Li, X.L., Shen, X.: Eppa: an efficient and privacy-preserving aggregation scheme for secure smart grid communications. IEEE Trans. Parallel Distrib. Syst. 23(9), 1621–1631 (2012)
18. Melicher, W., Zahur, S., Evans, D.: An intermediate language for garbled circuits. In: IEEE Symposium on Security and Privacy Poster Abstract (2012)
19. Molina-Markham, A., Shenoy, P., Fu, K., Cecchet, E., Irwin, D.: Private memoirs of a smart meter. In: Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building, pp. 61–66. ACM (2010)
20. Nergiz, A.E., Nergiz, M.E., Pedersen, T., Clifton, C.: Practical and secure integer comparison and interval check. In: 2010 IEEE Second International Conference on Social Computing (SocialCom), pp. 791–799. IEEE (2010)
21. Quinn, E.L.: Privacy and the new energy infrastructure. SSRN E. J. (2009). http://dx.doi.org/10.2139/ssrn.1370731
22. Quinn, E.L.: Smart metering and privacy: existing laws and competing policies. SSRN eLibrary (2009). http://dx.doi.org/10.2139/ssrn.1462285
23. Rial, A., Danezis, G.: Privacy-preserving smart metering. In: Proceedings of the 10th annual ACM Workshop on Privacy in the Electronic Society, pp. 49–60. ACM (2011)
24. Rouf, I., Mustafa, H., Xu, M., Xu, W., Miller, R., Gruteser, M.: Neighborhood watch: security and privacy analysis of automatic meter reading systems. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 462–473. ACM (2012)
25. Salinas, S., Li, M., Li, P.: Privacy-preserving energy theft detection in smart grids. In: 2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), pp. 605–613. IEEE (2012)
26. Samanthula, B.K., Chun, H., Jiang, W., McMillin, B.M.: Secure and threshold-based power usage control in smart grid environments. Int. J. Parallel, Emergent Distr. Syst. 29(3), 264–289 (2014)
27. Saputro, N., Akkaya, K.: Performance evaluation of smart grid data aggregation via homomorphic encryption. In: 2012 IEEE Wireless Communications and Networking Conference (WCNC), pp. 2945–2950. IEEE (2012)
28. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)
29. Yao, A.C.-C.: How to generate and exchange secrets. In: 1986 27th Annual Symposium on Foundations of Computer Science, pp. 162–167. IEEE (1986)