2015

Privacy-preserving power usage control in smart grids

Huchun

Department: Computer Science

PRIVACY-PRESERVING POWER USAGE CONTROL IN SMART GRIDS

by

HUCHUN

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2015

Approved by

Dr. Wei Jiang, Advisor
Dr. Bruce M. McMillin
Dr. Maggie Cheng
Dr. Dan Lin
Dr. Xuerong (Meggie) Wen

ABSTRACT

The smart grid (SG) has been emerging as the next-generation intelligent power grid system because of its ability to efficiently monitor, predicate, and control energy generation, transmission, and consumption by analyzing users' real-time electricity information. Consider a situation in which the utility company would like to smartly protect against a power outage. To do so, the company can determine a threshold for a neighborhood. Whenever the total power usage from the neighborhood exceeds the threshold, some or all of the households need to reduce their energy consumption to avoid the possibility of a power outage. This problem is referred to as threshold-based power usage control (TPUC) in the literature. In order to solve the TPUC problem, the utility company is required to periodically collect the power usage data of households. However, it has been well documented that these power usage data can reveal consumers' daily activities and violate personal privacy. To avoid the privacy concerns, privacy-preserving power usage control (P-PUC) protocols are proposed under two strategies: adjustment based on maximum power usage and adjustment based on individual power usage. These protocols allow a utility company to manage power consumption effectively and at the same time, preserve the privacy of all involved parties. Furthermore, the practical value of the proposed protocols is empirically shown through various experiments.

ACKNOWLEDGMENTS

I would like to express the deepest appreciation to my my advisor, Dr. Wei Jiang, who helped me continuously to overcome various obstacles. Without his guidance and persistent help this dissertation would not have been possible.

I would also like to thank the members of my Ph.D. committee, Dr. Bruce M. McMillin, Dr. Maggie Cheng, Dr. Dan Lin, and Dr. Xuerong (Meggie) Wen, for their great suggestions and encouragement.

I appreciate the advice and support of Yousef Elmehdwi, Bharath Samanthula, Renjie Wang, Wenyong Lu, and Quanmin Ye. I also thank the staff members of the computer science department and Amy Ketterer, in the graduate editing services, for their great assistance.

I owe a deep sense of gratitude to my father Shuanglong, to whom I would like to dedicate this work for his undying love and faith in me, my mother Shuqing, my dear wife Wuriyihan, my parents-in-law Wulijibayaer and Gaowa, and my lovely kid, Daniel, and all other family members. With their support over the years, I have defeated the difficulties in my study and life. Thank you all.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

Table Page

1. INTRODUCTION

In recent years, there has been a rapid growth in the use of smart grid technology [1, 2, 3, 4, 5], which is envisioned as the future's energy-efficient and self-monitoring power system. As the smart grid offers tremendous benefits, it has attracted a significant amount of attention from researchers across academia, industry, and government agencies. Much work has been done to address various issues and implementation details related to the deployment of smart grids in various sectors [5, 6, 7]. A lot of telecommunication, power, and IT companies are already enjoying the benefits of smart grid technology by deploying it in their daily business operations. Nevertheless, more research needs to be done to address the various open challenges such as security and privacy issues [8, 9, 10].

This dissertation primarily focuses on the power outage issue in the smart grid environment. Consider a utility company that self-monitors the power supply to a neighborhood (i.e., uses smart grid technology) by collecting the power usage data of each household in the neighborhood. When the total power usage from the neighborhood is extremely high, the physical components in a smart grid could be overloaded, which would lead to a long-lasting regional power failure (i.e., the power outage of the entire system) and huge financial losses. Therefore, preventing this kind of power outage by controlling the power usage of households (especially in an electricity peak demand time) is an important task since it is beneficial for both the utility company and its customers.

In order to prevent the failure of these physical components, one kind of power usage control is to dynamically adjust the consumers' power consumption. For each servicing neighborhood, the utility company can determine a power usage threshold beyond which the physical components of a smart grid may work dangerously above their expected capacities. The threshold can be compared with the total power usage of a neighborhood at a particular time, which can be computed based on the power usage readings from the smart meters of individual households. Whenever a neighborhood's total power usage exceeds its related threshold, some or all of the households in the neighborhood need to reduce their energy consumption based on their current usage. To effectively achieve this threshold-

based power usage control (TPUC), the utility company should first provide consumers with incentives to participate. Then, the utility company needs to frequently collect and analyze the power usage data from each household in the participating neighborhood. To attract participation, the utility company can reduce the rate for participating consumers. In return, the consumers agree to reduce their power consumption when necessary, i.e., household with more power consumption in the neighborhood needs to reduce more of its energy consumption to bring down the total power usage of the neighborhood when the total usage exceeds the threshold.

However, existing research [11] has shown that specific types of appliances and generators can be identified through their signatures exhibited in the electric usage data (collected from a meter) when the collection frequency is very high (when the collection period is less than 1 hour) compared to the monthly-based collections. Similar research [12] also demonstrated that the usage patterns of most major home appliances can be determined by analyzing 15-minute interval aggregate household energy consumption data. More detailed power consumption information increases the likelihood of discovering consumers' private information because activities of daily living (ADL) can be inferred (such as how many people are in the building, when household owners are asleep, etc) from appliance usage patterns. Therefore, customer privacy might be disclosed to the utility company through the frequently collected power usage data. Even if the utility company is allowed to hold customers' personal information, it may still not want to store or use these data directly because the company is liable to the negative consequences if the data are disclosed to a malicious party. If those pieces of sensitive information are disclosed to adversaries, malicious attacks can be launched more easily. In addition, the threshold set by the utility company can reveal its operational capacity and its number of customers in a neighborhood. To preserve a competitive advantage, any information regarding the threshold values should not be disclosed to the public. Thus, it is beneficial or even necessary to develop a protocol that achieves TPUC without individual households disclosing their power usage data or the company disclosing its threshold values. This process is called a privacy-preserving TPUC, denoted as P-PUC.

Secure protocols to solve the P-PUC problem under different power adjusting strategies are proposed [13, 14]. Those protocols are executed directly between a utility company and its household customers. However, at least one of the following limitations appears in those protocols:

- 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 these problems, a novel P-PUC protocol that allows computations to be completely outsourced to cloud servers is also considered. Recently, cloud computing has emerged as an approach that offers cost efficiency and operational flexibility 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 computations related to P-PUC protocols to a cloud.

As discussed before, the power usage data and threshold values are sensitive information, so these data should not be disclosed to the cloud. Thus, before outsourcing, the data need to be encrypted, so the cloud would only be storing and processing 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, a multi-server framework to securely and efficiently implement the proposed protocol is also developed and termed as outsourceable P-PUC (OP-PUC) [15].

1.1. PROBLEM DEFINITION

This dissertation contains a proposal for P-PUC protocols under both basic setting and multiple cloud servers setting. Therefore, the basic setting comes first and is followed by the multi-server setting. Some other overall assumptions appear last.

1.1.1. Basic Setting. Without loss of generality, let $A_1, \ldots, A_n$ be n households from a neighborhood with $a_1, \ldots, a_n$ as their power consumption in a specific time interval, respectively (e.g., 15 minutes). In addition, let t denote the threshold for the neighborhood that is commonly determined by the utility company C (note that the threshold value may vary from different neighborhoods), and let a denote the total power usage for the neighborhood during the same time interval (i.e., $a = \sum_{i=1}^{n} a_i$). Whenever $a > t$, some or all consumers are required to reduce their power consumption in order to prevent the possibility of a power outage in the neighborhood. Suppose δ_i is the reduction in power consumption for consumer A_i, for $1 \leq i \leq n$. In general, the δ_i values can be zero*, and the actual reduction in power consumption for each A_i depends on either a_i (i.e., power consumed by A_i during a fixed period) or its current usage (the user might sign an agreement to reduce the power of devices that appear on a list). This purely depends on the underlying strategy used.

The literature presents several strategies to prevent a power outage in a smart grid. Nevertheless, such techniques assume the availability of household power consumption data to the utility company (or intermediate substation or data concentrator) to perform diagnostics in order to predict and prevent power outages. In the proposed problem setting, each user's power consumption data are treated as his/her private information. Under such a scenario, the standard schemes are not applicable. Therefore, this dissertation gives two strategies described below:

- Strategy 1 - Adjustment Based on Maximum Power Usage: Whenever $a > t$, the consumer with the maximum power usage is requested to reduce his/her power consumption. After this, if the updated average total power usage a is still greater than

*Observe that whenever $t \geq a$, $\delta_i = 0$, for $1 \leq i \leq n$.

t, then the above process is repeated in an iterated fashion. In each iteration, the consumer with the maximum power usage is requested to cut down the power for some devices that do not affect his/her daily life, such as shutting down the dryer or washer or adjusting the temperature of the air conditioner.

- Strategy 2 - Adjustment Based on Individual Power Usage: Under this strategy, each consumer A_i computes the lower bound on the amount of power usage to be reduced (i.e., δ_i) locally based on his/her last usage a_i. Whenever $a > t$, A_i computes his/her least amount of energy to be reduced as follows:

$$\begin{aligned} \delta_i &= \frac{a_i}{a} * (a - t) \\ &= a_i * \left(1 - \frac{t}{a}\right) \end{aligned}$$

Under Strategy 2, after the reductions are made by each consumer, the updated total power usage a will always be less than t.

It is straightforward to develop TPUC protocols based on the previous two strategies. However, building a P-PUC protocol by taking a user's privacy into consideration is a challenging task. Though the two strategies are different, a P-PUC can be formally defined as follows:

$$(\langle A_1, \delta_1 \rangle, \ldots, \langle A_n, \delta_n \rangle) \leftarrow \text{P-PUC}(\langle A_1, a_1 \rangle, \ldots, \langle A_n, a_n \rangle, \langle C, t \rangle) \quad (1)$$

In general, for any given P-PUC protocol, the following requirements need to be satisfied to preserve the user's privacy:

- The user's average power consumption a_i should not be revealed to the other consumers or C.

- The threshold t is private to C, and it should not be revealed to the consumers (t might be used to infer the power capacity of the power plant or other private knowledge of the neighborhood).

- The intermediate results, which are different for the above two strategies, should not be released to either consumers or C (to avoid any inference attacks).

First of all, the input values a_i and t are private information of A_i and C, respectively; therefore, they should be protected from other parties for obvious security reasons. In addition, under a secure multi-party computation (MPC) framework, a protocol is considered secure against semi-honest adversaries if the intermediate results seen by a party are either random or pseudo-random. This is an inherent security property that is required to prevent any inference attacks. For example, consider the intermediate result $a = \sum_{i=1}^{n} a_i$. For simplicity, let $n = 2$. Now, revealing the value of a to A_i's will also reveal one party's private input to the other party. This clearly violates the user's privacy. Therefore, all the intermediate results must be protected while constructing the secure protocols to satisfy the security definition of MPC. More details about the security definition adopted in this dissertation are given in Section 3.2.1.. At the end of the P-PUC protocol, $a \leq t$. During this process, only consumer A_i knows δ_i (as his/her output), and nothing else is revealed to C and A_i, for $1 \leq i \leq n$.

1.1.2. Multi-Server Setting. When an outsourcing environment is considered, the input values $a_1, \ldots, a_n$ and t should be hidden from the cloud servers because these cloud servers cannot be trusted. That is, before outsourcing, these values need to be either encrypted or secretly shared. In the proposed OP-PUC protocols, an additive secret sharing scheme is adopted to hide the original values. The proposed OP-PUC protocol can be formulated as follows:

$$(\langle A_1, \delta_1 \rangle, \ldots, \langle A_n, \delta_n \rangle) \leftarrow \text{OP-PUC}(\langle A_1, a_1 \rangle, \ldots, \langle A_n, a_n \rangle, S_1, S_2, \langle C, t \rangle) \tag{2}$$

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 C. The input for each household or customer A_i is its average power consumption a_i within a specific period of time, and the input of C 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 A_i receives a value denoted by δ_i for the minimum amount of the energy consumption that needs to be reduced by A_i. The other participating entities do not receive any outputs.

During the execution of the OP-PUC protocol, a_i is private to A_i and 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 C, t should not be known to the other participating entities.

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

1.2. ASSUMPTIONS

In this dissertation, choosing an optimal value for t is not discussed. In general, the value of threshold t is set by the utility company C. Approaches based on threshold are not new and have been commonly used in various problems within the smart grid environment (e.g., [16, 17, 18]). Nevertheless, deciding the value of t is an important step in threshold-based approaches, including the proposed protocols. The existing threshold-based approaches do not address how to decide the value of t. In particular to the proposed protocols, the value of t may vary between the neighborhoods. Some important factors that decide the value of t are (i) number of households in the neighborhood, (ii) percentage of residential, industrial, or commercial consumers in the neighborhood, and (iii) time and location of the neighborhood. These three factors will enable C to set an appropriate value of t for a given neighborhood. Alternatively, the utility company can use the existing historical dataset available from the neighborhood to predict a suitable t value using the data mining techniques.

Since the usages are random, even when the total power usage exceeds the threshold, there may not be an outage (as the usage may fall below the threshold shortly). This is termed as false alarm. Even in the case of false alarm, reducing the power consumed by households will only act as a precautionary step, and there are no negative consequences

in doing so. The goal of this dissertation is to develop P-PUC protocols. Therefore, estimating the false alarm probability, which itself is a separate and interesting problem, is outside the scope of this dissertation.

Also, in this dissertation, network delays were not considered. First of all, the threshold value t is not an upper-bound on power consumption. Instead, t was chosen to help the utility company efficiently manage and distribute the power for any given neighborhood. Therefore, even if there is a delay, there will be a sufficient gap between t and the upper-bound on power consumption. Nevertheless, the network delays in smart grid environments are usually in milliseconds. Hence, even in the case of network delays, the households will shortly cut down their usages if necessary (assuming $a > t$) and the total power usage will fall below t.

The threat model adopted for the dissertation is the commonly accepted security definition of secure multiparty computation (SMC). More specifically, the participating entities are assumed to be 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 fewer steps are needed to enforce honest behaviors. The following are motivations to adopt the semi-honest model:

- The P-PUC protocols need 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 tamper proof, so the households cannot modify the readings from smart meters or the messages sent from the smart meters to the power company. Thus, the semi-honest model fits this 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.

2. RELATED WORK

A smart grid is more efficient, more resilient, and more affordable to manage and operate than a traditional power grid. In the smart grid, the power company can not only gather power consumption information from power users (as in a traditional power grid), but also send information flow to power users for smart control and emergency services. However, big benefits come along with tremendous risks. The smart grid could also be vulnerable to various cyber security threats. New functions (e.g. smart control of household devices) provide many opportunities for cyber attacks such as spoofing attacks [19] and man-in-the-middle attacks [20]. Those vulnerabilities, when mastered by malicious attackers, could be used to launch cyber attacks to steal energy, affect the quality of service (QoS) of a power grid, or even lead to cascading failures of the whole power network, which could result in huge financial losses.

At the same time, to achieve smart control and better load balancing, a smart grid needs to frequently collect and analyze energy consumption data of individual power users. However, the existing literature showed that with the collection of 15-minute interval household energy consumption data, major home appliances will be detected with accuracy rates of over 90 percent [12]. In addition, it has been shown that the identification success rate is nearly perfect regarding larger two-state household appliances such as dryers, refrigerators, air conditioners, water heaters, and well pumps[21]. A survey on different types of information that can be inferred from the power consumption data is given in [22]. For example, using the household power consumption data, other household activities such as how many people are at home, sleeping routines, and eating routines can also be inferred [19, 23].

Because of these security and privacy risks, the National Institute of Standards and Technology (NIST) rolled out guidelines [6] for some security and privacy issues, which could be classified into three categories: Integrity, Confidentiality and Availability. In this dissertation, customer privacy issues are the main focus. Privacy issues are included in Confidentiality. However, in this dissertation, Privacy is highlighted as one distinct point.

That is because Confidentiality is to protect information against unauthorized parties, whereas Privacy can also be violated by authorized parties:

- Integrity: Protecting against the unauthorized modification or destruction of information. In the smart grid, data should not be modified without authorization; the source of data should be authenticated; the time stamp associated with the data should be known and authenticated; and the quality of the data should be known and authenticated. Loss of Integrity would lead to unauthorized modification or destruction of the information. For example, if a malicious customer alters his meter readings, he can commit energy theft to gain profit. On the other hand, if an attacker successfully controls a group of meters and injects fake data, the state estimation made by the controllers will be incorrect so that failures might happen in the power grid.

- Confidentiality: Protecting privacy and proprietary information by authorized restrictions on information access and disclosure. In the smart grid, customer information, power consumption data, and network topology information should not be disclosed to unauthorized parties. Loss of confidentiality would disclose the information to unauthorized parties. For example, robbers who learn the individual power consumption data can infer the daily lifestyle of the customer so as to rob the house when nobody is there. Attackers who gain control of the network topology can have a view of the entire network and find weak points for invasion. Competitor companies who control the customer information can make opposite strategies to seize markets.

- Privacy: Partly included in Confidentiality. On the other hand, authorized parties in the smart grid (e.g. utility companies) cannot be fully trusted. With Confidentiality breaks, attackers who master customer information can make inferences about individuals' daily living. Authorized parties like utility companies are also potential risks for users' privacy. Utility companies may sell customer data to third party advertisers to gain profit, or the employees of utility companies can get the customer data by chance. If individuals' time-series collections of power usage data are disclosed to

adversaries, bad things might happen. Therefore, the privacy requirement is highlighted: high frequently collection of power consumption data cannot be disclosed, even to authorized parties.

- Availability: Ensuring timely and reliable access to information and services. Availability will not be discussed further here, since the issue is far from privacy.

Some survey papers have already studied the security and privacy issues in a smart grid. *Cyber security in the smart grid: Survey and challenges* [24] gave a review of security issues in the smart grid. This paper discussed many potential attacks and made a use case study for their critical security requirements. Next, it went through existing counter attack solutions from both networking (including attack detection and attack mitigation mechanisms) and cryptography (including encryption, authentication, and key management). This paper also proposed research challenges from different aspects.

A survey on cyber security for smart grid communications [25] is another survey focusing on the cyber security issues of smart grid communications. The authors proposed security requirements of smart grid communications among confidentiality, availability, and integrity. Then they showed the challenges such as internetworking, security policy and operations, and security services. They studied the current solutions of privacy, integrity, authentication, and trusted computing as well.

At the rest of this chapter, existing works about spoofing attacks in smart grid communication networks will be reviewed at first. Then, existing security authentication schemes to keep smart grid system integrity will be studied. Literature that addresses privacy problems will be demonstrated last.

2.1. SPOOFING ATTACKS

A spoofing attack is a scenario in which an attacker successfully pretends to be another and gains unauthorized access. In a smart grid network, the goal for a spoofing attack might be:

- Smart meter spoofing: The attacker masquerades as a smart meter and can send fake data to the readers and collectors. The purpose of the attack can be energy theft or even to mislead power grid state estimation.

- Data reader/collector spoofing: The attacker acts as a data collector and learns the users' private information, which might be used in other inference attacks (e.g. robbery).

- Controller spoofing: The attacker pretends to be a controller and learns messages between other controllers and smart meters to make inference attacks; or even worse, the attacker can gain access to modify network packets to launch Denial of Service (DoS) attacks.

Although the smart meters are assumed to be protected from physical damages (tamper-resistant) in many existing works, since legacy meters will still be part of the power grid system in the near future, old meters are vulnerable points for the whole grid. Unlike traditional power grid, a smart grid introduces new features (for example, smart meters also have storage and computation abilities and can send and receive messages from other smart devices) in order to make smart control and demand requests. To fulfill these new tasks, smart devices are connected to the network and utilize different communication methods such as zig-bee and wireless, which leads to various vulnerabilities. For instance, the authors of *Neighborhood watch: security and privacy analysis of automatic meter reading systems* [19] showed that automatic meter reading (AMR) systems are vulnerable to spoofing attacks. They studied the wireless signals sent by AMR meters and found that no encryption mechanism was applied, so adversaries could decode the signal easily to get the real message information. Besides, adversaries can also act as meters to send fake messages to the readers, possibly to commit energy theft. Considering so many AMR meters had already been deployed in the US, the authors in [19] suggested a "Jammer add-on" solution, which required attaching an extra hardware device to every AMR meters to jam the original signal. Whenever a reader device came, it would temporally close the jammer device so that real data would be read. The authors claimed that this method did not

require updating the whole AMR meter system from either software or hardware. Yet a new problem came: the new jammer devices also needed to be tamper-resistant.

Another paper *Man-in-the-middle attack test-bed investigating cyber-security vulnerabilities in smart grid scada systems* [20] showed a man-in-the-middle attack test-bed to the supervisory control and data acquisition (SCADA) system of the smart grid. Today, the SCADA system's architecture is open standard and protocols and distributes functionalities across a wide area network (WAN), which allows more and more interoperability, connectivity, and compatibility. However, these new features make the systems more vulnerable to various unintentional or malicious cyber attacks. Introduced in *Man-in-the-middle attack test-bed investigating cyber-security vulnerabilities in smart grid scada systems* [20], the attacker can first launch an ARP spoofing attack to associate a malicious host's MAC address with the IP of a target host. Then he could perform a man-in-the-middle attack and gain access to confidential information.

