Privacy-Preserving Energy Theft Detection in Smart Grids: A P2P Computing Approach

Sergio Salinas, *Student Member, IEEE,* Ming Li, *Student Member, IEEE,* and Pan Li, *Member, IEEE*

***Abstract*—In the U.S., energy theft causes about six billion dollar losses to utility companies (UCs) every year. With the smart grid being proposed to modernize current power grids, energy theft may become an even more serious problem since the "smart meters" used in smart grids are vulnerable to more types of attacks compared to traditional mechanical meters. Therefore, it is important to develop efficient and reliable methods to identify illegal users who are committing energy theft. Although some schemes have been proposed for the UCs to detect energy theft in power grids, they all require users to send their private information, e.g., load profiles or meter readings at certain times, to the UCs, which invades users' privacy and raises serious concerns about privacy, safety, etc. To the best of our knowledge, we are the first to investigate the energy theft detection problem considering users' privacy issues. Specifically, in this paper, utilizing peer-to-peer (P2P) computing, we propose three distributed algorithms to solve a linear system of equations (LSE) for users' "*honesty coefficients*". Extensive simulations are carried out and the results show that the proposed algorithms can efficiently and successfully identify the fraudulent users in the system.**

***Index Terms*—Energy theft detection, smart grids, privacy, P2P computing.**

I. Introduction

ENERGY theft has been a notorious problem in traditional power systems. The utility companies (UCs) in the U.S. lose approximately six billion dollars every year due to this problem [2]. Recently, the smart grid has been proposed as a new type of electrical grid to modernize current power grids to efficiently deliver reliable, economic, and sustainable electricity services. One of the most salient features of smart grids is the replacement of conventional analog mechanical meters by digital meters, usually called "smart meters". In addition to recording users' energy usage, due to their communication capability, smart meters can provide a two-way communication path between UCs and energy users, which can facilitate efficient power system control and monitoring. However, compared to mechanical meters which can only be physically tampered, smart meters are vulnerable to more types of attacks (e.g., network attack), which may make energy theft easier to commit and hence an even more serious problem in smart grids.

Some research has been conducted to detect energy theft in traditional power grids. Nizar et al. [3] employ a data mining technique known as Extreme Learning Machine (ELM) to classify users' electricity consumption patterns or load-profiles. By comparing the results to a database of users' load profiles, the proposed algorithm yields a list of users who could be stealing energy, which we call "energy thieves". Nagi et al. [4] propose a similar approach but choose to use genetic algorithms and Support Vector Machine (SVM) instead of ELM. Depuru et al. [5] develop another data mining based scheme utilizing SVM as well. Unfortunately, these techniques cannot sort out the energy thieves with absolute certainty. In contrast, Bandim et al. [6] propose a central observer to measure the total energy consumption of a small number of users, and are able to identify all the energy thieves by comparing the total energy consumption with the reported energy consumption from the users.

Notice that in all the above works, the UC has to know some of users' private information, e.g., users' load profiles or meter readings at certain times, in order to find the energy thieves. However, the disclosure of such information would violate users' privacy and raise concerns about privacy, safety, etc. In particular, users' private information may be sold to interested third-parties. Insurance companies may buy load-profiles from the UC to make premium adjustments on the users' policies. For example, they could find electricity consumption patterns that increase the risk of fire in a property and increase insurance premiums accordingly. Marketing companies may also be interested in this data to identify potential customers. Moreover, criminals may utilize such private information to commit crimes. For instance, robbers may analyze the energy consumption pattern of potential victims to deduce their daily behavior. They can even know if a robbery alarm has been set at their target location [7]. Many researchers, such as Quinn [8], have realized how high resolution electricity usage information can be used to reconstruct many intimate details of a consumer's daily life and invade his/her privacy, and thus call for state legislators and public utility commissions to address this new privacy threat [9].

Unfortunately, there is currently a lack of research on privacy-preserving energy theft detection in smart grids. Li et al. [10] design a privacy-preserving aggregation protocol to collect the total energy consumption of a group of users at a distribution station in smart grids, which shares a similar idea to those works like [11] on privacy-preserving data aggregation in wireless sensor networks. However, such algorithms cannot be used to detect energy theft in smart grids. To the best of our knowledge, we are the first to investigate the energy theft detection problem considering users' privacy issues.

In particular, intuitively and as in previous works, we

Manuscript received February 28, 2012; revised July 14, 2012.

This work was supported by the U.S. National Science Foundation under grants CNS-1149786 (CAREER Award), ECCS-1128768, and CNS-1147851. An earlier version of this paper was presented at IEEE SECON 2012 [1].

The authors are with the Department of Electrical and Computer Engineering, Mississippi State University, Mississippi State, MS, 39762 (e-mail: {sas573@, ml845@, li@ece.}msstate.edu).

Digital Object Identifier 10.1109/JSAC.2013.SUP.0513023

need to know about a user's electric energy consumption in order to tell whether he/she is committing fraud or not, which, however, results in the reveal of the user's privacy. Therefore, energy theft detection and users' privacy seem to be two conflicting problems. How to detect energy theft while preserving users' privacy is a challenging problem. In this paper, utilizing peer-to-peer (P2P) computing [12], we propose three distributed algorithms to solve a linear system of equations (LSE) for the users' "*honesty coefficients*". If a user's honesty coefficient is equal to 1, this user is honest. Otherwise, if the honesty coefficient is larger than 1, then this user has reported less consumed energy and hence is committing fraud. The users' privacy can be preserved because they do not need to disclose any of their energy consumption data to others.