2.2. SMART METER AUTHENTICATION SCHEMES

One way to address the aforementioned problems is to provide concrete and efficient authentication schemes for communications in the smart grid. Existing smart meter authentication schemes are reviewed next since smart meters are usually plotted at the users' houses and more vulnerable than other smart grid infrastructures. The computation ability of smart meters is also highly restricted. Therefore, the authentication scheme adopted needs to be significantly efficient.

A lightweight message authentication scheme for smart grid communications [26] talks about a situation in which Internet Protocol (IP)-based communication technologies are considered to set up smart grid communication networks. In this kind of smart grid communication network, a wide variety of malicious attacks that existed in IP-based schemes, such as replay, traffic analysis, and denial of service attacks, also need to be addressed in the smart grid. Considering the limited resources (i.e., low memory and computational capacity) of the smart meters, when facing some time-critical demand request

and keeping the quality of service, the smart grid needs an efficient authentication mechanism. Therefore, the authors proposed a lightweight message authentication scheme for securing communication amongst various smart meters at different points of the SG based on the Diffie-Hellman key establishment protocol and hash-based message authentication code.

Multicast authentication in the smart grid with one-time signature [27] proposed a multicast authentication scheme in the smart grid using a one-time signature. Compared with the existing multicast authentication scheme, HORS, their scheme reduced the storage overhead of the receivers by a factor of 8. Their scheme could also flexibly allocate the increased computations between the sender and receiver based on their computing resources. However, they assumed an adversary could eavesdrop some valid signatures to make a signature forgery attack and compare the possibility to link those signatures to the real message with their scheme, whereas the possibility of this link is really insignificant.

In *A privacy preserving and secure authentication protocol for the advanced metering infrastructure with non-repudiation service* [28], the authors proposed an ID-based authentication protocol for the advanced metering infrastructure. This protocol can provide source authentication, data integrity, and non-repudiation services while preserving the end-customer's privacy.

Considering the security issues happen during the data transmission between smart meters and utility servers, *Authentication and key management for advanced metering infrastructures utilizing physically unclonable functions* [29] proposed an approach based on PUF (physically unclonable function) technology for providing a hardware-based authentication of smart meters and an efficient key management scheme. Their major advantage by using PUFs was that there was no need to modify the existing smart meter communication system (no software or hardware upgrades were needed).

Multilayer consensus ecc-based password authenticated key-exchange (mcepak) protocol for smart grid system [30] considered a situation in which different layers were shown in the smart grid communication infrastructure: home area networks (HANs), building area networks (BANs), neighbor area networks (NANs), and SG central controllers (SGCC),

and each layer had separate controllers. Those controllers were connected from layer to layer, and each adjacent layers shared the same password for secure communication. HAN controllers were linked to smart appliances so that smart control from HAN controllers could be done directly. However, the control command from the upstream controllers of HAN had to go through different layers' re-encryptions which made it much too costly. To address this problem, the authors of *Multilayer consensus ecc-based password authenticated key-exchange (mcepak) protocol for smart grid system* [30] presented a multilayer consensus elliptic curve cryptography based password authenticated key exchange (MCEPAK) protocol that allowed each new smart meter to be authenticated and share a password with every upstream controller in an efficient way.

The authors of *A privacy-preserving smart metering scheme using linkable anonymous credential* [31] tried to build a scheme that could not only authenticate the messages sent from smart meters, but also preserve the user's privacy. They used Camenisch-Lysyanskaya (CL) signature to build a linkable anonymous credential protocol and proposed a privacy-preserving smart metering scheme based on the new linkable anonymous credential. Their protocol also had the property to trace the fault meters. However, it was not clear in their authentication scheme if meter credentials were used.

2.3. PRIVACY PRESERVATION

Even if messages in smart grid communication are authenticated and fully protected, there are potential privacy implications that arise from the collection and usage of smart grid data [32, 33]. As mentioned before in this chapter, fine-grained collections of power usage data can be used to identify most household appliances. It is clear that there is a strong need to develop privacy-preserving frameworks for various problems in the smart grid infrastructure. There have been considerable works along privacy-preserving smart grids from different aspects:

- Power Consumption Data Aggregation: A number of papers [34, 35, 36, 37, 38, 39, 40, 41, 42] utilized data aggregation techniques to hide the sensitive information.

Basically, the power consumption data of a group of customers were aggregated together so that even fine-grained data collection would not leak an individual customer's private information. However, the extent to which those data could be used for analysis to keep the quality of service (e.g., the state estimation decided from aggregated data may not be accurate enough) was not clear. At the same time, the aggregation technique could not totally overcome the privacy issues either.

- Dynamic Energy Management through Battery: Another big category of works involves using rechargeable battery algorithms [43, 44, 45, 46, 47, 48, 49] to mask the real household usage patterns. Typically, those batteries were combined with power controllers and deployed for each individual household. Their function was to make the meter load close to constant. The battery was deployed between the meter and the household, so the meter would record the energy charging the battery in each time-series. When household appliances needed more energy, the battery would discharge more; on the contrary, the battery will recharge. Yet, those methods usually need to have the prior knowledge of household activities in hand to make perfect masks. These methods are not safe if the adversaries already know the algorithms power controllers use.

- Power Usage Data Anonymization: Power usage data anonymization is another direction for preserving the user's privacy. For example, in *Smart grid privacy via anonymization of smart metering data* [50] an anonymization technique was studied. Smart meters were set with two IDs by a trusted third party (e.g., the manufacturers). One ID was known by the utility company, and the other ID was anonymous. The known ID was attached to low frequency data message packets that were used for billing purpose while the anonymous ID was attached to high frequency data message packets that were used for analysis. Even the utility company didn't know the relation between the two IDs. However, in *Smart metering depseudonymization* [51] the authors showed an attack model in which the knowledge of anonymous consumption traces and some recent activities of a household allowed attackers to

link the household's identity to its consumption trace so they could launch inference attacks.

- State Estimation Vector Obfuscation: In *Cooperative state estimation for preserving privacy of user behaviors in smart grid* [52], a privacy-preserving cooperative state estimation technique was investigated. Based on an unbiased linear estimation scheme [53], the authors developed a protocol that enabled each smart meter to add a distributed obfuscation vector to mask an individual's original measurements and, later, keep the result of estimation same. However, the authors didn't give a concrete security proof, so the privacy-preserving property might not be guaranteed.

Although many works had been posted to use privacy-preserving technologies in a smart grid system, they do not address the P-PUC problem in particular. The protocols presented in [13, 14] are the only existing work related to the proposed problem. The first two P-PUC protocols proposed in [13] were built based on two strategies. The protocols were efficient, whereas the total energy consumption was leaked to one of the households, and the maximum energy consumption among the households was also revealed. In addition, the secure division protocol utilized before was among several proposed protocols in [54], which were either not secure or not efficient.

In [14], another two P-PUC protocols were developed to address the security issues of the earlier P-PUC protocols. The protocols were more secure and practical. Secure sub-protocols were built to be used as basic generic secure primitives for other privacy-preserving problems.

3. BACKGROUND

This chapter include a brief review of a basic smart grid control system model. Some preliminaries that are highlighted include: the security definition of this work under the semi-honest model and the cryptography techniques adopted in this dissertation.

3.1. SMART GRID SYSTEM MODEL

Figure 3.1. gives a typical view of the smart grid control system. In this system, each household is equipped with a smart meter that links to every smart device in the household through a home area network (HAN). The smart meters deployed under the same neighborhood, combined with a substation/data-concentrator, compose a neighborhood area network (NAN). At the same time, substations connect to the utility company through the wide area network (WAN). For each level of network, there are different privilege levels of controllers that manage data flow and make smart controls. To assist with the work of the controllers, the smart meters not only read or deliver normal power consumption information in a very high frequency (e.g., 15-minute intervals), but also have some computation abilities to help higher level controllers (e.g., deployed in the utility company) to make smart controls. Here are the functionalities of the main components in a smart grid control system:

- Utility company: has the central controller, is in charge of analyzing data and billing. The utility company connects to multiple substations through the WAN. Sometimes the utility company would handle data analysis tasks in order to make smart controls for household appliances. They could also give these jobs to third party service providers. Alarms and alerts are also the responsibilities of utility company. It could send the highest level control signals during an emergency condition.

- Substation/data-concentrator: has a data collector that collects data from the smart meters in a neighborhood. The link between the substation/data-concentrator and the smart meter could be wired or wireless connections (e.g., in some existing AMR

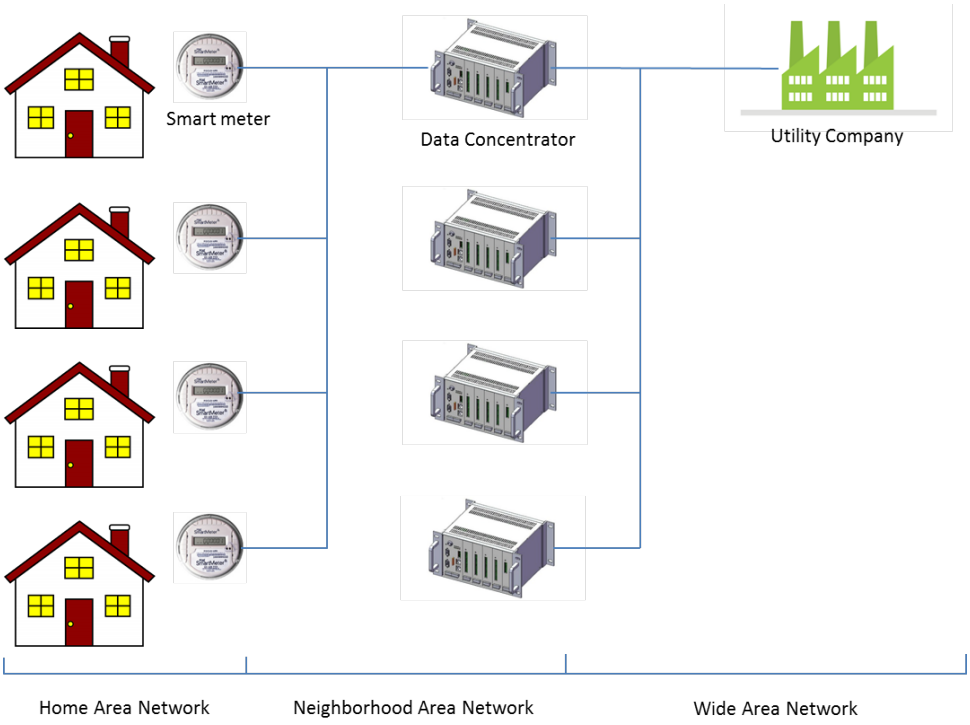


Figure 3.1. An Overview of Smart Grid Control System

systems, meter data could be gathered by car-carried readers remotely). After accumulation, the substation/data-concentrator transfers the processed data to the utility company. It could also route the messages between smart meters and the utility company.

- Smart meter: has a micro-controller and limited power, storage, and computation ability. Smart meters can engage in two-way communications with substations and household devices. They may also have the functionality to enable utility companies or customers to remotely connect or disconnect home appliances and services.

3.2. PRELIMINARIES

This section includes a discussion of the security definition of this dissertation. The tools of this study, additive homomorphic encryption that forms the core of P-PUC and Yao's garbled circuit that used to build the secure division protocol for OP-PUC are also

introduced. A definition and some notations that are used throughout this paper are presented at the end of this section.

3.2.1. Security Definition. The security definition adopted in this paper is from the field of secure multi-party computation (MPC). The notion of MPC was first introduced by Yao who also proposed a provably secure solution for the well-known Millionaires' problem (under two-party setting). Briefly, the Millionaires' problem involves two parties, holding their private wealths, who want to know which one is richer without revealing their actual wealth to the other party [55, 56]. This work was later extended to the multi-party case by Goldreich et al. [57], and it was shown that any computation that can be done in polynomial time by a single party can also be done securely by multiple parties. Since then, there have been many theoretical advancements as well as practical frameworks developed for the multi-party case [58, 59, 60, 61, 62, 63].

In this paper, it is assumed that the parties are semi-honest, often referred to as honest-but-curious, where each party follows the rules of the specified protocol but is free to later deduce any additional information by using the intermediate results the party sees during the execution of the protocol. Under the semi-honest model, whatever a party can infer from its private input and output is not considered as a privacy violation. More specifically, this paper adopts the semi-honest security definition from the field of MPC [64]. Briefly, the following definition captures the security definition under the semi-honest model.

Definition 1. *Let a_i be the input of party A_i, $VIEW_i(\pi)$ be A_i's execution image of the protocol π and δ_i be the output for A_i computed from π. Then, π is secure if $VIEW_i(\pi)$ can be simulated from $\langle a_i, \delta_i \rangle$ and distribution of the simulated image is computationally indistinguishable from $VIEW_i(\pi)$.*

In this definition, an execution image generally includes the input, the output, and the messages communicated during an execution of a protocol. To prove a protocol is secure under the semi-honest model, one generally needs to show that the execution image of a protocol does not leak any information regarding the private inputs of the

participating parties [64]. The reader may refer to *The Foundations of Cryptography* [64] for more detailed security definitions and their proofs.

3.2.2. Additive Homomorphic Probabilistic Encryption. This paper utilizes an additive homomorphic and probabilistic public key encryption system (denoted by HEnc), such as Paillier cryptosystem [65], for constructing our secure protocols. Let E_{pk} and D_{pr} be the encryption and decryption functions based on the HEnc system with public key pk and private key pr. It is then impossible for a computationally bounded adversary to decrypt any given ciphertext successfully in polynomial time without the private key pr. In addition, let N denote the group size or RSA modulus (which is usually of 1024 bits). For any two given plaintexts $m_1, m_2 \in \mathbb{Z}_N$, the HEnc system exhibits the following properties [66].

- Homomorphic Addition:

$$E_{pk}(m_1 + m_2) \leftarrow E_{pk}(m_1) * E_{pk}(m_2);$$

- Homomorphic Multiplication:

$$E_{pk}(m_1 * m_2) \leftarrow E(m_2)^{m_1};$$

- Semantic Security: The encryption scheme is semantically secure as defined in [67, 68]. Given a set of ciphertexts, an adversary cannot deduce any additional information about the plaintext. This further implies that the ciphertexts are statistically indistinguishable under chosen-plaintext attack (IND-CPA).

Any HEnc system can be used to implement the proposed protocols. Depending on the underlying HEnc system used, the homomorphic additions or multiplications are followed by modulo operations for security reasons. For example, in the Paillier cryptosystem [65], operations on ciphertexts are always followed by a modulo N^2 operation in order to ensure that the resulting ciphertext is still in $\mathbb{Z}_{N^2}$ and uniformly random. For presentation purposes, the modulo operations are simply omitted in the rest of this paper.

3.2.3. Yao's Garbled Circuit. In order to implement an efficient secure division protocol for OP-PUC in which all input values are promised to be encrypted, the garbled circuit approach introduced by Yao [69] is adopted. Yao's garbled circuit is a method Proposed by Andrew Yao in 1986 for Secure multi-party computation[69]. Basically, this protocol evaluate a function between two-party under semi-honest adversaries, and at the end of the protocol, the value of function is outputted without leaking any information to each party except the output value. In more detail, there is one party called *garbler* and another party called *evaluator*. At first, the *garbler* builds a 'garbled' version of circuit computing function. Then this function along with *garbler*'s input values (garbled corresponding to *garbler*'s real input values) are given to the *evaluator*. Upon receiving these, the *evaluator* obliviously obtains the garbled corresponding inputs of him by using 1-out-of-2 oblivious transfer protocol[70, 71, 72]. After this, the *evaluator* have all the inputs to calculate the function.

Recently, an intermediate language for describing and executing garbled circuits - the GCParser [73] 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 pipelining circuit generation and execution. We adopt this framework to build a garbled circuit for secure division.

3.2.4. Notations. Some common notations that are used extensively in this paper are shown in Table 3.1..

Definition 2. *Let $a_1, \ldots, a_n$ be the average power consumptions of n households in a given neighborhood. We define $a_{\max}$ as the maximum value out of the n power usages, i.e., $a_{\max} = \max(a_1, \ldots, a_n)$. In addition, for any given integer x such that $0 \leq x < l$ and $2^{m-1} \leq l < 2^m$, we define $[x]$ as a vector of encryptions of the individual bits of x. More formally, we define $[x]$ as follows:*

$$[x] = \langle E_{pk}(B_x[1]), \ldots, E_{pk}(B_x[m]) \rangle$$

Table 3.1. Common Notations

SMC	Secure Multi-Party Computation
P-PUC	Threshold-Based Power Usage Control
OP-PUC	Outsourceable P-PUC
HEnc	An additive homomorphic probabilistic encryption system
$\langle E_{pk}, D_{pr} \rangle$	A pair of encryption and decryption functions based on the HEnc system with $\langle pk, pr \rangle$ as the public-private key pair
C	The utility company
n	Number of households in a given neighborhood
A_i	i^{th} household in the neighborhood
S_1 and S_2	Two independent cloud servers
a_i	Average power consumption for household A_i
a_i' and a_i''	Secret shares of a_i between two parties
a	Sum of average power consumption in the neighborhood
a' and a''	Secret shares of a between two parties
t	Threshold value for the neighborhood
t' and t''	Secret shares of t between two parties
$A_{\max}$	User with the maximum average power consumption in the neighborhood
$a_{\max}$	Maximum average power consumption in the neighborhood
l	Domain size for a and t
m	Bit length of domain size l
B_x	Binary representation vector for integer x
δ_i	The lower bound of power user A_i has to cut off if $a > t$

where l is the domain size of x and m is the minimum number of bits required to represent l. B_x is the vector denoting the binary representation of x such that $B_x[1]$ and $B_x[m]$ are the most and least significant bits of x respectively.

For example, let us assume that $x = 5$ and $m = 4$. Then $[x = 5]$ is given by $\langle E_{pk}(0), E_{pk}(1), E_{pk}(0), E_{pk}(1) \rangle$. Also, if r is a random number chosen from group $\mathbb{Z}_N$, it is denoted by $r \in_R \mathbb{Z}_N$.

4. PRIVACY-PRESERVING POWER USAGE CONTROL

The first attempt at building P-PUC protocols, which are privacy-preserving power usage control protocols, is proposed in *Privacy-Preserving Power Usage Control in the Smart Grid* [13]. This chapter introduces the details of these protocols. A naive way to implement privacy-preserving power usage control is to utilize a trusted third party (TTP). As shown in Figure 4.1., each participating user A_i sends a_i to a TTP, and the utility company sends t to the TTP. Then, the TTP compares t with $a = \sum_{i=1}^{n} a_i$. If $t < a$, the TTP computes δ_i and sends it to A_i. Under this TTP-based P-PUC protocol, the privacy-preserving requirements can be easily achieved. However, such a TTP hardly exists in practice. Therefore, the goal of this chapter is to develop P-PUC protocols without utilizing a TTP so that the protocols achieve a similar degree of privacy protection as if there were a TTP.

The rest of the chapter is organized as follows: Section 4.1. proposes two P-PUC protocols suitable for different adjustment strategies, Section 4.2. discusses the security issues regarding the proposed protocols, and Section 4.3. gives some experimental results.

4.1. PRIVACY-PRESERVING POWER USAGE CONTROL PROTOCOLS

According to the two power usage adjustment strategies discussed in Section 1.1., two privacy-preserving TPUC protocols: P-PUC$_1$ and P-PUC$_2$ for strategy 1 and strategy 2, respectively, are proposed. The following assumptions must be made before introducing these two protocols:

- Following the previous notations: $A_1, \ldots, A_n$ denote n users in a participating neighborhood, and $a_1, \ldots, a_n$ denote the average power usage during a fixed time interval set by their utility company C.
- Let $a = \sum_{i=1}^{n} a_i$, and $a_{max} \in \{a_1, \ldots, a_n\}$ denotes the maximum individual energy usage of user $A_{max} \in \{A_1, \ldots, A_n\}$. Without loss of generality and for illustration purpose, it is assumed that a_{max} is unique and $a_1, \ldots, a_n$ are integer values. In

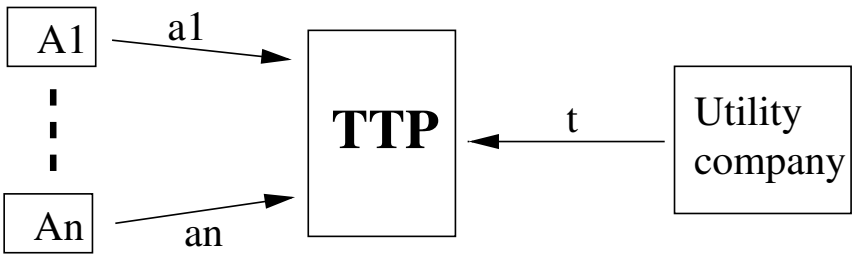


Figure 4.1. TTP-based P-PUC protocol

practice, $a_1, \ldots, a_n$ can be fraction numbers. Before using the proposed protocols, they need to be scaled up to become integers. At the end, the final results returned from the protocols are divided by the scaling factor to obtain the appropriate values.