More specifically, we propose to take advantage of distributed LU and QR decompositions to solve our LSE. Although some distributed algorithms for LU or QR decomposition [13] have been proposed in the literature, e.g., [14]–[17], they cannot preserve each node's private information. In this paper, we first develop a distributed privacy-preserving energy theft detection algorithm leveraging LU decomposition, called LUD. We find that LUD can successfully identify all the energy thieves in a small size network but may be unstable in large networks[1]. Then, we design another algorithm based on LU decomposition with partial pivoting, called LUDP, which can find all the energy thieves even in large-size networks. We also propose a third algorithm by QR decomposition, called QRD, which also works well in large-size networks. Moreover, the LUD, LUDP, and QRD algorithms are proposed in the case that users commit energy theft at a constant rate, i.e., with constant honesty coefficients. We further propose adaptive LUD/LUDP/QRD algorithms to account for the scenarios where the users have variable honesty coefficients.

In addition, after presenting the proposed algorithms, we analyze the computational and communication complexities of the two stable algorithms, i.e., LUDP and QRD. We find that LUDP algorithm has a computational complexity of $\Theta(2n^3/3)$ and a communication complexity of $\Theta(2n^3/3)$, and the QRD algorithm has a computational complexity of $\Theta(2n^3)$ and a communication complexity of $\Theta(5n^3/6)$. In other words, the QRD algorithm has higher computational complexity and higher communication complexity compared with the LUDP algorithm.

The rest of this paper is organized as follows. Section II introduces network model. Section III presents the linear system of equations for energy theft detection. Section IV details the proposed distributed algorithms for solving the LSE. Computational and communication complexity analysis is provided in Section V. Simulation results are shown in Section VI. Finally, we conclude this paper in Section VII.

II. Network Model

In this section, we first present the network architecture considered in this paper, and then briefly introduce the possible attacks on smart meters (SMs) by energy thieves, and possible implementations of the proposed algorithms on the SMs.

[1]This is due to the rounding errors in LU decomposition.

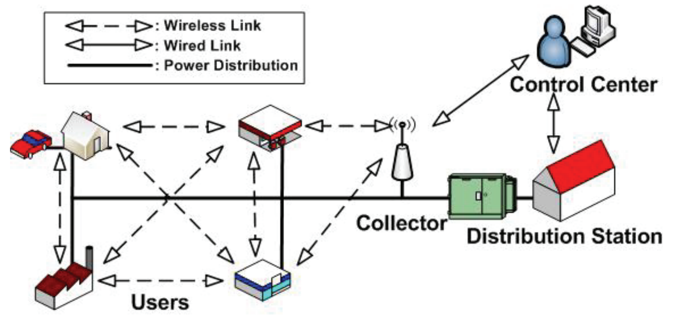


Fig. 1. A typical architecture of Neighborhood Area Network (NAN).

A. Network Architecture

In the smart grid, communications and electricity networks overlay each other. Utility companies (UCs) deploy control centers (CCs) to monitor their distribution stations (DSs) and distribution networks, and deploy SMs at users's premises to measure their individual energy consumption. Since a CC is usually physically far away from users, a communication entity that can facilitate the communication between users and the CC is necessary. To this end, an access point, called "*the collector*", is placed at each of the serviced areas. One SM is installed at each collector to measure the total energy consumed by the serviced area.

A typical network architecture is depicted in Fig. 1. In a serviced area, the users' SMs together with the collector form a Neighborhood Area Network (NAN). The communications among SMs and between SMs and the collector are carried out wirelessly due to SMs' communication capability, while the communications among the CC, the DS, and the collector are conducted via wired medium.

B. Attacks on Smart Meters

Smart meters can provide the users with a plethora of unique features. For example, users can be provided with real time electricity pricing and thus determine when to turn on/off some of their electrical devices. Smart meters can also send incentive-based load reduction signals to users so that they can be compensated for their efforts to save energy. However, compared to mechanical meters which can only be physically tampered, smart meters are vulnerable to more types of attacks, which may make energy theft easier to commit and hence an even more serious problem in smart grids.

1) Physical Attack: Conventional mechanical meters and SMs are both vulnerable to this type of attack. It refers to the scenarios where illegal users physically modify their meters to record wrong values that will lower their electric bills. Physical attack to electricity meters includes meter reversing, tampering with strong magnets, pressure coil damaging, supply voltage regulation, and even disconnecting the meters. The readers are referred to [18] for a more extensive description on physical attacks.

One way to detect physical attacks is to visually check the meter for any broken seals or other signs of damage. However, this detection method is both resource and time consuming because employees of the UC have to visit the users' premises to verify the meters' integrity. Moreover, signs of damage may not be obvious and seals may be replaced.

2) Network Attack: An illegal user can operate a malicious node to perform network attack. For example, an illegal user may impersonate his/her own SM and make it record lower energy consumption. Network attack may be easier to launch and more difficult to detect.

In addition to attacking his/her smart meter, a user may also get some energy that is not being measured, e.g., through a conductor that bypasses the meter. In this case, the smart meter does not correctly measure the energy consumption of the user and hence can also be considered as being attacked. The proposed algorithms can address all these problems.

C. Possible Implementation of the Proposed Algorithms

The proposed algorithms can be implemented in the firmware of the smart meters. Many mechanisms have been proposed to protect the firmware of embedded systems, such as passwords, centralized intrusion detection, local intrusion detection, and intrusion self-reporting. For example, LeMay et al. [19] develop a remote attestation mechanism that allows centralized intrusion detection of all SMs in a neighborhood. If any intrusion to the firmware is detected by the UC, the suspect SMs cannot be trusted and must be inspected. Consequently, SMs can be trusted in correctly executing the proposed privacy preserving energy theft detection algorithms[2].

III. A Linear System of Equations for Energy Theft Detection

In this section, we present a mathematical model for energy theft detection. As mentioned before, we assume that an SM is installed at the collector such that the collector can know the total energy consumption of the users in the service area. We also assume that the UC installs an SM at each of the users' premises, which is capable of recording energy consumption at any time instant.

Consider a NAN of n users. We define a *sampling period* denoted by SP. Then, after every sampling period, all the $n+1$ SMs will record their energy consumption in the past sampling period. We denote such energy consumption recorded by user j $(1 \leq j \leq n)$ and by the collector at time t_i, by $p_{t_i,j}$ and $\overline{\mathcal{P}}_{t_i}$, respectively. We further define an *honesty coefficient*, denoted by k_j where $k_j > 0$, for each user j. Thus, $k_j \cdot p_{t_i,j}$ gives the real energy consumption of user j from time instant $t_i - SP$ to time instant t_i. Since the sum of all users' real energy consumption in the past sampling period must be equal to the total energy consumption of the neighborhood measured at the collector at time t_i, we have

$$k_1 p_{t_i,1} + k_2 p_{t_i,2} + ... + k_n p_{t_i,n} = \overline{\mathcal{P}}_{t_i} \tag{1}$$

Our objective is to find all the k_j's. Obviously, 1) if $k_j = 1$, then user j is honest and did not steal energy; 2) if $k_j > 1$, then user j recorded less energy than what he/she consumes and hence is an energy thief; and 3) if $0 < k_j < 1$, then user j recorded more than what he/she consumes, which means that his/her smart meter may be malfunctioning.

[2]Note that although we assume a secure firmware for SMs, the upper layer software for SMs can still be compromised.

In particular, with n linear equations, we can have a linear system of equations (LSE) as follows:

$$\begin{aligned} &k_1 p_{t_1,1} + k_2 p_{t_1,2} + ... + k_n p_{t_1,n} = \overline{\mathcal{P}}_{t_1} \\ &\vdots \\ &k_1 p_{t_n,1} + k_2 p_{t_n,2} + ... + k_n p_{t_n,n} = \overline{\mathcal{P}}_{t_n} \end{aligned}$$

which can also be formulated in matrix form:

$$\boldsymbol{P}\boldsymbol{k} = \overline{\boldsymbol{\mathcal{P}}}. \tag{2}$$

The jth column of $\boldsymbol{P}$ represents the data recorded and stored by user j or SM_j, while the ith row of $\boldsymbol{P}$ represents the data recorded by all the users at t_i. The collector can choose n time instances when $\overline{\mathcal{P}}_{t_i}$'s all have different values. In this case, it is highly likely that the LSE is independent and the rows of $\boldsymbol{P}$ are independent as well, especially when n is large[3]. Thus, the above LSE only has a single unique solution, i.e., the feasible solution $k_j = \overline{p}_{t_i,j}/p_{t_i,j}$ where $\overline{p}_{t_i,j}$ is the real energy consumption of user j from time instant $t_i - SP$ to time instant t_i.

Note that our model does not take into account energy dissipation, or technical losses (TLs), in the power system, which are mainly caused by the intrinsic inefficiencies in transformers and low voltage power lines. However, TLs can be calculated without using consumers' energy measurements. For example, Oliveira et al. [20] describe how to calculate TLs using measurements at the distribution station and the knowledge of the distribution network which does not need to compromise users' privacy. Thus, once the technical losses are calculated by the collector, the collector can adjust the model by subtracting the TLs from vector $\overline{\boldsymbol{\mathcal{P}}}$.

Besides, note that finding the honesty coefficient vector, $\boldsymbol{k}$, is delay tolerant. In other words, $\boldsymbol{k}$ is not required to be found and transmitted to the collector in real time. This gives priority to other real time traffic in the NAN, such as electricity pricing, incentive-based load reduction signals, and emergency load reduction signals.

IV. Finding Honesty Coefficients by P2P Computing

In what follows, based on P2P computing (or distributed/collaborative computing), we propose three algorithms that can solve the linear system of equations in (2) while preserving the users' privacy. The challenge is that each smart meter SM_j needs to find its own honesty coefficient k_j without knowing any of the other smart meters' recorded energy consumption data $p_{t_i,l}$'s, where $1 \leq i \leq n$ and $j \neq l$.

Specifically, we first develop an LU decomposition based approach, called LUD, to detect the energy thieves while preserving users' privacy. We notice that in its original form, LUD may not be numerically stable in large size networks. The reason is that the inaccuracies involved when using finite resolution numbers may lead to solutions with significant errors when n is large. Therefore, after proposing the LUD algorithm, we design another algorithm to achieve numerical

[3]Besides, note that usually there are always some appliances running at users' premises, such as refrigerators and air conditioners, whose working powers are in practice dynamic with some fluctuations instead of constants.

stability by exchanging rows of the matrix $\boldsymbol{P}$ during LU decomposition, i.e., LUD with partial pivoting (LUDP). After that, we also propose a QR Decomposition based algorithm, called QRD, to perform stable P2P computing in large-scale networks to find the energy thieves.

In the following, we detail these three algorithms, respectively, when the honesty coefficient vector, $\boldsymbol{k}$, is a constant, and then discuss the cases where $\boldsymbol{k}$ changes with time.

A. The LUD Algorithm

We first describe the LUD algorithm as follows, which is based on distributed LU decomposition. The LU decomposition is to factorize the energy consumption data matrix $\boldsymbol{P}$ into two triangular matrices: a lower triangular matrix $\boldsymbol{L}$ and an upper triangular matrix $\boldsymbol{U}$, i.e., $\boldsymbol{P} = \boldsymbol{LU}$.