- The ideal privacy-preserving requirements presented in Section 1.1. are difficult to achieve without using a trusted third party. In the proposed protocols, the privacy-preserving requirements are relaxed a little bit:
  - Under strategy 1: Only a and a_{max} can be disclosed to $A_1, \ldots, A_n$.
  - Under strategy 2: Only a can be disclosed to $A_1, \ldots, A_n$.

  The relaxed requirements allow more efficient protocols to be developed.

- One of the consumers (say A_1) plays an important role in performing some intermediate operations on handling some overall tasks of the neighborhood. Since every user in the neighborhood is equivalent, A_1 can also be chosen in an equal probability based on any leader election algorithm.

- It is assumed that the consumers and the utility company are semi-honest. That is, each party (either A_i or C) behaves as per the rules of the protocol; however, he/she is free to later deduce any additional information by utilizing the messages seen during the execution of the protocol [64]. This further implies that there is no collusion between the parties.

Both P-PUC$_1$ and P-PUC$_2$ protocols require several secure primitive protocols as subroutines, and these secure primitive protocols are defined as follows:

- Secure_Sum$(a_1, \ldots, a_n) \rightarrow a$

  There are n (at least three) parties participating in this protocol, and each party A_i has a_i as the input to the protocol. At the end of the protocol, summation a is known only to A_1.

- Secure_Max$(a_1, \ldots, a_n) \rightarrow a_{max}$

  There are n parties participating in this protocol, and each party A_i has a_i as the input to the protocol. At the end of the protocol, a_{max} is known to every participating party, but a_i is only known to A_i.

- Secure_Compare$(a, t) \rightarrow 1$ if $a > t$, and 0 otherwise

  There are 2 participants in this protocol. At the end, both parties know if $a > t$.

- Secure_Division$((x_1, y_1), (x_2, y_2)) \rightarrow \frac{x_1+x_2}{y_1+y_2}$

  There are 2 participants in this protocol. The private inputs (x_1, y_1) are from party 1, and (x_2, y_2) are the private inputs from party 2. At the end, both parties know $\frac{x_1+x_2}{y_1+y_2}$.

All these primitives have privacy-preserving properties in which the private input values are never disclosed to other participating parties. Next, let us discuss the implementations of these primitives based on existing research.

4.1.1. Implementations of Secure Primitives. There are several ways to implement the Secure_Sum protocol. In this chapter, a randomization approach is adopted, and the main steps of the protocol are given below (N indicates a very large integer):

1. A_1 randomly selects $r \in \{0, N-1\}$, computes $s_1 = a_1 + r \mod N$, and sends s_1 to A_2.

2. A_i, for $1 < i < n$, receives s_{i-1}, computes $s_i = s_{i-1} + a_i \mod N$, and sends s_i to A_{i+1}.

3. A_n receives s_{n-1}, computes $s_n = s_{n-1} + a_n \mod N$, and sends s_n to A_1.

4. A_1 receives s_n, computes $a = s_n - r \mod N$.

Because r is randomly chosen, s_1 is also a random value from A_2's perspective. Therefore, from s_1, A_2 is not able to discover a_1. Following the same reason, $a_1, \ldots, a_n$ are never disclosed to the other users during the computation process. Because A_1 is the only party who knows r, a can be derived correctly by A_1.

The steps proposed in [74] are adopted to implement the Secure_Max protocol. For Secure_Compare, we can use the generic solution given in [60]. Different solutions from [54] and [75] are considered to implement Secure_Division. An efficiency and security trade-off analysis between different Secure_Division sub-protocols are proposed in Section 4.3..

4.1.2. The P-PUC$_1$ Protocol. Once all the primitives described before are obtained, the P-PUC$_1$ protocol can be easily implemented. Key steps are given below:

1. A_1 obtains $a \leftarrow \text{Secure\_Sum}(a_1, \ldots, a_n)$
2. A_1 and the utility company jointly perform the Secure_Compare protocol.
   If $\text{Secure\_Compare}(a, t) = 1$, then
   (a) Each A_i obtains $a_{max} \leftarrow \text{Secure\_Max}(a_1, \ldots, a_n)$
   (b) A_{max} (self-identified via a_{max}) reduces his or her energy consumption
3. Repeat these steps until $\text{Secure\_Compare}(a, t) = 0$

Since A_1 has the value a, the Secure_Compare protocol at step 2 can only be executed between A_1 and the utility company. However, any user can become A_1, and this can be achieved through a leader election process among the users to determine who wants to be A_1. Alternatively, A_1 could be randomly chosen before each execution of the protocol.

4.1.3. The P-PUC$_2$ Protocol. In this protocol, A_1 is also responsible for the secure summation and secure comparison operations. An additive homomorphic probabilistic public key encryption (HEnc) system is used as the building block in the proposed protocol. The private key is only known to the utility company, and the public key is known to all the participating users. Let E_{pk} and D_{pr} be the encryption and decryption functions in an HEnc system with public key pk and private key pr. Without pr, no one

can discover x from $E_{pk}(x)$ in polynomial time. When the context is clear, the pk and pr subscripts in E_{pk} and D_{pr} are omitted. The HEnc system has the following properties:

- The encryption function is additive homomorphic: $E_{pk}(x_1) \times E_{pk}(x_2) = E_{pk}(x_1 + x_2)$;
- Given a constant c and $E_{pk}(x)$, $E_{pk}(x)^c = E_{pk}(c \cdot x)$;
- The encryption function has semantic security as defined in *The knowledge complexity of interactive proof systems* [67], i.e., a set of ciphertexts do not provide additional information about the plaintext to an adversary or $E_{pk}(x) \neq E_{pk}(x)$ with a very high probability.
- The domain and range of the encryption system are suitable.

Any HEnc system is applicable, but in this paper Paillier's public-key homomorphic encryption system [65] is adopted due to its efficiency. Informally speaking, the public key in the system is (g, N), where N is resulted from multiplication of two large prime numbers and $g \in \mathbb{Z}^*_{N^2}$ is randomly chosen.

To implement the P-PUC$_2$ protocol according to Equation 3, each user A_i needs to calculate $\frac{a_i \cdot t}{a}$ between A_i and the utility company C so that a_i is not disclosed to C and t is not disclosed to A_i. The Secure_Division primitive and an HEnc system will be adopted to solve this problem.

$$\delta_i = \frac{a_i}{a}(a - t) = a_i - \frac{a_i \cdot t}{a} \tag{3}$$

Assume that $E(t)$ is broadcasted by the utility company initially. The main steps of the P-PUC$_2$ protocol are given below:

1. A_1 obtains $a \leftarrow \text{Secure\_Sum}(a_1, \ldots, a_n)$
2. A_1 and the utility company C jointly perform the Secure_Compare protocol.
   If Secure_Compare$(a, t) = 1$, then
   (a) A_1 randomly selects r from $\{0, N - 1\}$
   - Set $y_1 = N - r$ and $y_2 = a + r \mod N$

- Send y_1 to $A_2, \ldots, A_n$ and y_2 to C

(b) Each A_i $(2 \leq i \leq n)$ randomly selects r_i from $\{0, N-1\}$

- Compute $E(t)^{a_i}$ to get $E(a_i \cdot t)$
- Set $x_{1i} = N - r_i$ and $s_i = E(a_i \cdot t) \times E(r_i) = E(a_i \cdot t + r_i)$
- Send s_i to C

(c) The utility company C sets $x_{2i} = D(s_i)$ for $2 \leq i \leq n$

(d) For $2 \leq i \leq n$, A_i with input (x_{1i}, y_1) and C with input (x_{2i}, y_2) jointly perform the Secure_Division protocol

- A_i obtains $\kappa_i = \text{Secure\_Division}((x_{1i}, y_1), (x_{2i}, y_2))$
- A_i sets $\delta_i = a_i - \kappa_i$
- A_i reduces his or her power consumption according to δ_i

A_1 in the P-PUC$_2$ protocol is a designated user in the participating neighborhood who is responsible for computing a and distributing $N - r$ to the other users and $a + r \mod N$ to the utility company. Note that a computed at step 1 should not include the value a_1, (This can easily be achieved through a small modification to the Secure_Sum protocol) and A_1 does not adjust his or her energy consumption. This prevents the disclosure of t to A_1. For instance, if A_1 obtains a δ_1, A_1 can derive t based on Equation 3. To be fair, A_1 can be randomly selected among the participating users before each execution of the protocol.

The purpose of step 2(a) is to hide a from the utility company and the other users. Since r is randomly chosen, y_1 and y_2 are randomly distributed in $\{0, N-1\}$. As a result, the other users $A_2, \ldots, A_n$ cannot discover a from y_1, and similarly, the utility company cannot discover a from y_2. The goal of step 2(b) is to hide a_i from the utility company and t from A_i. Since the encryption scheme is semantically secure, from $E(t)$ and without the private key, the users cannot learn anything about t. In addition, because r_i is randomly chosen, the x_{2i} value computed at step 2(c) does not reveal anything regarding a_i. The operations performed at steps 2(b) and 2(c) are based on the aforementioned

additive homomorphic property of the encryption function E. Since $x_{1i} + x_{2i} = a_i \cdot t$ and $y_1 + y_2 = a$, $\kappa_i = \frac{a_i \cdot t}{a}$, the protocol correctly returns δ_i for each A_i, except for A_1.

4.2. SECURITY ANALYSIS

Regarding the P-PUC$_1$ protocol, a is disclosed to A_1 and a_{max} is disclosed to all the participating users. Since a is aggregated information, the disclosure of a can hardly cause any privacy violations. Although a_{max} is disclosed, no one can link a_{max} to a particular user. Thus, the disclosure risk of the P-PUC$_1$ protocol is not significant.

The P-PUC$_2$ protocol only discloses a to A_1, so it is more secure compared to the P-PUC$_1$ protocol. However, because the Secure_Division protocol needs to be executed between every user and the utility company, the protocol is less efficient than P-PUC$_1$. Therefore, depending on whether security or efficiency is more important, both proposed protocols are applicable in practice.

4.3. EXPERIMENTAL RESULTS

The protocols were implemented in C using the Paillier encryption scheme [65]. All experiments were conducted on a Linux machine running Ubuntu 10.04 LTS with Intel® Xeon® Six-Core™ 3.07GHz and 12GB RAM.

For P-PUC$_1$, the main components are Secure_Sum, Secure_Compare, and Secure_Max. Since the Secure_Sum protocol only performs additions, and the size of the input from each party is 1, the protocol is very efficient. For Secure_Max, a probabilistic scheme [74] is adopted in which multiple rounds of communications and computations are involved. The computation complexity of this part is very neglectable since there are only several simple operations for each party. On the other hand, one must consider the communication. Since the result accuracy of this protocol increases steadily with the number of rounds performed and introduced in [74], accuracy is nearly 100% when 5 rounds of computation are performed. In this computation, 5 rounds are adopted. As the input size for each party in every round is still 1, this protocol is also acceptable. Considering the

Secure_Compare protocol, the total computation time for $m = 10$ is 1.12 seconds. Even for the larger $m = 50$, the computation time is still less than 6 seconds. Figure 4.2. gives the overall computation time of P-PUC$_1$ for every involved party in one iteration. The use of A_i denotes the average cost for A_i. As shown in Figure 4.2., the computation costs of A_1, C, and A_i are 0.36, 0.76, and <0.01 second, respectively, for $m = 10$. The computation times of A_1 and C are linear, expanding with the bit length of domain size increasing as expected, whereas the portion of computations of each individual A_i is small and constant. This is because individual A_i only makes simple computations during Secure_Sum and Secure_Max, while the most time consuming part, Secure_Compare, is executed by A_1 and C. This approach is really suitable for the condition in which the smart meter has a very limited processing ability compared to other devices in the network. And in such condition, the work of A_1 could also be given to a data concentrator deployed in the neighborhood area network that has a better computation ability.

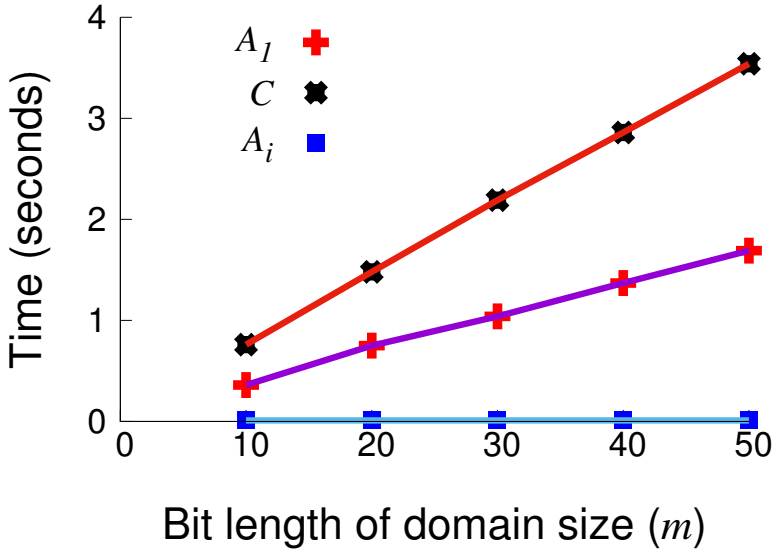


Figure 4.2. Complexity of P-PUC$_1$ for $n = 50$

The P-PUC$_2$ protocol is composed of Secure_Sum, Secure_Compare, and Secure _Division. The Secure_Sum and Secure_Compare protocols are studied in the last paragraph. For Secure_Division, there is a trade-off between security and efficiency, as shown in Table 4.1.. Two division protocols are adopted from [54]. The protocol using Oblivious Transfer takes 2.15 seconds for 50 divisions (compare to a neighborhood size of $n = 50$),

Table 4.1. A Comparison of Secure_Division Protocols

Secure_Division Protocol	Fully Secure	Time (for $n = 50$)
Division using Oblivious Transfer [54]	No	21.5 seconds
Division using Homomorphic Encryption [54]	No	6.92 seconds
Division using Bit Representation [75]	Yes	more than 50 hours

and the Homomorphic Encryption based protocol takes 6.92 seconds for the same data size. However, those protocols are not fully secure. A more secure protocol [75] may take more than an hour for 1 division. A Secure_Division protocol that is both fully secure and efficient is needed for P-PUC$_2$.

5. PRIVACY-PRESERVING POWER USAGE CONTROL WITH MINIMUM INFORMATION DISCLOSURE

The solutions shown in Chapter 4. revealed intermediate results such as the average total power usage a and the maximum power usage among a_i's to consumers and/or C. In addition, the primitives that were used as building blocks in Chapter 4. are either insecure or inefficient. Therefore, in *Secure and threshold-based power usage control in smart grid environments* [14], two new P-PUC protocols based on the same two strategies are introduced. This chapter gives the details of these novel protocols. The main contributions of this chapter can be summarized as follows:

- Security - The proposed protocols provide better security than the protocols in Chapter 4. in terms of protecting the individual parties' private inputs as well as the intermediate results. More specifically, the protocols' security is followed from the standard semi-honest security definition in the field of secure multi-party computation (MPC) [55, 56, 64].

- Efficiency - Efficient sub-protocols are developed for performing binary conversion, comparison, maximum, and division operations in a privacy-preserving manner. Note that the existing protocols in Chapter 4. utilize an inefficient generic circuit method in order to perform the basic secure comparison operation. In addition, the existing Secure_Division protocol [75] required in P-PUC$_2$ is very costly, which leads to the overall inefficiency.

- Generality - The newly proposed set of sub-protocols acts as a generic solution; therefore, the protocols can also be utilized in many other MPC applications, such as secure electronic voting and private auctioning.

With respect to user utility, the utility company is not assumed to automatically cut off the power supply for a user. It is assumed that users are responsible, and when they receive an indication of using too much electricity, they can turn off one or more appliances that are not crucial for their well-being. The threshold t is determined by the utility company

based on its historical data, and t is generally larger than the total energy usage for a given neighborhood. Thus, when the neighborhood's power usage exceeds the predefined threshold t, some households are using more energy than they normally need. Cutting off or reducing the power usage of these households will likely not affect their daily activities. As a result, the negative impact of reducing power consumption of these households is minimal. Plus, as a positive impact, the proposed protocols will help these households to effectively manage their power consumption to save on utility costs.

Since the main goal of this chapter is to develop secure P-PUC protocols, the basic characteristics of the smart grid were not taken into consideration while developing the protocols. Nevertheless, due to expensive encryption costs in the proposed protocols, the communication delays (which are usually in milliseconds) and the other basic smart grid factors will not affect the computation costs much. Hence, the proposed protocols can easily be deployed on top of the smart grids without incurring significant overhead costs due to the underlying smart grid models or factors.

The rest of this chapter is organized as follows. The proposed P-PUC protocols along with the newly proposed set of sub-protocols are discussed in detail in Section 5.1.. Section 5.2. makes some security analysis, and Section 5.3. presents various experimental results based on the proposed protocols under different parameter settings and demonstrates their practical value.

5.1. PRIVACY-PRESERVING POWER USAGE CONTROL WITH MINIMUM INFORMATION DISCLOSURE PROTOCOLS

This section presents the proposed P-PUC protocols, which are based on the two strategies mentioned in Section 1.1.. The first protocol (following from Strategy 1) is an iterative approach and is based on reducing the power usage for the household with the maximum power consumption during the last time interval. Whereas, the second protocol (following from Strategy 2) involves the computation of power reduction by each consumer, independently. Most assumptions from the previous chapter are followed, while some of them are updated:

- A_1 is still playing as a coordinator. However, at this time, it is assumed that A_1 holds a public-private key pair (pk_{A_1}, pr_{A_1}) based on the HEnc system mentioned in Section 3.2.. Similarly, the utility company C holds (pk_C, pr_C). Note that the private keys pr_{A_1} and pr_C are kept as secret by A_1 and C, respectively.

  Under the above assumption, A_1 has to play a bigger role than other consumers in the proposed protocols. Since it is not possible to perform arbitrary computations over encrypted data by C alone, the proposed protocols require a second party (A_1 in this case) to evaluate the desired functionalities in a privacy-preserving manner (two-party secure computations). It is worth pointing out that the role of the second party can be played by any one of the consumers. This further implies that the second party can vary in each protocol execution. In general, before the execution of each protocol, all the consumers can have an agreement and decide who will be playing the role of the second party. A simple and straightforward solution is to choose the second party in a ring topology. That is, A_1 in the 1st execution, A_2 in the 2nd execution, and so on. For simplicity, A_1 is treated as the second party in the rest of this chapter.

- Suppose that $1 \leq a, t \leq l$, where l is the maximum domain size for the values of a and t such that $2^{m-1} \leq l < 2^m$. At least m bits are required to represent l. This is a practical assumption due to the following reason. Because the values of a and t are usually small in this problem domain, the value of l can be chosen appropriately to satisfy the condition. Also, l and m are assumed to be public. However, t remains private to C, and a should not be revealed to consumers or C.

- Again, the consumers and the utility company are semi-honest, and all the devices in the smart grid are tamper-resistant. That is, there will be no collusion between individual users and the utility company and no modification of data during the communications. However, parties can make inference of anything from what they have.

In the rest of this chapter, (pk, pr) are used as the public-private key pair to avoid cluttering the presentation (in this chapter, depending on the context, (pk, pr) can be either (pk_{A_1}, pr_{A_1}) or (pk_C, pr_C)). Since the existing primitives used in Chapter 4. are either insecure or inefficient, new solutions are developed to secure binary conversion, comparison, maximum, and division problems (more details are provided later in this section). More specifically, the proposed protocols utilize the following security primitives as the building blocks.

- Sum_Random_Shares$(a_1, \ldots, a_n) \rightarrow (s_1, s_2)$:

  Each consumer, with his/her private input a_i, participates in this protocol. At the end of this protocol, A_1 gets a random share s_1 and C gets a random share s_2 such that $s_1 + s_2 \bmod N = a$. During this process, no other information is revealed to the consumers or C.

- Secure_Binary_Conv$(E_{pk}(a)) \rightarrow [a]$:

  C with input $E_{pk}(a)$ and A_1 with private key compute the encryptions of the individual bits of binary representation of a. At the end, the output $[a]$ is known only to C.

- Secure_Comp$([a], t) \rightarrow h$

  C with input $([a], t)$ and A_1 with private key securely check whether $a > t$ or not. At the end of this protocol, only A_1 knows the output h, where $h = 1$ if $a > t$, and $h = 0$ otherwise.