The elements of upper triangular matrix $\boldsymbol{U}$ can be calculated as follows:

$$\begin{aligned} &u_{i,j} = 0, \quad i > j \\ &u_{1,j} = p_{t_{1,j}}, \quad j = 1, 2, ..., n \\ &u_{r,j} = p_{t_{r,j}} - \sum_{k=1}^{r-1} l_{r,k} u_{k,j}, \qquad r = 2, ..., n,\ j = r, ..., n \end{aligned} \tag{3}$$

where $p_{t_{i,j}}$ is the ith element of column j in matrix $\boldsymbol{P}$. Besides, the elements of lower triangular matrix $\boldsymbol{L}$ can be obtained by

$$\begin{aligned} &l_{i,j} = 0, \quad i < j \\ &l_{i,1} = \frac{p_{t_{i,1}}}{p_{t_{1},1}}, \quad i = 1, 2, ..., n \\ &l_{i,q} = \frac{p_{t_{i,q}} - \sum\limits_{k=1}^{q-1} l_{i,k} u_{k,q}}{u_{q,q}}, \quad q = 2, ..., n,\ i = q, .., n \end{aligned} \tag{4}$$

Note that the diagonal elements of $\boldsymbol{L}$ are equal to 1. This guarantees that the decomposition of $\boldsymbol{P}$ is unique.

After matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ are collaboratively computed, the following system can be solved:

$$\boldsymbol{Ly} = \overline{\boldsymbol{\mathcal{P}}}, \tag{5}$$

$$\boldsymbol{Uk} = \boldsymbol{y}. \tag{6}$$

In particular, to solve for $\boldsymbol{y}$, each SM_{j-1} will calculate y_j as follows, i.e., $y_1 = \overline{\mathcal{P}}_{t_1}$, and

$$y_j = \overline{\mathcal{P}}_{t_j} - \sum_{q=1}^{j-1} l_{j,q} y_q, \quad j = 2, ..., n. \tag{7}$$

The required values for this computation are the elements of $\boldsymbol{y}$ with index less than j and row j of $\boldsymbol{L}$. Finally, Each SM_j solves for k_j using backward substitution, i.e., $k_n = \frac{y_n}{u_{n,n}}$, and

$$k_j = \frac{y_j - \sum\limits_{p=j+1}^{n} u_{j,p} k_p}{u_{j,j}} \quad j = 1, ..., n-1. \tag{8}$$

Therefore, our LUD algorithm is composed of two parts: Distributed LU Decomposition and Backward Substitution, which are detailed in Procedure 1 and Procedure 2, respectively. Besides, before the algorithm can begin, the collector must number all the SMs from 1 to n, and the index number of any SM is only known to the SM itself and the collector. The collector also transmits $\overline{\mathcal{P}}_{t_{j+1}}$ to each SM_j to allow the SMs to collaboratively compute $\boldsymbol{L}$, $\boldsymbol{U}$, and $\boldsymbol{y}$. We denote the smart meter at the collector as SM_0. All SMs start running Procedure 1 when the collector requests them to by sending a control message.

Procedure 1 Distributed LU Decomposition

```
Input: $j \rightarrow SM_j$ , $\overline{\mathcal{P}}_{t_{j+1}} \rightarrow SM_j$
1: if $j = 0$ or $SM_j$ receives packets from $SM_{j-1}$ then
2:     if $j = 0$ then
3:         Compute $y_1$ using (7)
4:         Transmit $y_1$ only to $SM_1$
5:     end if
6:     if $1 \leq j \leq n-1$ then
7:         for $q = 1$ to $j$ do
8:             Compute $u_{q,j}$ using (3)
9:         end for
10:        for $q = j+1$ to $n$ do
11:            Compute $l_{q,j}$ using equation (4)
12:        end for
13:        Compute $y_{j+1}$ using (7)
14:        Transmit columns $1, 2, ..., j$ of $\boldsymbol{L}$ and all known
           elements of $y_1, ..., y_{j+1}$ only to $SM_{j+1}$
15:    end if
16:    if $j = n$ then
17:        Notify the collector that $\boldsymbol{L}$, $\boldsymbol{U}$, and $\boldsymbol{y}$ are avail-
           able
18:    end if
19: end if
```

Specifically, SM_0 first calculates $y_1 = \overline{\mathcal{P}}_{t_1}$, and then transmits it to SM_1. SM_0 does not need to compute any element of $\boldsymbol{L}$ or $\boldsymbol{U}$. After SM_1 receives y_1, it computes column 1 of $\boldsymbol{U}$, column 1 of $\boldsymbol{L}$, and y_2. Then, SM_1 transmits column 1 of $\boldsymbol{L}$, y_1, and y_2 to SM_2. SM_j $(1 < j < n)$, receives y_1 through y_j and columns 1 through $j-1$ of $\boldsymbol{L}$ from SM_{j-1}, based on which it calculates column j of $\boldsymbol{U}$, column j of $\boldsymbol{L}$, and y_{j+1}. After that, SM_j transmits columns 1 through j of $\boldsymbol{L}$ and y_1 through y_{j+1} to SM_{j+1}. Finally, SM_n receives y_n and columns 1 through $n-1$ from SM_{n-1}, calculates column n of $\boldsymbol{U}$ and column n of $\boldsymbol{L}$, and notifies the collector that the Back Substitution procedure, i.e., Procedure 2, can start. Notice that each SM_j $(j > 0)$ is responsible for computing column j of $\boldsymbol{U}$, column j of $\boldsymbol{L}$, and y_{j+1} $(1 \leq j < n)$.