- Secure_2P_Max$([x], y) \rightarrow [\max(x, y)]$

  Consumer A_i with input $([x], y)$ and C with private key compute the encryptions of the individual bits of maximum between x and y. During this process, x is not revealed to A_i and C. In addition, y is not revealed to C. The output $[\max(x, y)]$ is known only to A_i.

- Secure_2P_Enc_Max$([x], [y]) \rightarrow [\max(x, y)]$

  One of the consumers, say A_i, holding $([x], [y])$, and C, who has the private key,

are jointly involved in this protocol, where both x and y are regarded as private information. That is, (x, y) should not be revealed to A_i and C. The output of this protocol is the encryptions of the individual bits of maximum between x and y (i.e., $[\max(x, y)]$). At the end, $[\max(x, y)]$ is revealed only to A_i.

- Secure_MP_Max$(a_1, \ldots, a_n) \rightarrow [a_{\max}]$

  The n consumers (with their respective private inputs a_i) and the utility company C with private key are involved in this protocol. The output of this protocol is the encryptions of the individual bits of $\max(a_1, \ldots, a_n) = a_{\max}$. The output $[a_{\max}]$ is revealed to all consumers whereas no information is revealed to C.

- Secure_Div$(E_{pk}(a), t) \rightarrow \frac{t}{a}$

  C, with input $(E_{pk}(a), t)$, and A_1, with private key, securely compute the ratio of $\frac{t}{a}$. During this process, neither a nor t is revealed to A_1. In addition, a is not revealed to C. At the end of Secure_Div, the output $\frac{t}{a}$ is known to all consumers.

5.1.1. The P-PUC$_1^*$ Protocol. The first protocol, denoted as P-PUC$_1^*$, is based on Strategy 1 as discussed in Section 1.1.. The overall steps involved in P-PUC$_1^*$ are highlighted in Algorithm 7. Before going into the details of P-PUC$_1^*$, and to make the presentation more clear, the new solutions to various secure primitives, which are utilized as building blocks in P-PUC$_1^*$, are presented. A discussion of how they are combined together in constructing the P-PUC$_1^*$ protocol is then presented.

1) Sum_Random_Shares:

The main steps involved in the Sum_Random_Shares protocol are given in Algorithm 1. Initially, A_1 chooses a random number r from $\mathbb{Z}_N$ (note that here N is part of A_1's public key) and sets his/her random share as $s_1 = N - r$. Then, A_1 randomizes his/her private input by computing $z_1 = a_1 + r \bmod N$ and sends it to A_2. For $2 \leq i \leq n$, each consumer A_i, upon receiving the randomized partially aggregated value z_{i-1} from A_{i-1}, adds his/her private input by computing $z_i = z_{i-1} + a_i \bmod N$, and sends z_i to A_{i+1}. However, the last consumer A_n sends z_n to C. Finally, C sets his/her random share s_2 to z_n. Observe that $s_1 + s_2 \bmod N = a$ always holds.

Algorithm 1 Sum_Random_Shares$(a_1, \ldots, a_n) \rightarrow (s_1, s_2)$

Require: a_i is private to A_i, for $1 \leq i \leq n$; $(pk, pr) = (pk_{A_1}, pr_{A_1})$, where pr is known only to A_1. The consumers use a ring topology to communicate with each other.

1: A_1:

(a). Pick a random number $r \in \mathbb{Z}_N$

(b). $s_1 \leftarrow N - r$

(c). $z_1 \leftarrow a_1 + r \bmod N$; send z_1 to A_2

2: A_i, **for** $2 \leq i \leq n$ **do**:

(a). Receive z_{i-1} from A_{i-1}

(b). $z_i = z_{i-1} + a_i \bmod N$

(c). **if** $i = n$ **then**

- Send z_i to C

else

- Send z_i to A_{i+1}

3: C:

(a). Receive z_n from A_n

(b). $s_2 \leftarrow z_n$

2) Secure_Binary_Conv:

It is assumed that C has $E_{pk}(a)$, where a is not known to either consumers or C, and A_1 holds the private key (i.e., $(pk, pr) = (pk_{A_1}, pr_{A_1})$). The goal of the secure binary conversion protocol is to compute the encryptions of the individual bits of binary representation of a. That is, the output is $[a] = \langle E_{pk}(B_a[1]), \ldots, E_{pk}(B_a[m])\rangle$, where B_a denotes the binary representation vector of integer a with $B_a[1]$ and $B_a[m]$ as the most and least significant bits of a, respectively. Here m denotes the minimum number of bits required to represent the domain size l. Note that $1 \leq a \leq l$ and $2^{m-1} \leq l < 2^m$. At the end, the output $[a]$ is known only to C. The overall steps involved in the Secure_Binary_Conv protocol are summarized in Algorithm 2.

To start with, C initially computes $E_{pk}(a - i) = E_{pk}(a) * E_{pk}(N - i)$, for $1 \leq i \leq l$. Note that $N - i$ denotes "$-i$" under domain $\mathbb{Z}_N$. Then, C randomizes them by performing homomorphic multiplications as $P[i] = E_{pk}(a - i)^{\bar{r}_i}$, where $\bar{r}_i$ is a random number in $\mathbb{Z}_N$, for $1 \leq i \leq l$. Observe that $P[i] = E_{pk}(0)$ iff $i = a$. After this, C permutes the encrypted

vector P using a random permutation function π and sends the resulting permuted vector P' to A. Upon receiving P', A decrypts it component-wise to get $\tau[i] = D_{pr}(P'[i])$, for $1 \leq i \leq l$. Then, A generates a new vector u depending on vector τ as follows:

$$u[i] = \begin{cases} 1 & \text{if } \tau[i] = 0 \\ 0 & \text{otherwise} \end{cases}$$

It is emphasized that $\tau[i] = 0$ happens exactly once since $E_{pk}(a - i) = E_{pk}(0)$ occurs exactly once, for $1 \leq i \leq l$. A encrypts u component-wise using his/her public key to get $U[i] = E_{pk}(u[i])$, for $1 \leq i \leq l$, and sends the encrypted vector U to C. Upon receiving U, C performs inverse permutation on U to get $V = \pi^{-1}(U)$. Finally, C computes $[a]$ locally by performing the following homomorphic operations.

- Compute encrypted matrix S, where $S[i][j] = V[i]^{B_i[j]}$, for $1 \leq i \leq l$ and $1 \leq j \leq m$. Here, B_i denotes the binary representation vector of integer i with $B_i[1]$ and $B_i[m]$ as the most and least significant bits of i, respectively.
- Compute the encryption of j^{th} bit of a as $E_{pk}(B_a[j]) = \prod_{i=1}^{l} S[i][j]$, for $1 \leq j \leq m$.

3) Secure_Comp:

C, with private input $([a], t)$, and A_1, with private key pr_{A_1}, are jointly involved in the secure comparison protocol to decide whether $a > t$ or not. The output h is revealed only to A_1, where $h = 1$ if $a > t$ and 0 otherwise. The main steps involved in the Secure_Comp protocol are highlighted in Algorithm 3.

To start with, C computes the encrypted bit-wise XOR between the bits of a and t as $T[j] = E_{pk}(B_a[j] \oplus B_t[j])$ for $1 \leq j \leq m$ using the following formulation (in general, for any two bits x and y, the property $x \oplus y = x + y - 2xy$ always holds):

$$T[j] = E_{pk}(B_a[j] \oplus B_t[j]) = E_{pk}(B_a[j]) * E_{pk}(B_t[j]) * E_{pk}(B_a[j] * B_t[j])^{N-2}$$

However, since t is known to C, if $B_t[j] = 0$, then $T[j] = E_{pk}(B_a[j])$. Whereas, if $B_t[j] = 1$, then $T[j] = E_{pk}(1 - B_a[j])$. In general, C can easily compute $E_{pk}(B_a[j] * B_t[j])$ by

Algorithm 2 Secure_Binary_Conv($E_{pk}(a)$) $\rightarrow [a]$

Require: C has $E_{pk}(a)$, where $1 \leq a \leq l$ and $2^{m-1} \leq l < 2^m$; (Note: The private key $pr = pr_{A_1}$ is known only to A_1)

1: C:

(a). **for** $i = 1$ to l **do**:

- $E_{pk}(a - i) \leftarrow E_{pk}(a) * E_{pk}(N - i)$
- $P[i] \leftarrow E_{pk}(a - i)^{\bar{r}_i}$, where $\bar{r}_i \in_R \mathbb{Z}_N$

(b). $P' \leftarrow \pi(P)$; send P' to A_1

2: A_1:

(a). Receive P' from C

(b). **for** $i = 1$ to l **do**:

- $\tau[i] \leftarrow D_{pr}(P'[i])$
- **if** $\tau[i] = 0$ **then**:
  - $u[i] \leftarrow 1$

  else
  - $u[i] \leftarrow 0$

(c). $U[i] \leftarrow E_{pk}(u[i])$, for $1 \leq i \leq l$; send U to C

3: C:

(a). Receive U from A_1

(b). $V \leftarrow \pi^{-1}(U)$

(c). **for** $i = 1$ to l **do**:

- $S[i][j] \leftarrow V[i]^{B_i[j]}$, where B_i denotes the binary representation of i and $B_i[j]$ denotes the j^{th} component of vector B_i, for $1 \leq j \leq m$

(d). **for** $j = 1$ to m **do**:

- $E_{pk}(B_a[j]) \leftarrow \prod_{i=1}^{l} S[i][j]$

performing homomorphic multiplication as $E_{pk}(B_a[j])^{B_t[j]}$. Observe that $T[j] = E_{pk}(1)$ only if exactly one of the bits, i.e., either $B_a[j]$ or $B_t[j]$, is 1. After this, C performs the following homomorphic addition and multiplication operations for $1 \leq j \leq m$:

- Compute an encrypted vector W such that $W[j] = E_{pk}(B_a[j])^{1-B_t[j]} = E_{pk}(B_a[j] * (1-B_t[j]))$. It is observed that $W[j] = E_{pk}(1)$ only if $B_a[j] > B_t[j]$, otherwise $W[j] = E_{pk}(0)$. In particular, W stores $E_{pk}(1)$ at those locations where the corresponding bit of a is greater than that of t and $E_{pk}(0)$ otherwise.

Algorithm 3 Secure_Comp$([a], t) \rightarrow h$

Require: C has $[a]$ and t, where $1 \leq a, t \leq l$; (Note: The private key $pr = pr_{A_1}$ is known only to A_1)

1: C:

(a). **for** $j = 1$ to m **do**:

- $T[j] \leftarrow E_{pk}(B_a[j] \oplus B_t[j])$
- $W[j] \leftarrow E_{pk}(B_a[j])^{1-B_t[j]}$
- $X[j] \leftarrow X[j-1]^{r_j} * T[j]$, where $r_j \in_R \mathbb{Z}_N$ and $X[0] = E_{pk}(0)$
- $\Phi[j] \leftarrow E_{pk}(-1) * X[j]$
- $Y[j] \leftarrow W[j] * \Phi[j]^{r'_j}$, where $r'_j \in_R \mathbb{Z}_N$

(b). $Y' \leftarrow \pi(Y)$; send Y' to A_1

2: A_1:

(a). Receive Y' from C

(b). $M[j] \leftarrow D_{pr}(Y'[j])$, for $1 \leq j \leq m$

(c). **if** $\exists\ k$ such that $M[k] = 1$ **then**:

- $h = 1$ (denoting $a > t$)

else

- $h = 0$ (denoting $a \leq t$)

- Compute an encrypted vector X by preserving the first occurrence of $E_{pk}(1)$ (if one exists) in T by initializing $X[0] = E_{pk}(0)$. The rest of the entries of X are computed as $X[j] = X[j-1]^{r_j} * T[j]$. At most, one of the entries in X is $E_{pk}(1)$, and the remaining entries are encryptions of either 0 or a random number. In addition, if $\exists\ k$ such that $X[k] = E_{pk}(1)$, then index k is the first position (starting from the most significant bit) at which corresponding bits of a and t differ.

- After this, C computes $\Phi[j] = E_{pk}(-1) * X[j]$. From the above discussions, it is clear that $\Phi[j] = E_{pk}(0)$ once at most since $X[j]$ is equal to $E_{pk}(1)$ once at most. Also, if $\Phi[k] = E_{pk}(0)$, then index k is the first position at which bits of a and t differ.

- Then, C computes an encrypted vector Y by combining W and Φ. (Note that $W[j]$ stores the result of $B_a[j] > B_t[j]$ functionality.) Precisely, C computes $Y[j] = W[j] * \Phi[j]^{r'_j}$, where r'_j is a random number in $\mathbb{Z}_N$. The observation here is if $\exists$ an index k such that $\Phi[k] = E_{pk}(0)$ denoting the first flip in the corresponding bits of

a and t, then $W[k]$ stores the corresponding information, i.e., whether $B_a[k] > B_t[k]$ or not.

Finally C permutes Y, and sends the resulting permuted vector Y' to A_1. Upon receiving Y', A_1 decrypts it component-wise to get $M[j] = D_{pr}(Y'[j])$, for $1 \leq j \leq m$. Now, depending on the existence of index k, A_1 proceeds as follows. If there exists an index k in M such that $M[k] = 1$, then $B_a[k] > B_t[k]$ and also $B_a[i] = B_t[i] = 0$, for $1 \leq i \leq k-1$. In this case, $a > t$; therefore, A_1 sets the output h to 1. On the other hand, if $M[k] = 0$, then $a < t$. However, if there exists no such index k, then $t = a$. In either of the last two cases, A sets h to 0 (denoting $a \leq t$).

4) Secure_2P_Max:

In this protocol, consumer A_i, with the private input $([x], y)$, and C, with private key $pr = pr_C$, securely compute the encryptions of the individual bits of $\max(x, y)$, i.e., the output is $[\max(x, y)]$. At the end, the output $[\max(x, y)]$ is known only to A_i.

The basic idea of the proposed Secure_2P_Max protocol is for A_i to randomly choose the functionality f (by flipping a coin), where f is either $x > y$ or $y > x$, and to obliviously execute f with C using similar steps in Secure_Comp protocol. Since f is randomly chosen and known only to A_i, the output of functionality f is oblivious to C. After this, depending on f, A_i securely computes $[\max(x, y)]$ using homomorphic properties.

The overall steps involved in the Secure_2P_Max protocol are shown in Algorithm 4. To start with, A_i initially chooses the functionality f as either $x > y$ or $x < y$, randomly. Then, depending on f, A proceeds as follows, for $1 \leq j \leq m$:

- If $f : x > y$, A computes

$$\begin{aligned} W[j] &= E_{pk}(B_x[j]) * E_{pk}(B_x[j] * B_y[j])^{N-1} = E_{pk}(B_x[j] * (1 - B_y[j])) \\ \Gamma[j] &= E_{pk}(B_x[j] - B_y[j]) * E_{pk}(\hat{r}_j) = E_{pk}(B_x[j] - B_y[j] + \hat{r}_j) \end{aligned}$$

- Otherwise,

$$\begin{aligned} W[j] &= E_{pk}(B_y[j]) * E_{pk}(B_x[j] * B_y[j])^{N-1} = E_{pk}(B_y[j] * (1 - B_x[j])) \\ \Gamma[j] &= E_{pk}(B_y[j] - B_x[j]) * E_{pk}(\hat{r}_j) = E_{pk}(B_y[j] - B_x[j] + \hat{r}_j) \end{aligned}$$

- Observe that if $f : x > y$, then $W[j] = E_{pk}(1)$ only if $B_x[j] > B_y[j]$, and $W[j] = E_{pk}(0)$ otherwise. Similarly, if $f : y > x$, then $W[j] = E_{pk}(1)$ only if $B_y[j] > B_x[j]$, and $W[j] = E_{pk}(0)$ otherwise.

Then, A_i computes the encrypted vector Y based on the similar steps as mentioned in Step 1(a) of Secure_Comp (i.e., Algorithm 3). Secure_Comp protocol is based on the public-private key pair of A_1 whereas the Secure_2P_Max protocol utilizes the public-private key pair of the utility company C.

After this, A_i permutes the encrypted vectors Γ and Y using two random permutation functions π_1 and π_2 (known only to A_i). Specifically, A_i computes $\Gamma' = \pi_1(\Gamma)$ and $Y' = \pi_2(Y)$, and sends them to C. Upon receiving, C decrypts Y' component-wise to get $M[j] = D_{pr}(Y'[j])$, for $1 \leq j \leq m$, and checks for index k as mentioned in the Secure_Comp protocol. That is, if $M[k] = 1$, then C sets h to 1. Otherwise, C sets it to 0. In addition, C computes a new encrypted vector M' such that $M'[j] = \Gamma'[j]^h$, for $1 \leq j \leq m$, and sends both M' and $E_{pk}(h)$ to A_i. After receiving M' and $E_{pk}(h)$, A_i computes the inverse permutation of M' as $\widetilde{M} = \pi_1^{-1}(M')$. Then, A_i performs the following homomorphic operations to compute the encryption of j^{th} bit of $\max(x, y)$ (i.e., $E_{pk}(B_{\max(x,y)}[j])$), for $1 \leq j \leq m$:

- Remove the randomness from $\widetilde{M}[j]$ by computing

$$\lambda[j] = \widetilde{M}[j] * E_{pk}(h)^{N-\hat{r}_j}$$

- If $f : x > y$, then compute the j^{th} encrypted bit as $E_{pk}(B_{\max(x,y)}[j])) = E_{pk}(B_y[j]) * \lambda[j]$. Otherwise, compute $E_{pk}(B_{\max(x,y)}[j]) = E_{pk}(B_x[j]) * \lambda[j]$.

In the Secure_2P_Max protocol, one main observation (upon which one can also justify the correctness of the final output) is that if $f : x > y$, then $B_{\max(x,y)}[j] = h*x[j]+(1-h)*y[j]$ always holds for $1 \leq j \leq m$. Similarly, if $f : y > x$, then $B_{\max(x,y)}[j] = h*y[j]+(1-h)*x[j]$ always holds.

5) Secure_2P_Enc_Max:

Algorithm 4 Secure_2P_Max$([x], y) \rightarrow [\max(x, y)]$

Require: A_i has $([x], y)$ and (π_1, π_2), where $1 \leq x, y \leq l$ and $2^{m-1} \leq l < 2^m$; the private key $pr = pr_C$ is known only to C

1: A_i:

(a). Randomly choose the functionality f

(b). **for** $j = 1$ to m **do**:

- **if** $f : x > y$ **then**:
  - $W[j] \leftarrow E_{pk}(B_x[j]) * E_{pk}(B_x[j] * B_y[j])^{N-1}$
  - $\Gamma[j] \leftarrow E_{pk}(B_x[j] - B_y[j]) * E_{pk}(\hat{r}_j)$, where $\hat{r}_j \in_R \mathbb{Z}_N$

  else
  - $W[j] \leftarrow E_{pk}(B_y[j]) * E_{pk}(B_x[j] * B_y[j])^{N-1}$
  - $\Gamma[j] \leftarrow E_{pk}(B_y[j] - B_x[j]) * E_{pk}(\hat{r}_j)$, where $\hat{r}_j \in_R \mathbb{Z}_N$
- $T[j] \leftarrow E_{pk}(B_x[j] \oplus B_y[j])$
- $X[j] \leftarrow X[j-1]^{r_j} * T[j]$, where $r_j \in_R \mathbb{Z}_N$ and $X[0] = E_{pk}(0)$
- $\Phi[j] \leftarrow E_{pk}(-1) * X[j]$
- $Y[j] \leftarrow W[j] * \Phi[j]^{r'_j}$, where $r'_j \in_R \mathbb{Z}_N$

(c). $\Gamma' \leftarrow \pi_1(\Gamma)$

(d). $Y' \leftarrow \pi_2(Y)$; send Γ' and Y' to C

2: C:

(a). Receive Γ' and Y' from A_1

(b). Decryption, **for** $1 \leq j \leq m$ **do**: $M[j] \leftarrow D_{pr}(Y'[j])$

(c). **if** $\exists\ k$ such that $M[k] = 1$ **then**: $h \leftarrow 1$

else $h \leftarrow 0$

(d). **for** $1 \leq j \leq m$ **do**: $M'[j] \leftarrow \Gamma'[j]^h$

(e). Send M' and $E_{pk}(h)$ to A_1

3: A_i:

(a). Receive M' and $E_{pk}(h)$ from C

(b). $\widetilde{M} \leftarrow \pi_1^{-1}(M')$

(c). **for** $j = 1$ to m **do**:

- $\lambda[j] \leftarrow \widetilde{M}[j] * E_{pk}(h)^{N-\hat{r}_j}$
- **if** $f : x > y$ **then**: $E_{pk}(B_{\max(x,y)}[j]) \leftarrow E_{pk}(B_y[j]) * \lambda[j]$
  else $E_{pk}(B_{\max(x,y)}[j]) \leftarrow E_{pk}(B_x[j]) * \lambda[j]$

In this protocol, it is assumed that both x and y are unknown to A_i and C (whereas in Secure_2P_Max y is known to A_i). More specifically, A_i with private input $([x], [y])$ and C securely compute $[\max(x, y)]$. The main steps involved in Secure_2P_Enc_Max are

Algorithm 5 Secure_2P_Enc_Max($[x], [y]$) $\rightarrow$ [max(x, y)]

Require: A_i has $[x]$ and $[y]$, where $1 \leq x, y \leq l$ and $2^{m-1} \leq l < 2^m$; the private key $pr = pr_C$ is known only to C

1: A_i:

(a). **for** $j = 1$ to m **do**:

- $X'[j] \leftarrow E_{pk}(B_x[j]) * E_{pk}(r_{x,j})$, where $r_{x,j} \in_R \mathbb{Z}_N$
- $Y'[j] \leftarrow E_{pk}(B_y[j]) * E_{pk}(r_{y,j})$, where $r_{y,j} \in_R \mathbb{Z}_N$

(b). Send X' and Y' to C

2: C:

(a). **for** $j = 1$ to m **do**:

- $x'[j] \leftarrow D_{pr}(X'[j])$
- $y'[j] \leftarrow D_{pr}(Y'[j])$
- $\omega[j] \leftarrow x'[j] * y'[j] \mod N$
- Compute $\Omega[j] \leftarrow E_{pk}(\omega[j])$

(b). Send Ω to A_i

3: A_i:

(a). Receive Ω from C

(b). **for** $j = 1$ to m **do**:

- $\chi[j] \leftarrow E_{pk}(B_x[j])^{r_{y,j}} * E_{pk}(B_y[j])^{r_{x,j}}$
- $\chi'[j] \leftarrow \chi[j] * E_{pk}(r_{x,j} * r_{y,j})$
- $E_{pk}(B_x[j] * B_y[j]) \leftarrow \Omega[j] * \chi'[j]^{N-1}$

4: A_i and C proceed with steps 1-3 of Algorithm 4

shown in Algorithm 5. The basic idea used in Secure_2P_Enc_Max is the same as in Secure_2P_Max protocol. However, since both x and y are unknown, one cannot directly compute $E_{pk}(B_x[j] * B_y[j])$, which is required for the computation of $W[j]$ and $T[j]$, for $1 \leq j \leq m$. Other than this, the rest of steps are pretty much the same as in Algorithm 4. Therefore, the steps involved in securely computing $E_{pk}(B_x[j] * B_y[j])$, for $1 \leq j \leq m$ are provided here.

Initially, A_i randomizes $[x]$ and $[y]$ component-wise to compute $X'[j] = E_{pk}(B_x[j] + r_{x,j})$ and $Y'[j] = E_{pk}(B_y[j] + r_{y,j})$, for $1 \leq j \leq m$, where $r_{x,j}, r_{y,j} \in_R \mathbb{Z}_N$. Then, A sends X' and Y' to C. Upon receiving, C decrypts and multiplies them component-wise and proceeds as follows for $1 \leq j \leq m$:

- Compute $\omega[j] = x'[j] * y'[j] \bmod N$, where

$$x'[j] = D_{pr}(X'[j]) \text{ and } y'[j] = D_{pr}(Y'[j]);$$

- Encrypt the result: $\Omega[j] = E_{pk}(\omega[j])$.

After this, C sends Ω to A_i. Then, A_i performs the following homomorphic operations to compute $E_{pk}(B_x[j] * B_y[j])$ by removing extra random factors, for $1 \leq j \leq m$:

- $\chi[j] = E_{pk}(B_x[j])^{r_{y,j}} * E_{pk}(B_y[j])^{r_{x,j}}$. Observe that, due to homomorphic properties, $\chi[j] = E_{pk}(B_x[j] * r_{y,j} + B_y[j] * r_{x,j})$.
- Compute $\chi'[j] = \chi[j] * E_{pk}(r_{x,j} * r_{y,j})$.
- Finally, compute $E_{pk}(B_x[j] * B_y[j])$ as $\Omega[j] * \chi'[j]^{N-1}$.

Once A_i knows $E_{pk}(B_x[j] * B_y[j])$, for $1 \leq j \leq m$, he/she can compute Y locally, as mentioned in Algorithm 4. In addition, with the help of C, A_i can obliviously compute $[\max(x, y)]$. That is, A_i and C are involved in steps 1-3 of Algorithm 4 to obliviously compute $[\max(x, y)]$.

6) Secure_MP_Max:

In this protocol, we consider n parties such that party A_i holds private input a_i, for $1 \leq i \leq n$. The goal of the secure multi-party computation of maximum (denoted by Secure_MP_Max) is to compute $[\max(a_1, \ldots, a_n)] = [a_{\max}]$ in a privacy-preserving manner. Our protocol utilizes two sub-routines, namely Secure_2P_Max and Secure_2P_Enc_Max as the building blocks. The proposed Secure_MP_Max protocol is an iterative approach, and it computes the desired output in a hierarchical fashion. In each iteration, the maximum between a pair of values is computed and fed as input to the next iteration. Therefore, a binary execution tree is generated in a bottom-up fashion, where leaf node i represents party A_i holding his/her private input a_i. The root node is A_1 holding the final result $[a_{\max}]$, which is broadcasted to other parties (i.e., A_i's, for $2 \leq i \leq n$). Hence, at the end of the Secure_MP_Max protocol, the output $[a_{\max}]$ is known to all A_i's, for $1 \leq i \leq n$.

Algorithm 6 Secure_MP_Max$(a_1, \ldots, a_n) \rightarrow [a_{\max}]$

Require: a_i is private to A_i, for $1 \leq i \leq n$; (Note: $pr = pr_C$ is known only to C)

1: $num \leftarrow n$

2: **for** $k = 1$ to $\lceil \log_2 n \rceil$:

(a). **for** $1 \leq i \leq \lfloor \frac{num}{2} \rfloor$:

- **if** $k = 1$ **then**:
  - A_{2i} sends $[a_{2i}]$ to $A_{2(i-1)+1}$, where a_{2i} is the private input of A_{2i}
  - $A_{2(i-1)+1}$ and C are jointly involved in Secure_2P_Max$([a_{2i}], a_{2(i-1)+1})$, where $a_{2(i-1)+1}$ is the private input of $A_{2(i-1)+1}$

  else
  - A_{2ki-1} sends $[x_{k,i}]$ to $A_{2k(i-1)+1}$, where $[x_{k,i}]$ is the output received by A_{2ki-1} from iteration $k-1$
  - $A_{2k(i-1)+1}$ and C are jointly involved in the Secure_2P_Enc_Max $([x_{k,i}], [y_{k,i}])$, where $[y_{k,i}]$ is the output received by the party $A_{2k(i-1)+1}$ from iteration $k-1$

(b). $num \leftarrow \lceil \frac{num}{2} \rceil$

3: A_1:

(a.) Send $[a_{\max}]$ to A_i, for $2 \leq i \leq n$

The main steps involved in the Secure_MP_Max protocol are highlighted in Algorithm 6. Initially, each party of A_i assigns n to the global variable num, where num represents the number of parties involved in each iteration. Since the Secure_MP_Max protocol executes in a binary tree hierarchy (bottom-up fashion), we have $\lceil \log_2 n \rceil$ iterations, and the number of involved parties varies in each iteration. In the first iteration (i.e., $k = 1$), party A_{2i} computes $[a_{2i}]$ and sends it to $A_{2(i-1)+1}$, for $1 \leq i \leq \lfloor \frac{num}{2} \rfloor$ (note that, when $k = 1$, the number of non-overlapping pairs are $\lfloor \frac{num}{2} \rfloor$). More specifically, A_2 sends $[a_2]$ to A_1, A_4 sends $[a_4]$ to A_3, and so on. After this, $A_{2(i-1)+1}$ and C are involved in Secure_2P_Max$([a_{2i}], a_{2(i-1)+1})$ protocol, for $1 \leq i \leq \lfloor \frac{num}{2} \rfloor$, where $a_{2(i-1)+1}$ is the private input of $A_{2(i-1)+1}$. As mentioned earlier, at the end of the Secure_2P_Max protocol, only the party $A_{2(i-1)+1}$ receives $[\max(a_{2i}, a_{2(i-1)+1})]$, and nothing is revealed to C, for $1 \leq i \leq \lfloor \frac{num}{2} \rfloor$. Also, at the end of first iteration, the value of num is updated to $\lceil \frac{num}{2} \rceil$.

During the k^{th} iteration, only the parties who received the output from the previous iteration are involved, for $2 \leq k \leq \lceil \log_2 n \rceil$. For example, during the second iteration (i.e.,

$k = 2$), only A_1, A_3, and so on are involved. In k^{th} iteration, A_{2ki-1} sends $[x_{k,i}]$ to $A_{2k(i-1)+1}$, where $[x_{k,i}]$ is the output received by A_{2ki-1} from the iteration $k-1$, for $1 \leq i \leq \lfloor \frac{num}{2} \rfloor$. Upon receiving, $A_{2k(i-1)+1}$ and C are involved in the Secure_2P_Enc_Max($[x_{k,i}], [y_{k,i}]$) protocol, where $[y_{k,i}]$ is the output received by $A_{2k(i-1)+1}$ from iteration $k-1$, for $1 \leq i \leq \lfloor \frac{num}{2} \rfloor$. Note that in each iteration, *num* is updated to $\lceil \frac{num}{2} \rceil$. We observe that $[x_{k,i}]$ and $[y_{k,i}]$ are the maximum values retrieved from the sub-trees rooted at A_{2ki-1} and $A_{2k(i-1)+1}$ (in iteration k), respectively, for $1 \leq i \leq \lfloor \frac{num}{2} \rfloor$ and $2 \leq k \leq \lceil \log_2 n \rceil$.

At the end of the last iteration (i.e., $k = \lceil \log_2 n \rceil$), only A_1 receives $[a_{\max}]$, which is forwarded to A_i, for $2 \leq i \leq n$.

Example: *Without loss of generality, consider 5 households in a neighborhood (i.e., $n = 5$). Then, in the Secure_MP_Max protocol, num is initially set to 5 and we have 3 iterations (i.e., k runs from 1 to 3). Various intermediate steps of executing the Secure_MP_Max protocol are summarized:*

(i). $k = 1$

- A_2 *sends* $[a_2]$ *to* A_1. *Then,* A_1*, with private input* $([a_2], a_1)$ *, and* C *are involved in Secure_2P_Max. At the end of this step,* A_1 *knows* $[\max(a_1, a_2)]$.
- A_4 *sends* $[a_4]$ *to* A_3. *After this,* A_3 *computes* $[\max(a_3, a_4)]$ *by executing Secure_2P_Max with* C. *Note that* A_5 *does not participate in this iteration.*
- $num = \lceil \frac{num}{2} \rceil = 3$

(ii). $k = 2$

- A_3 *sends* $[\max(a_3, a_4)]$ *to* A_1. *Then,* A_1*, with private input* $([\max(a_3, a_4)], [\max(a_1, a_2)])$*, and* C *are jointly involved in Secure_2P_Enc_Max. At the end of this step,* A_1 *gets* $[\max(a_1, \ldots, a_4)]$. *Observe that* A_5 *does not participate in this iteration.*
- $num = 2$

(iii). $k = 3$

- *A_5 sends $[a_5]$ to A_1. Now, A_1, with private input $([a_5], [\max(a_1, \ldots, a_4)])$, and C are involved in Secure_2P_Enc_Max to compute $[\max(a_1, \ldots, a_5)]$.*
- *$num = 1$*

At the end of the third iteration, the final output $[a_{\max}] = [\max(a_1, \ldots, a_5)]$ is known only to A_1 and forwarded to other parties. □

7) P-PUC$_1^*$:

We now discuss how the above security primitives are used in constructing the proposed P-PUC$_1^*$ protocol. The main steps involved in P-PUC$_1^*$ are shown in Algorithm 7.

To start with, the n consumers and the utility company C are initially involved in the Sum_Random_Shares sub-routine to compute the random shares of a (step 1 of Algorithm 7). Then, A_1 encrypts his/her random share s_1 (using $pk = pk_{A_1}$) and sends it to C. Upon receiving, C computes $E_{pk}(a)$ using his/her random share s_2 by computing $E_{pk}(s_1) * E_{pk}(s_2) = E_{pk}(s_1 + s_2 \bmod N) = E_{pk}(a)$. Note that $s_1 + s_2 \bmod N$ is always equal to a. After this, C, with private input $E_{pk}(a)$, and consumer A_1 are involved in the Secure_Binary_Conv protocol (step 2(c) of Algorithm 7). At the end of this step, only C knows $[a]$. Then, C, with private input $([a], t)$, and A_1 engage involve in the Secure_Comp protocol to securely compare a and t. At the end, the output h (denoting whether $a > t$ or not) is revealed only to A_1 and forwarded to other consumers (step 3(a) of of Algorithm 7). It is important to note that all consumers will know whether $a > t$ or not, but this information is expected to be revealed to the consumers by design; therefore, it is considered to be acceptable.

If $a > t$ (i.e., $h = 1$), then the consumers, with their respective private inputs a_i's, and C, with private key pr_C (i.e., $(pk, pr) = (pk_C, pr_C)$), engage in the Secure_MP_Max protocol to securely compute $[a_{\max}]$ (step 4(a) of Algorithm 7). At the end of this step, the output $[a_{\max}]$ is revealed to all consumers. After this, each A_i independently checks whether the maximum value belongs to him/her (i.e., whether $a_i = a_{\max}$ or not) in an

Algorithm 7 P-PUC$_1^*$

Require: a_i is private to A_i, for $1 \leq i \leq n$ (Note: pr_C and t are known only to C, pr_{A_1} is known only to A_1; pk_C and pk_{A_1} are public)

1: {**For steps 1 & 2** $(pk, pr) = (pk_{A_1}, pr_{A_1})$}

2: Sum_Random_Shares$(a_1, \ldots, a_n)$

3: A_1 and C:

(a). A_1 sends $E_{pk}(s_1)$ to C

(b). C computes $E_{pk}(a) \leftarrow E_{pk}(s_1) * E_{pk}(s_2)$

(c). A_1 and C are involved in the secure comparison protocol:

- $[a] \leftarrow$ Secure_Binary_Conv$(E_{pk}(a))$, here the output $[a]$ is revealed only to C
- $h \leftarrow$ Secure_Comp$([a], t)$, here the output h is revealed only to A_1

4: A_1:

(a). Send h to A_i, for $2 \leq i \leq n$

{**For step 4** $(pk, pr) = (pk_C, pr_C)$}

5: **if** $h = 1$ (denoting $a > t$) **then**:

(a). $[a_{\max}] \leftarrow$ Secure_MP_Max$(a_1, \ldots, a_n)$

(b). A_i, **for** $i = 1$ to n **do**:

- $\alpha_i \leftarrow$ Secure_Equality$(a_i, [a_{\max}])$
- **if** $\alpha_i = 1$ (denoting $a_i = a_{\max}$) **then**:
  - Reduce a_i

6: Repeat the steps until $a \leq t$

oblivious manner with C by using the Secure_Equality protocol (step 4(b) of Algorithm 7), for $1 \leq i \leq n$. We adopt Secure_Equality protocol steps shown below:

1. Initially, A_i computes $Z[i] = E_{pk}(r'_i * (a_{\max} - a_i))$, where $r'_i \in_R \mathbb{Z}_N$, for $1 \leq i \leq n$. After this, A_i sends $Z[i]$ to A_1, for $2 \leq i \leq n$.

2. A_1 randomly permutes Z using a random permutation function π, i.e., he/she computes $Z_1 = \pi(Z)$ and sends Z_1 to C.

3. Then, C decrypts each entry in Z_1 and computes a new vector H as follows. If $D_{pr}(Z_1[i]) = 0$, then $H[i] = E_{pk}(1)$. Else, $H[i] = E_{pk}(0)$. C sends H to A_1.

4. A_1 computes the inverse permutation on H to get $L = \pi^{-1}(H)$ and sends $L[i]$ to A_i, for $2 \leq i \leq n$. Observe that $L[i] = E_{pk}(1)$ iff $a_i = a_{\max}$. Otherwise, $E_{pk}(0)$.

5. Now A_i proceeds as follows, for $1 \leq i \leq n$:

   - Randomizes $L[i]$ by computing $L'[i] = L[i] * E_{pk}(r_i)$, where $r_i \in_R \mathbb{Z}_N$, and sends $L'[i]$ to C.
   - Upon receiving $L'[i]$, C decrypts it to get $\beta_i = D_{pr}(L'[i])$ and sends the result to A_i.
   - Finally, A_i removes the randomization and gets his/her result as $\alpha_i = \beta_i - r_i \mod N$.

Note that since $L[i] = E_{pk}(1)$ iff $a_i = a_{\max}$, the output $\alpha_i = 1$ always holds. Similarly, if $a_i \neq a_{\max}$, then $\alpha_i = 0$ always holds.

At the end of the Secure_Equality protocol, each party knows whether his/her private input is the maximum value. After this, only the self-identified consumer (i.e., for which $a_{\max} = a_i$) reduces his/her power usage accordingly. The process is repeated by computing the new random shares of updated a in each iteration until $a \leq t$.

5.1.2. The P-PUC$_2^*$ Protocol. The second protocol, denoted as P-PUC$_2^*$, is based on Strategy 2 as mentioned in Section 1.1.. The basic idea of P-PUC$_2^*$ is to securely compute $\frac{t}{a}$ using a secure division protocol whenever $a > t$. Once consumers know the value of $\frac{t}{a}$, they will reduce their power consumption locally as mentioned in Strategy 2. Since the existing division protocols are inefficient, we present a new solution to the two-party secure division problem (denoted by Secure_Div).

1) Secure_Div:

In the proposed Secure_Div protocol, the utility company C, with private input $(E_{pk}(a), t)$, and A_1, with private key $pr = pr_{A_1}$, are involved (two-party computation). The goal of the Secure_Div protocol is to compute $\frac{t}{a}$ such that t and a are not revealed to A_1 and a is not revealed to C. This implies that the output $\frac{t}{a}$ is revealed only to A_1 and forwarded to A_i's, for $2 \leq i \leq n$. We emphasize that the output should not be revealed to C because C can easily infer a from $\frac{t}{a}$ as t is already known to C. The overall steps involved in the proposed Secure_Div protocol are given in Algorithm 8.

Algorithm 8 Secure_Div$(E_{pk}(a), t) \rightarrow \frac{t}{a}$

Require: C has $(E_{pk}(a), t)$, where $1 \leq a, t \leq l$, and θ is a scalar factor assumed to be public; (Note: Here $(pk, pr) = (pk_{A_1}, pr_{A_1})$ and pr_{A_1} is known only to A_1)

1: C:

(a). **for** $i = 1$ to l **do**:

- $E_{pk}(a - i) \leftarrow E_{pk}(a) * E_{pk}(N - i)$
- $P[i] \leftarrow E_{pk}(a - i)^{\bar{r}_i}$, where $\bar{r}_i \in_R \mathbb{Z}_N$
- $Q[i] \leftarrow \left\lceil \theta * \frac{t}{i} \right\rceil$

(b). $P' \leftarrow \pi(P)$; send P' to A_1

2: A_1:

(a). Receive P' from C

(b). **for** $i = 1$ to l **do**:

- $\tau[i] \leftarrow D_{pr}(P'[i])$
- **if** $\tau[i] = 0$ **then**:
  - $u[i] \leftarrow 1$

  else
  - $u[i] \leftarrow 0$

(c). $U[i] \leftarrow E_{pk}(u[i])$, for $1 \leq i \leq l$; send U to C

3: C:

(a). Receive U from C

(b). $V \leftarrow \pi^{-1}(U)$

(c). $\gamma \leftarrow \prod_{i=1}^{l} V[i]^{Q[i]}$; send γ to A_1

4: A_1:

(a). Receive γ from C

(b). $\frac{t}{a} \leftarrow \frac{D_{pr}(\gamma)}{\theta}$

(c). Send $\frac{t}{a}$ to A_i, for $2 \leq i \leq n$

To start with, C initially computes $E_{pk}(a - i)$ and randomizes it by computing $P[i] = E_{pk}(a - i)^{\bar{r}_i} = E_{pk}(\bar{r}_i * (a - i))$, where $\bar{r}_i$ is a random number in $\mathbb{Z}_N$, for $1 \leq i \leq l$. Note that exactly one of the entries in P is an encryption of 0, namely $P[a] = E_{pk}(0)$. In addition, C also computes vector Q such that $Q[i] = \left\lceil \theta * \frac{t}{i} \right\rceil$, for $1 \leq i \leq l$, where θ is a scalar factor that is treated as public information. It is emphasized that θ acts as a precision parameter in computing $\frac{t}{a}$. Then, C permutes P using a random permutation function π (known only to C) and sends the resulting encrypted vector P' to A_1. Upon

Algorithm 9 P-PUC$_2^*$

Require: a_i is private to A_i, for $1 \leq i \leq n$ (Note: pr_C and t are known only to C, pr_{A_1} is known only to A_1, pk_C and pk_{A_1} are public)

Steps 1 - 3 are the same as in Algorithm 7

4. **if** $a > t$ **then**:

(a). A_1 and C:

- Secure_Div$(E_{pk}(a), t)$ (note that the output $\frac{t}{a}$ is revealed only to A_i's)

(b). A_i, **for** $1 \leq i \leq n$ **do**:

- $\delta_i \leftarrow a_i * (1 - \frac{t}{a})$
- Reduce a_i using δ_i

receiving P', A_1 decrypts it component-wise to get $\tau[i] = D_{pr}(P'[i])$. After this, he/she computes a vector u based on τ such that $u[i] = 1$ if $\tau[i] = 0$, and $u[i] = 0$ otherwise, for $1 \leq i \leq l$. Then A_1 encrypts u component-wise to get U, i.e., $U[i] = E_{pk}(u[i])$, for $1 \leq i \leq l$, and sends U to C.