After Procedure 1 ends, SM_j $(1 \leq j \leq n)$ has obtained the jth column of $\boldsymbol{U}$ and y_j. The collector then instructs all the smart meters to run Procedure 2 to solve for their own honesty coefficients according to (7), starting from SM_n. In particular, SM_n transmits $u_{n-1,n}k_n$, which is needed by SM_{n-1} to solve for k_{n-1}, along with $u_{n-2,n}k_n$, ..., $u_{1,n}k_n$, needed by SM_{n-2}, ..., SM_1, respectively, to SM_{n-1}. Similarly, SM_j $(1 < j < n)$ receives $\sum_{q=j+1}^{n} u_{j,q}k_q$ from SM_{j+1} and solves for k_j, and then transmits $\sum_{q=j}^{n} u_{j-1,q}k_q$ along with $u_{1,j}k_j$, ..., $u_{j-2,j}k_j$, $u_{1,j+1}k_{j+1}$, ..., $u_{j-2,j+1}k_{j+1}$, ..., $u_{1,n}k_n ... u_{j-2,n}k_n$ to SM_{j-1}. Finally, SM_1 receives $\sum_{q=2}^{n} u_{1,q}k_q$ from SM_2 and solves for k_1. Moreover, after obtaining its honesty coefficient, each smart meter SM_j encrypts k_j using the collector's public key, resulting in $E(k_j)$, and then transmits $E(k_j)$ to the collector. When all the $E(k_j)$'s have been reported to the collector, the LSE can be successfully solved, and hence the collector can decrypt all the elements of $\boldsymbol{k}$ and identify all the fraudulent users.

Notice that in LUD, the collector does not know $\boldsymbol{L}$ or $\boldsymbol{U}$, and hence cannot recover $\boldsymbol{P}$. Moreover, from equation (8), we can observe that SM_j will need all the elements of $\boldsymbol{U}$ in row j and $k_n, k_{n-1}, ..., k_{j+1}$ in order to compute k_j. If such data are transmitted to SM_j separately, an eavesdropper would be able to figure out all the elements of $\boldsymbol{U}$, except those on the diagonal, by eavesdropping on all the transmissions in the network. Since an eavesdropper is able to obtain $\boldsymbol{L}$ by eavesdropping, too, it can figure out some elements of $\boldsymbol{P}$ (e.g., all the elements above the diagonal of $\boldsymbol{P}$). To prevent this from happening, as mentioned above and shown in Procedure 2, we transmit the multiplication of an element of $\boldsymbol{U}$ and the corresponding honesty coefficient instead. Notice that in this case an eavesdropper may be able to guess the energy consumption of certain honest users (i.e., those whose honesty coefficients are equal to 1) at certain times by assuming $\boldsymbol{k} = \boldsymbol{1}$. However, even so since the eavesdropper does not know the mapping between smart meter indexes and users (only the collector knows), it cannot really know any user's energy consumption data. Besides, the eavesdropper does not know which users are honest anyway. In addition, to defend against the case that the collector has the capability of eavesdropping on all the transmissions in the network, we can just enable each neighboring two smart meters, i.e., SM_j and SM_{j+1} where $1 \leq j \leq n-1$, to establish a symmetric security key on their own to encrypt the data transmitted in Procedure 2 (line 7 and line 8). In so doing, no user's private data will be revealed to or recovered by anyone else.

Procedure 2 Backward Substitution

```
1:  if $j = n$ or $SM_j$ receives packet from $SM_{j+1}$ then
2:      Compute $k_j$ as described in (8) using $s_{j+1}$ if necessary
3:      Compute $E(k_j)$ and transmit $E(k_j)$ to the collector
4:      Compute $u_{q,j}k_j$ for $q = j-1, j-2, ..., 1$
5:      if $j \neq 1$ then
6:          Compute $s_j = \sum_{q=j}^{n} u_{j-1,q}k_q$
7:          Transmit $s_j$ to $SM_{j-1}$
8:          Transmit $u_{1,j}k_j$, ..., $u_{j-2,j}k_j$, $u_{1,j+1}k_{j+1}$, ..., $u_{j-2,j+1}k_{j+1}$, ..., $u_{1,n}k_n$, ..., $u_{j-2,n}k_n$ to $SM_{j-1}$
9:      end if
10: end if
```

B. The LUDP Algorithm

As mentioned before, LUD may not be numerically stable when n is large. Here, we propose another algorithm, i.e., LUD with partial pivoting (LUDP), to address the stability problem. Partial pivoting is to interchange rows of the matrix $\boldsymbol{P}$ in order to place the element that has the greatest absolute value in each column in the diagonal position of the matrix. Thus, LUDP decomposition has the following form, $\boldsymbol{EP} = \boldsymbol{LU}$, where $\boldsymbol{E}$ is the permutation matrix.

The LUDP algorithm consists of three procedures: LU Decomposition with Partial Pivoting, Forward Substitution, and Backward Substitution. Procedure 3 shows how LU decomposition with partial pivoting works. Specifically, we first let $\boldsymbol{U} = \boldsymbol{P}$. Then, SM_1 finds the maximum element in column 1 of $\boldsymbol{U}$, lets the pivot index of column 1 be the row this element is in, interchanges this element with the element in row 1 if it is not, and updates the first column of $\boldsymbol{U}$. SM_1 also computes the first column of $\boldsymbol{L}$, and transmits it together with the pivot index of column 1 to SM_2. Note that in LUDP, we compute $\boldsymbol{U}$ and $\boldsymbol{L}$ in a different way from that in LUD, as shown in Line 8 and Line 17 of Procedure 3, respectively, which now allows partial pivoting [13]. After receiving the data from SM_1, SM_2 repeats SM_1's row interchange, i.e., interchanging the element in column 2, SM_1's pivot index row of $\boldsymbol{U}$ with the element in column 2, row 1 of $\boldsymbol{U}$. Then, SM_2 performs its own row interchange, which is to interchange the maximum element in column 2 of $\boldsymbol{U}$ with the second element in column 2 of $\boldsymbol{U}$, lets the pivot index of column 2 be the row this maximum element was in, and updates the second column of $\boldsymbol{U}$. After that, SM_2 computes the second column of $\boldsymbol{L}$, and transmits the first two columns of $\boldsymbol{L}$ along with the pivot index of SM_1 and of SM_2 to SM_3. Finally, SM_n receives columns 1 to $n-1$ of $\boldsymbol{L}$ and all the previous nodes' pivot indexes from SM_{n-1}, repeats all the previous row interchanges, performs its own row interchange, and calculates column n of $\boldsymbol{U}$. Note that $l_{j,j} = 1$ for $1 \leq j \leq n$.