Upon receiving U, C performs inverse permutation to get $V = \pi^{-1}(U)$ and computes $\gamma = \prod_{i=1}^{l} V[i]^{Q[i]}$. After this, C sends γ to A_1. Finally, A_1 decrypts γ and divides the result by the scalar factor θ to get $\frac{t}{a}$ (step 4(b) of Algorithm 8). Furthermore, A_1 sends the value of $\frac{t}{a}$ to other consumers.

2) P-PUC$_2^*$:

Since the proposed P-PUC$_2^*$ protocol is based on Strategy 2, it is non-iterative (whereas P-PUC$_1^*$ is iterative). The main steps involved in P-PUC$_2^*$ are shown in Algorithm 9. The first three steps are the same as in the P-PUC$_1^*$ protocol. Similar to P-PUC$_1^*$, at the end of Step 3, all consumers know the output of the Secure_Comp protocol, i.e., whether $a > t$ or not.

Whenever $a > t$, C, with private input $(E_{pk}(a), t)$, and A_1 engage in the Secure_Div protocol (Step 4(a) of Algorithm 9). We observe that some of the computations performed in the Secure_Binary_Conv protocol can be re-used in the Secure_Div protocol. This is because the encrypted vector V that needs to be computed in the Secure_Div protocol is the same as in Secure_Binary_Conv. Therefore, during the actual implementation of

Secure_Div, C can directly proceed to Step 3(c) to compute γ using V, which was already computed in the Secure_Binary_Conv protocol. Therefore, the overall computation complexity of P-PUC$_1^*$ is reduced.

At the end of the Secure_Div protocol, each consumer knows the value of $\frac{t}{a}$. We emphasize that during this process, neither the value of t nor a is revealed to the consumers. In addition, a is not revealed to C. After this, each consumer A_i performs the following operations, for $1 \leq i \leq n$:

- Compute δ_i, the least amount of power to be reduced, as $a_i * (1 - \frac{t}{a})$. Note that, since $a > t$, we always have $0 < \frac{t}{a} < 1$.
- Finally, A_i reduces his/her power usage based on δ_i.

5.2. SECURITY ANALYSIS

To prove a protocol's security under the semi-honest model, we adopted the well-known security definitions from the literature of secure multi-party computation (MPC). First of all, since the Secure_2P_Max protocol is more complex than other sub-protocols, we are motivated to provide its security proof rather than providing proofs for each sub-protocol. Therefore, we only include a formal security proof for the Secure_2P_Max protocol based on the standard simulation argument [64]. Nevertheless, we stress that similar proof strategies can be used to show that other sub-protocols are secure under the semi-honest model. We now provide a formal security proof for Secure_2P_Max under the semi-honest model.

As mentioned in Section 3.2.1., to formally prove that Secure_2P_Max is secure [64] under the semi-honest model, we need to show that the simulated execution image of Secure_2P_Max is computationally indistinguishable from the actual execution image of Secure_2P_Max. An execution image generally includes the messages exchanged and the information computed from these messages. Therefore, according to Algorithm 4, the execution image of C can be denoted by Π_C, where

$$\Pi_C = \{\langle \Gamma'[j], \mu[j] + \hat{r}_j \bmod N \rangle, \langle Y'[j], h \rangle \mid \text{for } 1 \leq j \leq m\}$$

Observe that $\mu[j] + \hat{r}_j \bmod N$ is derived upon decrypting $\Gamma'[j]$, where the modulo operator is implicit in the decryption function. Also, C receives Y' from A_1. Let h denote the (oblivious) comparison result computed from Y'. Without loss of generality, suppose the simulated image of C be Π_C^S, where

$$\Pi_C^S = \{\langle s_1[j], s_2[j]\rangle, \langle s_3[j], h'\rangle \mid \text{for } 1 \leq j \leq m\}$$

Here $s_1[j]$ and $s_3[j]$ are randomly generated from $\mathbb{Z}_{N^2}$, and $s_2[j]$ is randomly generated from $\mathbb{Z}_N$. In addition, h' is a random bit. Since E_{pk} is a semantically secure encryption scheme with resulting ciphertext size less than N^2, $\Gamma'[j]$ and $Y'[j]$ are computationally indistinguishable from $s_1[j]$ and $s_3[j]$. Also, as $\hat{r}_j$ is randomly generated, $\mu[j] + \hat{r}_j \bmod N$ is computationally indistinguishable from $s_2[j]$. Furthermore, because the functionality is randomly chosen by A_1 (at step 1(a) of Algorithm 4), h is either 0 or 1 with equal probability. Thus, h is computationally indistinguishable from h'. Combining all these results allows us to conclude that Π_C is computationally indistinguishable from Π_C^S. This implies that, during the execution of Secure_2P_Max, C does not learn anything about x, y, and the actual comparison result. Intuitively speaking, the information P_2 has during an execution of Secure_2P_Max is either random or pseudo-random, so this information does not disclose anything regarding x and y. Additionally, as f is known only to A_1, the actual comparison result is oblivious to C.

On the other hand, the execution image of A_1, denoted by Π_{A_1}, is given by

$$\Pi_{A_1} = \{M'[j], E_{pk}(h) \mid \text{for } 1 \leq j \leq m\}$$

Here, $M'[j]$ is an encrypted value, which is random in $\mathbb{Z}_{N^2}$, received from C (at Step 3(a) of Algorithm 4). Let the simulated image of A_1 be $\Pi_{A_1}^S$, where

$$\Pi_{A_1}^S = \{s_4[j], b \mid \text{for } 1 \leq i \leq m\}$$

Both $s_4[j]$ and b are randomly generated from $\mathbb{Z}_{N^2}$. Since E_{pk} is a semantic secure encryption scheme with resulting ciphertext size less than N^2, $M'[j]$ and $E_{pk}(h)$ are

computationally indistinguishable from $s_4[j]$ and b. Therefore, Π_{A_1} is computationally indistinguishable from $\Pi^S_{A_1}$. This implies that A_1 does not learn anything about the comparison result or x. We can say Secure_2P_Max is secure under the semi-honest model when combined with the previous analysis. In a similar way, we can formally prove that all our sub-protocols are secure under the semi-honest model.

Based on the previous discussions, it is clear that the intermediate values in each of the proposed sub-protocols are either random or pseudo-random. Therefore, no information is revealed to the involved parties other than the expected output. Additionally, in the proposed P-PUC protocols, the output of a sub-protocol fed as an input to the next sub-protocol. Therefore, the sub-protocols are independent from one another, but they are the building blocks for the proposed P-PUC protocols. In addition, to maximize security protection, it is emphasized that the outputs in all our sub-protocols are either random or pseudo-random values. For example, the output of Secure_Binary_Conv, which is an encrypted value $E_{pk}(a)$, is passed as an input to the Secure_Comp protocol. Therefore, based on the well-known Composition Theorem [64], we claim that a sequential composition of the sub-protocols leads to our new proposed P-PUC protocols that are secure under the semi-honest model.

5.3. EXPERIMENTAL RESULTS

In this section, we first empirically analyze the computation costs of the proposed protocols based on different parameters. Then, we compare the computation costs of the protocols in Chapter 4. with our new proposed protocols in this chapter. We emphasize that P-PUC$_1$ and P-PUC$_2$ are not fully secure.

The protocols were implemented in C using the Paillier encryption scheme [65]. All experiments were conducted on a Linux machine running Ubuntu 10.04 LTS with Intel® Xeon® Six-Core™ 3.07GHz and 12GB RAM. For the rest of this section, we fix the Paillier encryption key size K to 1024 bits (for both C and A_1) since it is the commonly used key size in practice (Under the Paillier cryptosystem, a 1024-bit encryption key results

in 2048-bit ciphertexts, which are sufficiently secure for most applications). Nevertheless, we observed that for any given fixed parameters, the computation costs of the proposed protocols are increased by a factor of nearly 7 whenever K is doubled. In our experiments, we randomly generate the values of a_i's and t such that $a > t$ and $1 \leq a, t \leq l$, where l is the domain size.

5.3.1. Performance of P-PUC$_1^*$ and P-PUC$_2^*$. We first compute the computation costs of different parties involved in the P-PUC$_1^*$ protocol for $n = 50$ and varying values of l. That is, the running times of A_1, C, and A_i, where $i \neq 1$, are analyzed for one iteration. Note that the costs of individual parties in P-PUC$_1^*$ do not vary between iterations for any given fixed parameters (assuming $a > t$ holds under different iterations). Therefore, we present the computation costs of the individual parties in a given iteration. Since the computation costs of A_i's are different, for $2 \leq i \leq n$, we present the average cost for A_i. As shown in Figure 5.1.(a), the computation costs of A_1, C, and A_i are 7.78, 4.32, and 3.17 seconds, respectively, for $l = 1000$. As expected, the computation times of every involved parties increase with l, but the trend of computation times increasing for A_i is very small. For example, the computation time of A_1 increases from 7.78 to 27.37 seconds while that of C increases from 4.32 to 14.91 seconds when l is changed from 1000 to 5000. However, the cost of A_i increases slightly from 3.17 to 4.18 seconds when l is changed from 1000 to 5000. The high computation costs of A_1 and C come from two sub-protocols, namely Secure_Binary_Conv and Secure_MP_Max. We emphasize that A_i does not participate in the Secure_Binary_Conv protocol. Also, not all A_i's participate in the Secure_MP_Max protocol; therefore, resulting in small computation costs for A_i's, where $2 \leq i \leq n$.

In a similar manner, the computation costs of A_1 and C in the P-PUC$_2^*$ protocol are analyzed for varying l with $n = 50$ and $\theta = 100$, where θ is the scalar factor (note that the computation time of P-PUC$_2^*$ is independent of θ). The computation costs of different parties in P-PUC$_2^*$ are as shown in Figure 5.1.(b). We emphasize that the computation costs of A_i's in P-PUC$_2^*$ are negligible since they do not participate in any of the corresponding sub-protocols. On the other hand, for $l = 1000$, the computation costs of A_1 and C are

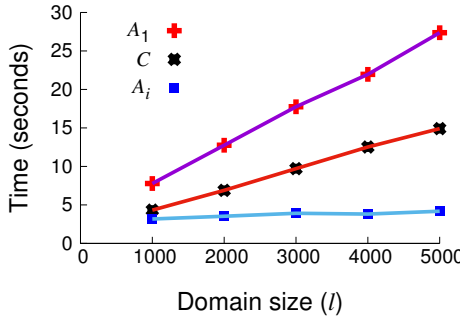
9.21 and 5.65 seconds, respectively. We observe that the costs (i.e., both computation and communication) of A_1 and C grow linearly with l. For example, the computation time of A_1 increases from 9.21 to 46.19 seconds when l is varied from 1000 to 5000 (i.e., the computation time increases by a factor of nearly 5). In addition, the computation cost of C increases from 5.65 to 25.34 seconds when l is changed from 1000 to 5000, resulting in a similar trend.

We now compare the total computation costs of P-PUC$_1^*$ (for one iteration) and P-PUC$_2^*$ for $l = 1000$ and varying n, where n denotes the number of households from a given neighborhood. As shown in Figure 5.1.(c), the total running time of P-PUC$_1^*$ varies from 12.12 to 19.3 seconds when n is changed from 50 to 250. On the other hand, the total running time of P-PUC$_2^*$ remains around 14.9 seconds since it is independent of n. Following from Figure 5.1.(c), P-PUC$_1^*$ works better than P-PUC$_2^*$ when the neighborhood size is below 100. After 100, P-PUC$_2^*$ gives a better performance.

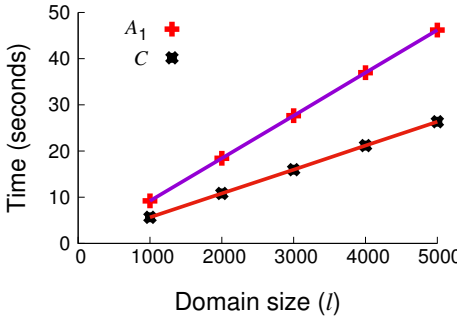
In general, the power usage control process is performed once every 30 minutes. Therefore, based on the above results, we conclude that the proposed protocols are very practical. Besides practicality, the main advantage of the proposed protocols is that they execute the entire $TPUC$ process in a privacy-preserving manner. In addition, following from our empirical analysis, P-PUC$_1^*$ and P-PUC$_2^*$ provide similar performance in small neighborhood. When the neighborhood size is greater than 100, P-PUC$_2^*$ is more steady and efficient than P-PUC$_1^*$.

The U.S. Energy Information Administration (EIA) reported in 2010 that the average electricity consumption for an American household was 958 kWh per month, i.e., roughly 32 kWh per day or 2 kWh per hour [76]. Assuming the power consumption data is collected (in encrypted form) every 30 minutes, the aggregated value a will always be less than 5000, even for a neighborhood of 500 households. Therefore, under our problem domain, $l = 5000$ is sufficient in satisfying the requirement of $1 \leq a \leq l$.

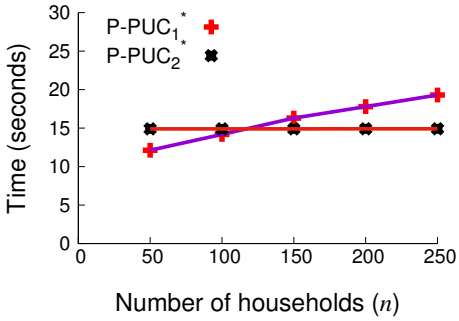
5.3.2. Performance Comparison with P-PUC$_1$ and P-PUC$_2$. Finally, we compare the computation costs of the protocols here with the protocols in Chapter 4.. For $n = 50$, the computation cost of P-PUC$_1$ is less than 2 seconds. P-PUC$_1$ is more efficient



(a) Complexity of P-PUC$_1^*$ for $n = 50$



(b) Complexity of P-PUC$_2^*$ for $n = 50$



(c) Complexity: P-PUC$_1^*$ Vs. P-PUC$_2^*$

Figure 5.1. Empirical results of P-PUC$_1^*$ and P-PUC$_2^*$ based on Paillier cryptosystem for K =1024.

than P-PUC$_1^*$. For example, when $n = 50$ and $l = 1000$, P-PUC$_1^*$ takes 12.12 seconds (i.e, P-PUC$_1$ is 6 times more efficient than P-PUC$_1^*$). Nevertheless, this efficiency gain comes at the expense of putting the user's privacy at risk in P-PUC$_1$ whereas P-PUC$_1^*$ protects the user's privacy.

On the other hand, the computation cost of P-PUC$_2$ is roughly around 50 hours for $n = 50$ because the existing secure division protocol [75] utilized in P-PUC$_2$ is very inefficient. More specifically, a single execution of the secure division protocol [75] took around an hour, and we need to execute it n number of times in P-PUC$_2$. Hence, P-PUC$_2^*$ is significantly more efficient than P-PUC$_2$. In particular, for $n = 50$ and $l = 1000$, P-PUC$_2^*$ (which takes 12.12 seconds) is around 14851 times more efficient than P-PUC$_2$.

In summary, P-PUC$_1^*$ is more secure but less efficient than P-PUC$_1$. However, P-PUC$_2^*$ is more secure as well as efficient when compared to P-PUC$_2$.

6. OUTSOURCEABLE PRIVACY-PRESERVING POWER USAGE CONTROL

In *Outsourceable Privacy-Preserving Power Usage Control in a Smart Grid* [15], 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. Details will be provided in this chapter. The proposed protocol is secure under the semi-honest model and satisfies all the security requirements discussed in Section 1.1.2.. Due to the fact that all computations are outsourced to the cloud servers and are only performed on encrypted data, the existing P-PUC protocols cannot be applied to this problem setting. In addition, the proposed protocol is more efficient because it takes advantage of both secret sharing based secure computation and Yao's garbled circuit [69]. The proposed protocol consists of three stages: (1) collecting and integrating data, (2) comparing a and t, and (3) computing the δ_i values. At the first stage, the two cloud servers collect the average power consumption data a_i from each household A_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 δ 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 are given in Section 6.1..

The rest of this chapter is organized as follows: Section 6.1. provides the detailed implementation of the proposed OP-PUC protocol, and Section 6.2. presents empirical results to show the practicality of OP-PUC.

6.1. OUTSOURCEABLE PRIVACY-PRESERVING POWER USAGE CONTROL PROTOCOLS

In this section, the same notations from the previous chapters, summarized in Table 3.1., are adopted. As shown in Section 1.1., the proposed OP-PUC protocol adopts the

following two power usage control strategies when $a > t$: (1) reducing the power usage for the user who uses the maximum amount of energy among all users, and (2) providing the specific power reduction amount for each individual user in a particular neighborhood.

6.1.1. The First Stage of OP-PUC. In this problem setting, the protocol is executed by two cloud servers. First, the cloud servers need to gather the necessary data from power customers and the utility company. Then, the cloud servers must compare the total power consumption of those customers a with the threshold given by utility company t. If $a > t$, users need to reduce their power usage for the next period. The first stage of OP-PUC is data collection.

During the first stage, data must be hidden before it is outsourced. In the previous P-PUC protocols, homomorphic encryptions were utilized to encrypt the power usage data. However, if the homomorphic encryption approach is extended to this outsourced environment, huge computations would be incurred on the cloud servers. For example, in previous P-PUC protocols the coordinator (party that is in duty of computation) deals with one input of cypher-text and one input of plain-text algorithm, whereas remote server (counterpart of coordinator) in the new OP-PUC protocols need to deal with two cypher-text inputs. When this algorithm has multiplications of inputs involved, there are far more computation costs for the second algorithm than the first one. This is not applicable in our case. Therefore, to have a more efficient protocol, a secret sharing approach is adopted for the data collection stage.

To get shared inputs, the Secure_Split protocol presented in Algorithm 10, where N is assumed to be a large number, is used. In this protocol, A splits its input value α to two random values α' and α'', so that $\alpha' + \alpha'' = \alpha \mod N$, and sends them to S_1 and S_2, respectively. At the end, S_1 holds α' and S_2 holds α''. They do not know anything about α except for N.

Algorithm 11 gives the main steps for the data collection stage of OP-PUC. For each user A_i, the power consumption value a_i is split into a_i' and a_i'' and sent to servers S_1 and S_2 using Secure_Split. At the same time, the utility company C 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

Algorithm 10 Secure_Split(α) $\rightarrow (\alpha', \alpha'')$

Require: A has α and N where $\alpha < N$

1: A:
   (a) $\alpha' \leftarrow \alpha + r \bmod N$, where $r \in_R \mathbb{Z}_N$
   (b) $\alpha'' \leftarrow N - r$
   (c) Send α' to S_1 and send α'' to S_2
2: S_1 and S_2:
   (a) Receive α' and α'' respectively

Algorithm 11 Data_Collection$\rightarrow \{(a'_1, a''_1), \ldots, (a'_n, a''_n), (a', a''), (t', t'')\}$

Require: A_i has a_i where $1 \leq i \leq n$, C has t, and N is publicly known

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

$a' = \sum_{i=1}^{n} a'_i$ 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.

6.1.2. 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, a secure comparison protocol is needed to compare a and t with secret shares of each value as inputs. Use of a garbled circuit to securely perform the comparison task is considered because the existing secure comparison protocols [77, 78, 79, 80, 81] are not directly applicable in this problem domain. These protocols require that the inputs are the actual values. In addition, the garbled circuit is known for its efficiency in securely evaluating 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 *Faster secure two party computation using garbled circuits*[82]. In this paper, it is assumed that the secure comparison protocol built by a garbled circuit is denoted by Secure_Comparison$(a', a'', t', t'') \rightarrow b$. The protocol is performed by S_1 and S_2, where a' and t' are the inputs of S_1, and a'' and t''

Algorithm 12 Secure_Maximum$\{(a'_1, a''_1), ..., (a'_n, a''_n)\} \rightarrow (a_m ax', a_m ax'')$

Require: A has $a'_1, ..., a'_n$, B has $a''_1, ..., a''_n$, N is public

1: $a_i = a'_i + a''_i \mod N$ for $1 \leq i \leq n$
2: $num \leftarrow n$
3: **for** $k = 1$ to $\log_2 n$:
   (a) **for** $1 \leq i \leq num/2$:
      - $a_{2k(i-1)+1} \leftarrow$Secure_Maxof2$(a_{2k(i-1)+1}, a_{2ki})$
   (b) $num \leftarrow num/2$

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.

6.1.3. The Third Stage of OP-PUC Based on Strategy 1. For Strategy 1, the user with the highest power consumption is selected and ordered to reduce his power usage. During the process, the cloud servers are not allowed to know which user has been chosen. Basically, in this stage, a Secure_Maximum protocol is used to securely pick out the maximum value among n shared values. The garbled circuit approach is utilized to implement Secure_Maximum.

To implement Secure_Maximum, the protocol Secure_Maxof2(a, b) is easy to derive from a simple Secure_Comparison protocol introduced in the last subsection. That is, firstly compute a comparison between those two values, then use the result (either 0 or 1) to multiply the first value, plus the multiplication of the second value and reverse of the result. Explicitly, to judge the maximum of value a and b, firstly we compare those two and get the 1-bit result c. After this we can compute $\max(a, b) = c \times a + \overline{c} \times b$.

For maximum among multiple values, the protocol becomes an iterative approach to compute maximum number in a hierarchical fashion. In each loop, maximum between two values is computed and are treat as input to the next iteration. Algorithm 12 gives a thought of how to compute Secure_Maximum. We emphasis that this algorithm could be paralleled performed such that it should be more efficient under more servers settings.

At the end, the maximum value is known by each user. The user with the maximum energy consumption reduces his or her power consumption. As stated in the existing work,

the amount by which the energy consumption needs to be reduced is hard to decide. Thus, the second strategy is more practical.

6.1.4. 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. Deciding a reasonable power reduction amount for everybody is really important. The function from the prior work, which is shown in Equation 1 is adopted here. Using this equation, every user A_i will reduce at least δ_i power, which is decided by a_i and weighted in a. Since party A_i has its own power consumption value a_i, $\frac{t}{a}$ needs to be calculated at the servers.

To securely compute $\frac{t}{a}$, two secure division protocols that use additive homomorphic encryption schemes are introduced in [14, 83]. However, 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 this problem domain. In particular, when both t and a are hidden, computing division between two encrypted values under homomorphic encryption is very expensive. In the garbled circuit approach, the secret shares of t and a can be directly used as inputs to the circuits.

The division circuit was built based on the "shift and subtract" non-restoring method. Algorithm 13 gives a detail of this method. In general, to calculate the quotient of l-bit number t and m-bit number a, one must first expand t with $m+1$ bits and perform an iterative algorithm. In each loop, t makes a left shift and a is subtracted or added 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 is performed. Otherwise, addition is performed. After m rounds, the latest t_0 to t_{l-1} store the quotient q.

An example of how this method works is provided. To calculate the quotient of 11 (e.g., 1011 in binary format) divided by 3 (i.e., 0011 in binary format), first one must expand 1011 to 000001011 and shift this number to the left. Then, $00001011x_1$ is obtained using the left most $l+m-1=5$ bits to subtract 0011. The result is $11110011x_1$. Now the first bit is 1, so one would set $x_1=0$ and shift left again. Then, the most 5 bits of

Algorithm 13 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 $\overline{t_{l+m-1} \ldots t_l}$ with a

(c) **else** add $\overline{t_{l+m-1} \ldots t_l}$ with a

2: $q \leftarrow \overline{t_{l-1} \ldots t_0}$

Algorithm 14 OP-PUC-Stage-3$(a_i, t', t'', a', a'') \rightarrow \delta_i$

Require: S_1 has a' and t', S_2 has a'' and t'', P_i has a_i for $1 \leq i \leq n$, N is public

1: S_1 and S_2:

(a) **do** Secure_Division$(t', t'', a', a'') \rightarrow (q', q'')$

(b) Send q' and q'' to every power users

2: A_i:

(a) Calculate $\delta_i = a_i * (1 - \frac{t}{a})$ and reduce at least δ_i power usages

$11100110x_2$ are used to add 0011 since $x_1 = 0$. The result is $11111110x_2$, and one would set $x_2 = 0$. One would then shift and add again for $x_2 = 0$. This round results in $00010100x_3$, $x_3 = 1$ because 0 is the most significant bit. For next and last round, one must shift and subtract. The final result is $00010001x_4$, $x_4 = 1$. Thus, the quotient of this example is 3 (i.e., 0011 in binary format).

The presented garbled division circuit follows the basic rules of the "shift and subtract" non-restoring method, and it is denoted by Secure_Division$(t', t'', a', a'') \rightarrow (q', q'')$. 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. In the end, every power user will get $q = q' + q'' \mod N$ so as to compute δ_i using equation 1. Algorithm 14 summarizes the main steps of the third stage of the OP-PUC protocol.

6.1.5. Complexity Analysis. In this section both computation and communication complexities of the proposed OP-PUC protocol are analyzed. First, the computation complexity for different sub-protocols is analyzed at each stage. At the first stage, each user A_i and C perform 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, one must consider the secure comparison protocol. For the garbled circuit approach, the inputs are two random shares with the 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 results in much smaller values than N, so the total number of gates for the comparison circuit is bounded by $O(\log N)$.

Protocols for the two strategies become different at the third stage. For Strategy 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 the initial stage is also bound by $O(\log N)$. Since the bit lengths of the dividend and divisor are much less than N, the computation complexity of the division circuit is also bound 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, one must know the size of the secret shares. Since the value of each share is bounded by N, $\log N$ bits are needed 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

the strategies, the total communication complexity of OP-PUC is bounded by $O(n \cdot \log N)$ bits.

6.1.6. Security Analysis. The security proof of the proposed protocols is straightforward. Only a high level discussion is provided here. In the second and third stages, comparison and division of the garbled circuit, which is proved secure under the semi-honest model [82], are used. Since all the intermediate outputs of these protocols are random shares, based on the sequential composition theorem [84], the OP-PUC protocols are also secure under the semi-honest model.

6.2. EXPERIMENTAL RESULTS

This section contains a discussion of the performance of the OP-PUC protocols in detail under different parameter settings. Then, the computation costs of the existing methods [14] are evaluated and compared with the proposed protocols.

In the OP-PUC protocols, each A_i and C 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 is 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, the communication complexity is ignored here. The computation complexity is simulated 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, the protocol is implemented on top of a FastGC [82], a Java-based framework that allows users to define Boolean circuits. After the circuits are constructed, the framework encrypts the circuits, performs an oblivious transfer, and evaluates the garbled/encrypted circuits. The size of inputs is fixed for cloud servers to a 1024-bit modulus. In these experiments, the values of a_i's and t are randomly generated

such that $a > t$ and $1 \leq a, t \leq 2^m$, where m is an upper bound bit length of the domain size.

6.2.1. Performance of OP-PUC and OP-PUC$_2$. Let OP-PUC$_2$ denote the proposed protocol based on the second strategy. The computation costs of different parties involved in the OP-PUC protocol are first computed for $n = 50$ and varying values of 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). The costs of individual users and the power company are not considered since almost all the computations are outsourced to the cloud servers. As shown in Figure 6.1., the computation costs of A and B are 6.043 and 7.767 seconds, respectively, for $m = 10$. Although the computation times of A and B increase with m, the portion of increase 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 seconds, respectively, and they are pretty close to what they were when $m = 10$. This is due to the inner structure of the 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.

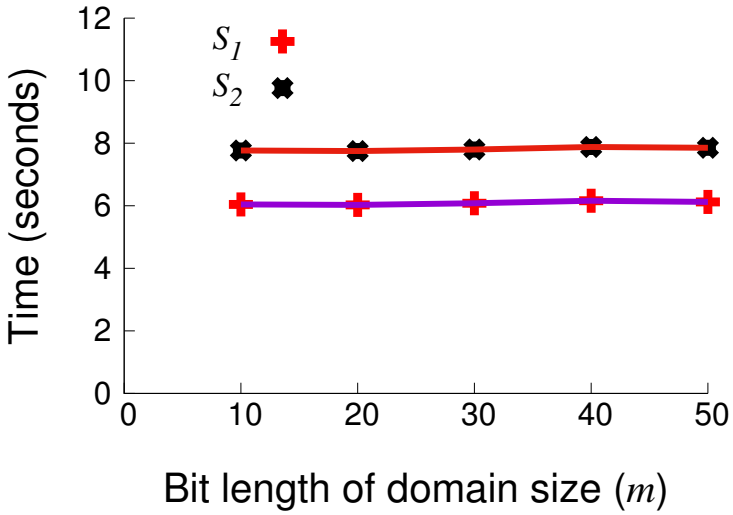


Figure 6.1. Complexity of OP_PUC for $n = 50$

In a similar manner, the computation costs of A and B in the OP_PUC^2 protocol are analyzed for varying values of m and with $n = 50$ and $\theta = 10$, where θ 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 a scalar factor is needed to come up with a 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 Figure 6.2.. As in 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 seconds, respectively. Similarly, the costs of A and B grow slightly with the increase of m. For instance, the computation time of A is 2.688 seconds when $m = 50$, and it is only increased by 0.191 seconds with a 40-bit size expansion.

The total computation costs of cloud providers A and B are now compared in OP_PUC (for one iteration) and OP_PUC^2 for $m = 10$ and varying values of n, where n denotes the number of households from a given neighborhood. As shown in Figure 6.3., the total running time of OP_PUC varies from 13.81 to 62.635 seconds when n is changed from 50 to 250. On the other hand, the total running time of OP_PUC^2 remains nearly constant at 6.692 seconds in average since t is independent of n. Following from Figure 6.3., 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, that the proposed protocols are very practical, especially for OP_PUC^2. Besides, there is nearly no computation cost for the individual users and the utility company.

6.2.2. Performance Comparison with Existing Work. Finally, the computation costs of these protocols are compared with the existing work [14]. For $n = 50$ and $m = 10$ (note that when $m = 10$, the domain size is $2^{10} = 1024$, which is already slightly bigger than $l = 1000$ from the previous paper), the performance of OP_PUC is close to P-PUC$_1^*$, which is roughly 13-15 seconds. 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 the 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 increases just less than 1 second). Note that in P-PUC$_1^*$, the execution time is significantly increased with the increase of the domain size. Experiments showed

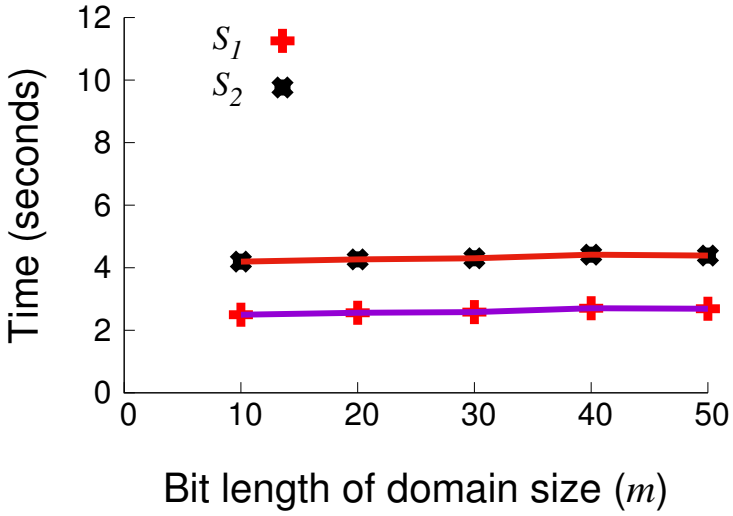


Figure 6.2. Complexity of OP_PUC^2 for $n = 50$

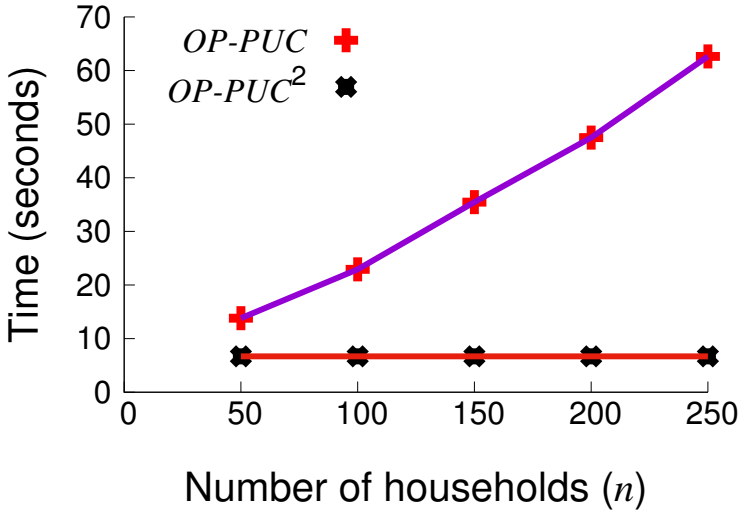


Figure 6.3. Complexity: OP_PUC Vs. OP_PUC^2

that when the domain size changes from 1024 to 4096, and the number of neighborhood is fixed to 50 users, the running time of P-PUC$_1^*$ increases from 11.02 seconds to 33.75 seconds. Also, OP_PUC^2 is more efficient and scalable than P-PUC$_2^*$. For example, when the domain size is 5000, OP_PUC^2 is faster than P-PUC$_2^*$ by a factor of 3 to 4. Although the problem definition of this work is different from the existing work, these protocols achieve the same power usage control in a more efficient way.

7. PRIVACY-PRESERVING POWER SUPPLY CONTROL

In this chapter, solutions for another situation – power shortage – is considered. When the regional power demand is greater than power supply ability of the utility company, then a power shortage situation is going to happen. Interestingly, threshold power usage control protocols can be applied for this situation, however, it needs the end users to sign an agreement with utility company for smartly power control anywhere anytime. If end users insist consuming much when regional power consumption level is high, the utility company needs to buy additional power supplies from other energy sources to meet the high demand.

1) Electricity Market

Just as other traditional commodities, electricity is also bought and re-sold many times before being consumed. These transactions are composing the wholesale electricity market. Anyone who obtaining necessary approvals, even without owning any generation or serving any end-use customers, can participant the power market in buying and selling powers [86]. An auction process can be used to decide the electricity price in which electricity suppliers place bids with an independent market administrator or a utility company for a particular time period. The independent administrator then chooses the lowest bids and tries to meet the electricity demand [87].

2) Second Price Auction

Here we consider an privacy-preserving auction process that helps the electricity market administrator to choose the best price properly, without loss of bidders' (or suppliers') privacy. Under our problem domain, we consider the price from each supplier is private information. For one, it is in the best interest of the suppliers not to disclose their prices publicly to lose competitive advantages. In addition, for the auction process to be performed properly and fairly, the prices should not be disclosed to the buyers or the utility company under our context. Specially, the auction is a sealed bid second price auction or

Vickrey auction [88]. In this type of auction the lowest bider wins with paying the second lowest bid. We choose this auction scheme because it has an important property that the optimal strategy of bidders is to simply bid their true valuation of the goods [89, 90]. At the same time, the sealed-bid second price auction requires less interaction and easier to run automatically in a smart grid.

The auctioneer is normally treated as trusted party in an auction process. However, this is not always true. A corrupt auctioneer may take advantage of bidders' information. To solve this problem, the value of each bid needs to be hidden from auctioneer. Let us consider the ideal model: the auctioneer opens the auction and bidders take part in it. Then there is a trusted third party help the auctioneer to decide which bidder wins, and tell him to pay the second lowest bid. During this process, only who wins and the second lowest price will be disclosed, nothing else will be learned by the auctioneer. This chapter proposes a scheme that reaches a solution to this problem with two cloud servers, and do not need a trusted third party.

7.1. PROBLEM DEFINITION FOR PRIVACY-PRESERVING POWER SUPPLY CONTROL: OP-PSC

Suppose there are n energy suppliers $E_1, \ldots, E_n$. For $1 \leq i \leq n$, let e_i denote the price or bid under which E_i is willing to sell to the utility company. The proposed outsourceable privacy-preserving power usage control (OP-PSC) protocol can be formulated as follows:

$$\langle U, (E_*, e_*) \rangle \leftarrow \text{OP-PSC}(\langle E_i, e_i \rangle_{1 \leq i \leq n}, S_1, S_2, U) \tag{4}$$

According to the above formulation, there are three types of participating entities: n energy companies or suppliers, two cloud service providers S_1 and S_2, and the utility company U. The input for each supplier E_i is its bid price e_i. The two cloud servers perform the necessary computations, and there are no explicit inputs for the two servers. After the execution of the OP-PSC protocol, the utility company knows who won the auction and the price, and the utility company passes this information to the corresponding supplier. The other participating entities do not receive any outputs.

7.1.1. Threat Model. In the chapter, we adopt secure multiparty computation (SMC) again. 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-PSC 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.

- 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.

7.1.2. Our Contribution. The contribution in this chapter is to develop a novel OP-PSC protocol that allows a utility company to purchase additional power from other energy companies in a privacy-preserving manner. The OP-PSC protocol is a secure implementation of the second price auction model.

The rest of the chapter is organized as follows: Section 7.2. provides implementation details regarding the proposed OP-PSC protocol. Section 7.3. presents empirical results to show the practicality of OP-PSC protocols.

7.2. THE PROPOSED OP-PSC PROTOCOL

In our problem domain, the utility company and n power suppliers $E_1,\ldots,E_n$ engage in a second price sealed-bid process. After the utility company starts the process, each power supplier E_i submits a bid e_i for $1 \leq i \leq n$, respectively, showing how much he wants

Algorithm 15 OP-PSC($\langle E_1, e_1\rangle, \ldots, \langle E_n, e_n\rangle, S_1, S_2, U) \rightarrow \langle U, (E_*, e_*)\rangle$

Require: pk is known to all parties and S_2 has pr

1: E_i $(1 \leq i \leq n)$:
   (a) $[e_i] \leftarrow E_{pk}(e_i^0), \ldots, E_{pk}(e_i^m)$
   (b) Send $\langle E_i, [e_i]\rangle$ to S_1

2: S_1 and S_2:
   (a) $\langle S_1, (E_j, e_j)\rangle \leftarrow$SMIN$_n(\langle E_1, [e_1]\rangle, \ldots, \langle E_n, [e_n]\rangle)$, where $e_j = \min(e_1, \ldots, e_n)$
   (b) $\langle S_1, (E_*, e_*)\rangle \leftarrow$SMIN$_{n-1}(\langle E_1, [e_1]\rangle, \ldots, \langle E_{j-1}, [e_{j-1}]\rangle, \langle E_{j+1}, [e_{j+1}]\rangle, \ldots,$ $\langle E_n, [e_n]\rangle)$, where $e_* = \min(e_1, \ldots, e_{j-1}, e_{j+1}, \ldots, e_n)$

3: S_1: send $\langle E_*, [e_*]\rangle$ to U and E_*

for selling a unit power. After all, the power supplier that gave the lowest bid wins and is paid by the second lowest price. Besides, we want the whole process secure, which is to say, the utility company will only know the winner and the price he is going to be paid, and the bidder will learn nothing about the others' bids except that the winner learns the second lowest price. Also, this protocol is build in an outsourced environment. Two non-collude cloud servers S_1 and S_2 jointly perform the needed computations.

The detailed steps are given in Algorithm 15. In the first stage, the cloud servers need to collect price information. We emphasize that during this process, the servers should learn nothing about the price. To achieve this requirement, the server S_2 firstly publishes the homomorphic encryption public key pk [65], and each bidder E_i encrypts his bidding price e_i bit-wisely to get $[e_i] \leftarrow E_{pk}(e_i^0), \ldots, E_{pk}(e_i^m)$. Note that the bit length of e_i is bounded by m. Then E_i sends those encrypted values to S_1.

In the next stage, S_1 and S_2 jointly decide who is the winner and the price he will be paid. First, S_1 and S_2 engage in a secure minimum among n items protocol (SMIN$_n$) introduced in [91] for the n encrypted prices and ID pairs $\langle E_1, e_1\rangle, \ldots, \langle E_n, e_n\rangle$. The result is that S_1 learns the lowest price with its bidder ID $\langle E_j, e_j\rangle$. Since those values are under encrypted format, S_1 only knows the lowest price. After that, S_1 removes this tuple from $\langle E_1, e_1\rangle, \ldots, \langle E_n, e_n\rangle$ and performs SMIN$_{n-1}$ jointly with S_2 again for the remaining data pairs. This time S_1 will learn $\langle E_*, e_*\rangle$ which will be delivered to U.

The complexity and security of OP-PSC is determined by the SMIN_n protocol. Detailed complexity and security analyses can be found in [91].

7.3. EXPERIMENTAL RESULTS

In this section, we discuss the performance of the OP-PSC protocols in details under different parameter settings. 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. We implement the protocol based on Pailliar's additive homomorphic encryption scheme using C and GMP library. We also fix the encryption key size of inputs for cloud servers to a 1024-bit. In our experiments, we randomly generate the values of inputs prices.