Procedure 3 LU Decomposition with Partial Pivoting

```
Input: $j \to SM_j$
1:  $\boldsymbol{U} = \boldsymbol{P}$
2:  if received packet from $SM_{j-1}$ or $j = 1$ then
3:      if $j \neq 1$ then
4:          Receive columns $1, 2, ..., j-1$ of $\boldsymbol{L}$ and pivot indexes
5:          for $f = 1$ to $j-1$ do
6:              Update column $j$ of $\boldsymbol{U}$ by interchanging the $jth$ element of row $f$ with the $j$the element of the pivot row of $SM_f$
7:              for $r = f+1$ to $n$ do
8:                  $u_{r,j} = u_{r,j} - l_{r,f}u_{f,j}$
9:              end for
10:         end for
11:     end if
12:     Compute $l_{j,j} = 1$
13:     if $j \neq n$ then
14:         Determine pivot rows in column $j$ of $\boldsymbol{U}$
15:         Interchange the $j$th element of row $j$ with the $j$th element of the pivot row in $\boldsymbol{U}$ and $\boldsymbol{L}$
16:         for $r = j+1$ to $n$ do
17:             Compute $l_{r,j} = \frac{u_{r,j}}{u_{j,j}}$
18:             Compute $u_{r,j} = 0$
19:         end for
20:         Transmit columns $1, 2..., j$ of $\boldsymbol{L}$ and pivot row indexes to $SM_{j+1}$.
21:     else
22:         Notify the collector that $\boldsymbol{L}$ and $\boldsymbol{U}$ are available
23:         Transmit all the $n$ pivot row indexes to $SM_0$
24:     end if
25: end if
```

Moreover, due to (2), we need make the same row interchanges for $\overline{\mathcal{P}}$ as those for $\boldsymbol{P}$. Thus, we let SM_n send all the n pivot row indexes to the collector SM_0, which then performs the same row interchanges for $\overline{\mathcal{P}}$. Now we can solve for $\boldsymbol{y}$ and $\boldsymbol{k}$ according to (5) and (6), respectively. In particular, since $\boldsymbol{y}$ is computed according to (7), $\boldsymbol{y}$ can only be computed after Procedure 3 is finished, which is different from that in LUD where $\boldsymbol{y}$ can be computed at the same time as $\boldsymbol{L}$ and $\boldsymbol{U}$. Therefore, we propose the Forward Substitution procedure as shown in Procedure 4, to enable the smart meters to solve for $\boldsymbol{y}$ in a distributed way. Forward Substitution calculates y_j according to (7), and works similar to Backward Substitution except that it starts from SM_1. At last, after $\boldsymbol{y}$ is available, Back Substitution as described in Procedure 2 can be used to solve for $\boldsymbol{k}$.

Procedure 4 Forward Substitution

```
Input: $j \to SM_j$
1: if $j = 0$ or $SM_j$ receives packets from $SM_{j-1}$ then
2:     if $j = 0$ then
3:         Compute $y_1 = \overline{\mathcal{P}}_{t_1}$ and transmit $y_1$ to $SM_1$
4:     end if
5:     if $1 \le j \le n-1$ then
6:         Compute $y_{j+1}$ as described in (7) using $s_{j-1}$ if necessary
7:         Transmit $y_{j+1}$ to $SM_{j+1}$
8:         Compute $l_{q,j}y_j$ for $q = j+1, j+2, ..., n$
9:         Compute $s_j = \sum_{q=1}^{j} l_{j+1,q}y_q$
10:        Transmit $s_j$ to $SM_{j+1}$
11:        Transmit $l_{j+2,1}y_1$, ..., $l_{n,1}y_1$, ..., $l_{j+2,j}y_j$, ..., $l_{n,j}y_j$ to $SM_{j+1}$
12:    end if
13:    if $j = n$ then
14:        Notify the collector that $\boldsymbol{y}$ is available
15:    end if
16: end if
```

Furthermore, the same as that in LUD, we can enable each neighboring two smart meters, i.e., SM_j and SM_{j+1} where $1 \le j \le n-1$, to establish a symmetric security key on their own and encrypt the data transmitted in Procedure 2 (line 7 and line 8) to protect users' privacy.

Compared to the LUD algorithm, LUDP takes more time to complete. This is because in LUDP the forward substitution to calculate $\boldsymbol{y}$ can only be carried out after $\boldsymbol{L}$ and $\boldsymbol{U}$ have been obtained, while in LUD, $\boldsymbol{y}$, $\boldsymbol{L}$, and $\boldsymbol{U}$ can be obtained at the same time. On the other hand, LUDP is numerically stable while LUD is not.

C. The QRD Algorithm

Here, we present another privacy-preserving energy theft detection algorithm, called QRD. In particular, by QR decomposition, matrix $\boldsymbol{P}$ can be decomposed into an orthogonal matrix $\boldsymbol{Q}$ and an upper triangular matrix $\boldsymbol{R}$, i.e., $\boldsymbol{P} = \boldsymbol{QR}$, where $\boldsymbol{Q^{-1}} = \boldsymbol{Q^T}$. Thus, we have $\boldsymbol{Pk} = \boldsymbol{QRk} = \overline{\boldsymbol{\mathcal{P}}}$, which yields a new system

$$\boldsymbol{Rk} = \boldsymbol{Q^T}\overline{\boldsymbol{\mathcal{P}}}. \tag{9}$$

The basic idea is to utilize distributed QR decomposition to enable each smart meter to compute its own honesty coefficient without using other smart meters' energy consumption data.

We first present how to determine $\boldsymbol{Q}$ and $\boldsymbol{R}$ in the following. In particular, $\boldsymbol{Q^T}$ is formed as the product of $\frac{n(n-1)}{2}$ plane rotation matrices as follows:

$$\boldsymbol{Q^T} = \boldsymbol{G_{n,n-1}}(\boldsymbol{G_{n-1,n-2}}\boldsymbol{G_{n,n-2}})\cdots(\boldsymbol{G_{2,1}}\cdots\boldsymbol{G_{n,1}}). \tag{10}$$

Let $\hat{\boldsymbol{P}}_{\boldsymbol{1,0}} = \boldsymbol{P}$ and

$$\hat{\boldsymbol{P}}_{\boldsymbol{i,j}} = (\boldsymbol{G_{i,j}}\cdots\boldsymbol{G_{n,j}})\cdots(\boldsymbol{G_{2,1}}\cdots\boldsymbol{G_{n,1}})\hat{\boldsymbol{P}}_{\boldsymbol{1,0}}.$$

Then, $\hat{\boldsymbol{P}}_{\boldsymbol{i,j}} = \boldsymbol{G_{i,j}}\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}$ when $i < n$ and $\hat{\boldsymbol{P}}_{\boldsymbol{i,j}} = \boldsymbol{G_{i,j}}\hat{\boldsymbol{P}}_{\boldsymbol{j,j-1}}$ when $i = n$. Besides, when $\boldsymbol{G_{i,j}}$ multiplies $\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}$ when $i < n$ (or $\hat{\boldsymbol{P}}_{\boldsymbol{j,j-1}}$ when $i = n$) from the left, it zeros element $\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}(i,j)$ when $i < n$ (or $\hat{\boldsymbol{P}}_{\boldsymbol{j,j-1}}(i,j)$ when $i = n$), modifies rows i and $i-1$ of $\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}$ (or $\hat{\boldsymbol{P}}_{\boldsymbol{j,j-1}}$), and preserves previously introduced zeros[4]. Finally, $\boldsymbol{Q^T P}$ reduces $\boldsymbol{P}$ into an upper triangular matrix $\boldsymbol{R}$, i.e., $\hat{\boldsymbol{P}}_{\boldsymbol{n,n-1}} = \boldsymbol{R}$.

The two most common methods to find plane rotation matrices ($\boldsymbol{G_{i,j}}$'s) are Householder Rotations and Givens Rotations (GR). In this paper we adopt the GR approach. The non-zero elements of $\boldsymbol{G_{i+1,j}}$ are

$$\begin{aligned} &g_{qq} = 1, q \neq i, j \\ &g_{i,i} = c_{i,j},\ g_{i+1,i+1} = c_{i,j},\ g_{i,i+1} = s_{i,j},\ g_{i+1,i} = -s_{i,j} \end{aligned} \tag{11}$$

where $c_{i,j}$ and $s_{i,j}$ are calculated as

$$p'_{i,j} = (p_{i,j}^2 + p_{i+1,j}^2)^{1/2},\ c_{i,j} = \frac{p_{i,j}}{p'_{i,j}},\ s_{i,j} = \frac{p_{i+1,j}}{p'_{i,j}}. \tag{12}$$

Note that for simplicity we use $p_{i,j}$ to denote the element in ith row and jth column in the previously rotated matrix, i.e., $\hat{\boldsymbol{P}}_{\boldsymbol{i+2,j}}$ when $i < n-1$ and $\hat{\boldsymbol{P}}_{\boldsymbol{j,j-1}}$ when $i = n-1$.

Besides, the elements of the matrix after rotation, i.e., $\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}$, are

$$\begin{aligned} &\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}(i,r) = c_{i,j}p_{i,r} + s_{i,j}p_{i+1,r} \quad \text{for } r \geq j \\ &\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}(i+1,j) = 0 \\ &\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}(i+1,r) = -s_{i,j}p_{i,r} + c_{i,j}p_{i+1,r} \quad \text{for } r > j \end{aligned} \tag{13}$$

We denote by $\mathbb{G}_{i,j}$ the set that contains $c_{i,j}$ and $s_{i,j}$, i.e., $\mathbb{G}_{i,j} = \{c_{i,j}, s_{i,j}\}$. From equation (12), we can observe that the values needed by SM_j to compute $\mathbb{G}_{i,j}$ reside in column j. This allows SM_j to find all its rotation matrices, i.e., $\boldsymbol{G_{j+1,j}}$,..., $\boldsymbol{G_{n,j}}$, only using its locally stored and calculated data. Moreover, (13) shows that SM_r, when $r > j$, needs the set $\mathbb{G}_{i,j}$ from SM_j to update its own data, i.e., column r of the rotated matrix $\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}$. In addition, notice that each column j ($1 \le j \le n$) has $n-j$ elements that need to be converted to zero in order to finally find $\boldsymbol{R}$ in (9). The set that contains all the $\mathbb{G}_{i,j}$'s which need to be calculated by SM_j, denoted by $\mathbb{B}_j$, is thus $\mathbb{B}_j = \{\mathbb{G}_{n-1,j}, \mathbb{G}_{n-2,j}, ..., \mathbb{G}_{j,j}\}$.