In the OP-PSC protocol, the main component and performance bottleneck is the SMIN_n protocol, which will return the minimum value among n values, under encrypted form. In all, the SMIN_n protocol that we adopted here will be executed twice and find the minimum and second minimum values from the input group. We implement the protocol using C and GMP library based on Pailier's additive homomorphic encryption scheme [65]. We also follow the same rules as before: encryption key size is fixed to 1024 bit. The input bid values are generated randomly.

In our experiments, we mainly compare the performance of S_1 and S_2, since the computation overhead for other participating parties are negligible. In Figure 7.1., we fix the number of bidders or suppliers $n = 10$ and compare the running time for various bit length of domain size m from 10 to 50. The time linearly increases with the increasing of bit length m. This is because in the protocol, the number of exponentiation functions that needs to be performed is based on m. Figure 7.2. gives us another comparison between S_1 and S_2 when m is fixed to 10 and n is varied. The range of n is between 10 and 50. The running time is expanded from $\langle 3.97, 0.88 \rangle$ to $\langle 17.15, 3.86 \rangle$ for $\langle S_1, S_2 \rangle$, respectively.

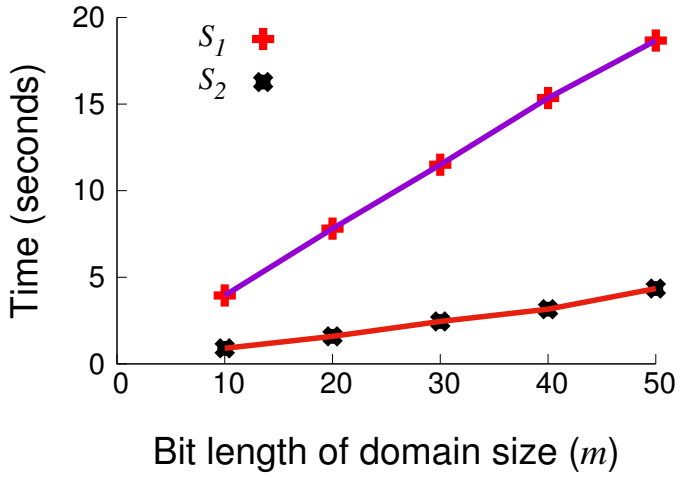


Figure 7.1. Complexity of OP-PSC for $n = 10$

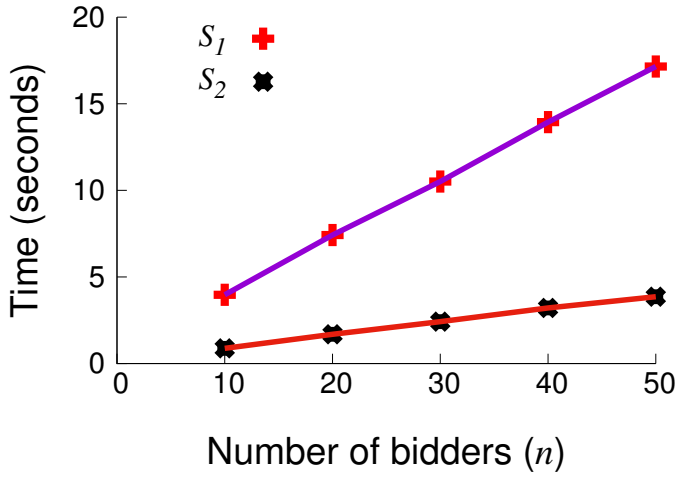


Figure 7.2. Complexity of OP-PSC for $m = 10$

The trend is also linear-like since the sub-protocols of SMIN_n are composed of $n - 1$ times secure minimum between two values. Overall, the running time of the protocol is acceptable: when the bit length of the domain size becomes 50, the total running time for one cloud server is still less than 20 seconds.

8. CONCLUSIONS

The smart grid system has been emerging as the next-generation's intelligent power system. In particular, the utility company in a smart grid environment can periodically record the power usage of household to efficiently manage its power supply to the households in a neighborhood. The power consumption data is useful in preventing power outage issues in the given neighborhood. However, due to the user's privacy concerns, direct collection of the user's power consumption data by the utility company is not allowed at first place. Along this direction, this dissertation proposes four novel and secure threshold-based power usage control protocols using two strategies discussed in Section 1.1. as a baseline.

At first, two efficient P-PUC protocols are proposed in Chapter 4.. However, since the adopted building blocks are not fully secure, some information will be disclosed during the execution of these protocols. Therefore, more secure and efficient protocols are proposed in Chapter 5.. Efficient and novel solutions to the basic security primitives, namely secure binary conversion, comparison, maximum, and division operations, are also constructed. We emphasize that these new sets of sub-protocols can also be used in other secure multi-party computation (MPC) based applications, such as secure clustering and classification. Furthermore, we have empirically shown the practical applicability of the proposed protocols through various experimental results.

As the emerging of cloud computing with its various advantages, protocols which is outsourceable to the cloud, are developed in Chapter 6. and 7.. On one hand, comparing with the previous four solutions, the proposed protocols in Chapter 6. are more efficient and as secure. On the other hand, one novel solution for power shortage is also proposed in Chapter 7.. More importantly, the computation costs for the users and the utility company are negligible. As a future research direction, we will develop OP-PUC and OP-PSC 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 efficient of the OP-PUC and OP-PSC protocols can be improved under the secret sharing model. To develop OP-PUC and OP-PSC protocols secure under the malicious model, we may adopt threshold homomorphic encryption [92] or Shamir secret sharing [93]. We will investigate the pros and cons under each direction.

BIBLIOGRAPHY

[1] T Baumeister. Literature review on smart grid cyber security, December 2010.

[2] C. Shuyong, S. Shufang, L. Lanxin, and S. Jie. Survey on smart grid technology. *Power System Technology*, 33:1–7, 2009.

[3] R. Hassan and G. Radman. Survey on smart grid. *IEEE SoutheastCon*, pages 210–213, 2010.

[4] G. F. Reed, P. A. Philip, A. Barchowsky, C. J. Lippert, and A. R. Sparacino. Sample survey of smart grid approaches and technology gap analysis. *IEEE Energy*, pages 1–10, 2010.

[5] X. Fang, S. Misra, G. Xue, and D. Yang. Smart grid - the new and improved power grid: A survey. *IEEE Communications Surveys and Tutorials*, 2011.

[6] NIST 7628. Guidelines for smart grid cyber security the smart grid interoperability panel – cyber security working group, August 2010.

[7] He H. Haibo. Toward a smart grid: Integration of computational intelligence into power grid. *International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, July 2010.

[8] P.P. Parikh, M.G. Kanabar, and T.S. Sidhu. Opportunities and challenges of wireless communication technologies for smart grid applications. In *Power and Energy Society General Meeting*, pages 1–7. IEEE, 2010.

[9] V.C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G.P. Hancke. Smart grid technologies: Communication technologies and standards. *IEEE Transactions on Industrial Informatics*, 7(4):529–539, 2011.

[10] Shahram S. Heydari, Walid Rjaibi, Khalil El-Khatib, and Julie Thorpe. Privacy and security of smart grid communication. In *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '11, pages 345–346, Riverton, NJ, USA, 2011. IBM Corp.

[11] E. L. Quinn. Smart metering & privacy: Existing law and competing policies., 2009.

[12] E. L. Quinn. Privacy and the new energy infrastructure, 2009.

[13] Chun Hu, Wei Jiang, and Bruce McMillin. Privacy-preserving power usage control in the smart grid. In *Critical Infrastructure Protection VI*, volume 390 of *IFIP Advances in Information and Communication Technology*, pages 127–137. Springer Berlin Heidelberg, 2012.

[14] Bharath K. Samanthula, Hu Chun, Wei Jiang, and Bruce M. McMillin. Secure and threshold-based power usage control in smart grid environments. *International Journal of Parallel, Emergent and Distributed Systems*, 29(3):264–289, 2014.

[15] Hu Chun, Kui Ren, and Wei Jiang. Outsourceable privacy-preserving power usage control in a smart grid. In *Data and Applications Security and Privacy XXIX*, pages 119–134. Springer, 2015.

[16] Iordanis Koutsopoulos and Leandros Tassiulas. Control and optimization meet the smart power grid - scheduling of power demands for optimal energy management. *CoRR*, abs/1008.3614, 2010.

[17] H. Wang, J. Huang, X. Lin, and H. Mohsenian-Rad. Exploring smart grid and data center interactions for electrical power load balancing. In *Greenmetrics workshop held in conjunction with ACM SIGMETRICS*, Pittsburgh, PA, 2013.

[18] P. Samadi, H. Mohsenian-Rad, V.W.S. Wong, and R. Schober. Tackling the load uncertainty challenges for energy consumption scheduling in smart grid. *IEEE Transactions on Smart Grid*, 4(2):1007–1016, 2013.

[19] Ishtiaq Rouf, Hossen Mustafa, Miao Xu, Wenyuan Xu, Rob Miller, and Marco Gruteser. Neighborhood watch: security and privacy analysis of automatic meter reading systems. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 462–473, New York, NY, USA, 2012. ACM.

[20] Y. Yang, K. McLaughlin, T. Littler, S. Sezer, Eul Gyu Im, Z.Q. Yao, B. Pranggono, and H.F. Wang. Man-in-the-middle attack test-bed investigating cyber-security vulnerabilities in smart grid scada systems. In *Sustainable Power Generation and Supply (SUPERGEN 2012), International Conference on*, pages 1–8, Sept 2012.

[21] Steven Drenker and Ab Kader. Nonintrusive monitoring of electric loads. *Computer Applications in Power, IEEE*, 12(4):47–51, 1999.

[22] Mikhail A. Lisovich, Deirdre K. Mulligan, and Stephen B. Wicker. Inferring personal information from demand-response systems. *IEEE Security and Privacy*, 8(1):11–20, January 2010.

[23] Andrés Molina-Markham, Prashant Shenoy, Kevin Fu, Emmanuel Cecchet, and David Irwin. Private memoirs of a smart meter. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pages 61–66, New York, NY, USA, 2010. ACM.

[24] Wenye Wang and Zhuo Lu. Cyber security in the smart grid: Survey and challenges. *Computer Networks*, 57(5):1344–1371, 2013.

[25] Ye Yan, Yi Qian, Hamid Sharif, and David Tipper. A survey on cyber security for smart grid communications. *Communications Surveys & Tutorials, IEEE*, 14(4):998–1010, 2012.

[26] M.M. Fouda, Z.M. Fadlullah, N. Kato, Rongxing Lu, and Xuemin Shen. A lightweight message authentication scheme for smart grid communications. *Smart Grid, IEEE Transactions on*, 2(4):675–685, Dec 2011.

[27] Qinghua Li and Guohong Cao. Multicast authentication in the smart grid with one-time signature. *Smart Grid, IEEE Transactions on*, 2(4):686–696, Dec 2011.

[28] Chakib Bekara, Thomas Luckenbach, and Kheira Bekara. A privacy preserving and secure authentication protocol for the advanced metering infrastructure with non-repudiation service. In *ENERGY 2012, The Second International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, pages 60–68, 2012.

[29] M. Nabeel, S. Kerr, Xiaoyu Ding, and E. Bertino. Authentication and key management for advanced metering infrastructures utilizing physically unclonable functions. In *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*, pages 324–329, Nov 2012.

[30] Hasen Nicanfar and Victor CM Leung. Multilayer consensus ecc-based password authenticated key-exchange (mcepak) protocol for smart grid system. *Smart Grid, IEEE Transactions on*, 4(1):253–264, 2013.

[31] F. Diao, F. Zhang, and X. Cheng. A privacy-preserving smart metering scheme using linkable anonymous credential. *Smart Grid, IEEE Transactions on*, 6(1):461–467, Jan 2015.

[32] Jing Liu, Yang Xiao, Shuhui Li, Wei Liang, and C. L. Philip Chen. Cyber security and privacy issues in smart grids. *Communications Surveys Tutorials, IEEE*, 14(4):981–997, 2012.

[33] Paria Jokar, Nasim Arianpoo, and Victor C. M. Leung. A survey on security issues in smart grids. *Security and Communication Networks*, 2012.

[34] F. Borges and M. Muhlhauser. Eppp4sms: Efficient privacy-preserving protocol for smart metering systems and its simulation using real-world data. *Smart Grid, IEEE Transactions on*, 5(6):2701–2708, Nov 2014.

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

[36] Fengjun Li and Bo Luo. Preserving data integrity for smart grid data aggregation. In *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*, pages 366–371. IEEE, 2012.

[37] Sushmita Ruj and Amiya Nayak. A decentralized security framework for data aggregation and access control in smart grids. *IEEE transactions on smart grid*, 4(1):196–205, 2013.

[38] Felix Gomez Marmol, Christoph Sorge, Osman Ugus, and Gregorio Martínez Pérez. Do not snoop my habits: preserving privacy in the smart grid. *Communications Magazine, IEEE*, 50(5):166–172, 2012.

[39] Zekeriya Erkin and Gene Tsudik. Private computation of spatial and temporal power consumption with smart meters. In *Applied Cryptography and Network Security*, pages 561–577. Springer, 2012.

[40] Rongxing Lu, Xiaohui Liang, Xu Li, Xiaodong Lin, and Xuemin Shen. Eppa: An efficient and privacy-preserving aggregation scheme for secure smart grid communications. *IEEE Transactions on Parallel and Distributed Systems*, 23(9):1621–1631, 2012.

[41] Marc Joye and Benoît Libert. A scalable scheme for privacy-preserving aggregation of time-series data. In *Financial Cryptography and Data Security*, pages 111–125. Springer, 2013.

[42] Elaine Shi, T-H Hubert Chan, Eleanor G Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *NDSS*, volume 2, page 4, 2011.

[43] Lei Yang, Xu Chen, Junshan Zhang, and H Vincent Poor. Cost-effective and privacy-preserving energy management for smart meters.

[44] Stephen McLaughlin, Patrick McDaniel, and William Aiello. Protecting consumer privacy from electric load monitoring. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 87–98. ACM, 2011.

[45] Georgios Kalogridis, Costas Efthymiou, Stojan Z Denic, Tim A Lewis, and Rafael Cepeda. Privacy for smart meters: Towards undetectable appliance load signatures. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 232–237. IEEE, 2010.

[46] Jinkyu Koo, Xiaojun Lin, and Saurabh Bagchi. Privatus: Wallet-friendly privacy protection for smart meters. In *Computer Security–ESORICS 2012*, pages 343–360. Springer, 2012.

[47] Onur Tan, Deniz Gunduz, and H Vincent Poor. Increasing smart meter privacy through energy harvesting and storage devices. *Selected Areas in Communications, IEEE Journal on*, 31(7):1331–1341, 2013.

[48] Zhi Chen and Lei Wu. Residential appliance dr energy management with electric privacy protection by online stochastic optimization. 2013.

[49] Weining Yang, Ninghui Li, Yuan Qi, Wahbeh Qardaji, Stephen McLaughlin, and Patrick McDaniel. Minimizing private data disclosures in the smart grid. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 415–427. ACM, 2012.

[50] Costas Efthymiou and Georgios Kalogridis. Smart grid privacy via anonymization of smart metering data. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 238–243. IEEE, 2010.

[51] Marek Jawurek, Martin Johns, and Konrad Rieck. Smart metering de-pseudonymization. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 227–236. ACM, 2011.

[52] Younghun Kim, EC Ngai, and Mani B Srivastava. Cooperative state estimation for preserving privacy of user behaviors in smart grid. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pages 178–183. IEEE, 2011.

[53] Allen J Wood and Bruce F Wollenberg. *Power generation, operation, and control*. John Wiley & Sons, 2012.

[54] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara. Private collaborative forecasting and benchmarking. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, WPES '04, pages 103–114. ACM, October 2004.

[55] Andrew C. Yao. Protocols for secure computations. In *the Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE Computer Society, 1982.

[56] Andrew C. Yao. How to generate and exchange secrets. In *the Annual Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society, 1986.

[57] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game - a completeness theorem for protocols with honest majority. In *19th ACM Symposium on the Theory of Computing*, pages 218–229, New York, New York, United States, 1987. ACM.

[58] D. Beaver. Foundations of secure interactive computing. In *Advances in Cryptology - CRYPTO '91*, pages 377–391. Springer-Verlag, 1991.

[59] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *IEEE FOCS*, pages 136 – 145. IEEE Computer Society, oct. 2001.

[60] Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: a system for secure multi-party computation. In *Proceedings of the ACM Computer and Communications Security Conference (ACM CCS)*, pages 257–266. ACM, 2008.

[61] Yehuda Lindell. General composition and universal composability in secure multiparty computation. *Journal of Cryptology*, 22(3):395–428, 2009.

[62] Wilko Henecka, Stefan K ögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Tasty: tool for automating secure two-party computations. In *ACM CCS*, pages 451–462. ACM, 2010.

[63] Dan Bogdanov, Roman Jagomägis, and Sven Laur. A universal toolkit for cryptographically secure privacy-preserving data mining. In *PAISI '12*, pages 112–126. Springer-Verlag, 2012.

[64] Oded Goldreich. *The Foundations of Cryptography*, volume 2, chapter General Cryptographic Protocols. Cambridge University Press, 2004.

[65] P. Paillier. Public key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - Eurocrypt '99 Proceedings, LNCS 1592*, pages 223–238. Springer-Verlag, 1999.

[66] Wei Jiang, Mummoorthy Murugesan, Chris Clifton, and Luo Si. Similar document detection with limited information disclosure. In *Proceedings of the 24th International Conference on Data Engineering (ICDE 2008)*, Cancun, Mexico, April 7-12 2008. IEEE Computer Society.

[67] Shafi Goldwasser, Silvio Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In *ACM STOC*, pages 291–304. ACM, 1985.

[68] Oded Goldreich. *The Foundations of Cryptography*, volume 2, chapter Encryption Schemes. Cambridge University Press, 2004.

[69] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986.

[70] Yehuda Lindell and Benny Pinkas. A proof of security of yaos protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[71] Michael O Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 2005.

[72] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[73] William Melicher, Samee Zahur, and David Evans. An intermediate language for garbled circuits. In *IEEE Symposium on Security and Privacy Poster Abstract*, 2012.

[74] Li Xiong, Subramanyam Chitti, and Ling Liu. Topk queries across multiple private databases. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 145–154. IEEE, 2005.

[75] Paul Bunn and Rafail Ostrovsky. Secure two-party k-means clustering. In *ACM CCS*, pages 486–497. ACM, 2007.

[76] U.S. Energy Information Administration. `http://www.eia.gov/tools/faqs/faq.cfm?id=97&t=3`.

[77] Ian F Blake and Vladimir Kolesnikov. One-round secure comparison of integers. *Journal of Mathematical Cryptology*, 3(1):37–68, 2009.

[78] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. Efficient and secure comparison for on-line auctions. In *Information Security and Privacy*, pages 416–430. Springer, 2007.

[79] Ivan Damgard, Martin Geisler, and Mikkel Kroigard. Homomorphic encryption and secure comparison. *International Journal of Applied Cryptography*, 1(1):22–31, 2008.

[80] Juan Garay, Berry Schoenmakers, and José Villegas. Practical and secure solutions for integer comparison. In *Public Key Cryptography–PKC 2007*, pages 330–342. Springer, 2007.

[81] Ahmet Erhan Nergiz, Mehmet Ercan Nergiz, Thomas Pedersen, and Chris Clifton. Practical and secure integer comparison and interval check. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 791–799. IEEE, 2010.

[82] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*, volume 201, 2011.

[83] Paul Bunn and Rafail Ostrovsky. Secure two-party k-means clustering. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 486–497. ACM, 2007.

[84] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*, volume 2. Cambridge university press, 2009.

[85] Chun Hu, Wei Jiang, and Bruce McMillin. Privacy-preserving power usage control in the smart grid. In Jonathan Butts and Sujeet Shenoi, editors, *Critical Infrastructure Protection VI*, volume 390 of *IFIP Advances in Information and Communication Technology*, pages 127–137. Springer Berlin Heidelberg, 2012.

[86] Phillip G Harris. The value of independent regional grid operators. *The Electricity Journal*, 5(19):82–85, 2006.

[87] Electric Power Supply Association. `https://www.epsa.org/industry/primer/?fa=wholesaleMarket`.

[88] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

[89] Felix Brandt and Tuomas Sandholm. Efficient privacy-preserving protocols for multi-unit auctions. In *Financial Cryptography and Data Security*, pages 298–312. Springer, 2005.

[90] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 129–139. ACM, 1999.

[91] Yousef Elmehdwi, Bharath K Samanthula, and Wei Jiang. Secure k-nearest neighbor query over encrypted data in outsourced environments. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 664–675. IEEE, 2014.

[92] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology – EUROCRYPT*, pages 280–299, 2001.

[93] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612 – 613, November 1979.

VITA

Huchun received the bachelor's degree in Software Engineering from Nankai University, Tianjin, P.R. China, in 2010. He received the Ph.D. degree in Computer Science from Missouri University of Science and Technology, Rolla, Missouri, in December 2015.