The QRD algorithm is composed of two procedures: Distributed QR Decomposition and Backward Substitution. Distributed QR Decomposition works as follows. SM_1 first generates $\boldsymbol{G_{n,1}}\cdots\boldsymbol{G_{2,1}}$ to zero $n-1$ elements in the first column of $\boldsymbol{P}$ and hence obtain the first column of $\boldsymbol{R}$. After

[4]$\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}(i,j)$ denotes the element of matrix $\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}$ in row i, column j.

that, it transmits $\mathbb{B}_1$ to SM_2. SM_2 uses $\mathbb{B}_1$ to update its energy consumption data, i.e,. the second column of $\boldsymbol{P}$, and generates $\boldsymbol{G_{n,2}} \cdots \boldsymbol{G_{3,2}}$ to find the second column of $\boldsymbol{R}$. After that, SM_2 transmits $\mathbb{B}_1$ and $\mathbb{B}_2$ to SM_3, and so on and so forth. Finally, SM_n receives $\mathbb{B}_1$, $\mathbb{B}_2$, ..., and $\mathbb{B}_{n-1}$ from SM_{n-1}, updates its own energy consumption data, finds the nth column of $\boldsymbol{R}$. SM_n then transmits $\mathbb{B}_1$, $\mathbb{B}_2$, ..., and $\mathbb{B}_n$ to the collector. Therefore, at the end of this procedure each smart meter SM_j can obtain the jth column of $\boldsymbol{R}$, and the collector can compute $\boldsymbol{Q^T}$ and hence $\boldsymbol{Q^T}\overline{\boldsymbol{\mathcal{P}}}$ using (10) and (11). The procedure is explained in details in Procedure 5.

Procedure 5 Distributed QR Decomposition

```
Input: $j \to SM_j$
1: if $j > 1$ and $SM_j$ receives $\mathbb{B}_1, \mathbb{B}_2 ... \mathbb{B}_{j-1}$ from $SM_{j-1}$ then
2:     for $f = 1$ to $j - 1$ do
3:         for $r = n - 1$ to $f$ do
4:             Update elements in column $j$ of $\boldsymbol{P}$ using $\mathbb{G}_{r,f}$ and $\mathbb{G}_{r+1,f}$ as described in (13).
5:         end for
6:     end for
7:     for $q = n$ to $j + 1$ do
8:         Compute $c_{q,j}$ and $s_{q,j}$ using (12) and store them
9:         Zero element $p_{q,j}$ using (13)
10:    end for
11: end if
12: if $j = 1$ then
13:    for $q = n$ to $j + 1$ do
14:        Compute $c_{q,j}$ and $s_{q,j}$ using (12) and store them
15:        Zero element $p_{q,j}$ using (13)
16:    end for
17: end if
18: if $j \neq n$ then
19:    Transmit $\mathbb{B}_1 ... \mathbb{B}_j$ to $SM_{j+1}$
20: else
21:    Transmit $\mathbb{B}_1 ... \mathbb{B}_j$ to the collector
22: end if
```

After Procedure 5, the collector will instruct the smart meters to run Backward Substitution to compute their honesty coefficients in a distributed way. In particular, according to (9), at SM_j $(1 \leq j \leq n)$ we have

$$r_{j,j}k_j + r_{j,j+1}k_{j+1} + ... + r_{j,n}k_n = \boldsymbol{Q^T}\overline{\boldsymbol{\mathcal{P}}}(j) \tag{14}$$

where $r_{i,j}$ is the element in the ith row and the jth column of $\boldsymbol{R}$, and $\boldsymbol{Q^T}\overline{\boldsymbol{\mathcal{P}}}(j)$ is the jth element of $\boldsymbol{Q^T}\overline{\boldsymbol{\mathcal{P}}}$. So, once the collector receives all the sets $\mathbb{B}_j$'s from SM_n, it will compute $\boldsymbol{Q^T}\overline{\boldsymbol{\mathcal{P}}}$ and distribute the jth element to SM_j. SM_n can then obtain its honesty coefficient k_n. After that, SM_n transmits $r_{n-1,n}k_n$, which is needed by SM_{n-1} to solve for k_{n-1}, along with $r_{n-2,n}k_n$, ..., $r_{1,n}k_n$, needed by SM_{n-2}, ..., SM_1, respectively, to SM_{n-1}. Similarly, SM_j receives $\sum_{q=j+1}^{n} r_{j,q}k_q$ from SM_{j+1} and solves for k_j, and then transmits $\sum_{q=j}^{n} r_{j-1,q}k_q$ along with $r_{1,j}k_j$, ..., $r_{j-2,j}k_j$, $r_{1,j+1}k_{j+1}$, ..., $r_{j-2,j+1}k_{j+1}$, ..., $r_{1,n}k_n ... r_{j-2,n}k_n$ to SM_{j-1}. Finally, SM_1 receives $\sum_{q=2}^{n} r_{1,q}k_q$ from SM_2 and solves for k_1. Moreover, after obtaining its honesty coefficient, each smart meter SM_j encrypts k_j using the collector's public key, resulting in $E(k_j)$, and then transmits $E(k_j)$ to the collector. When all the $E(k_j)$'s have been reported to the the collector, the LSE can be successfully solved, and hence the collector can decrypt all the elements of $\boldsymbol{k}$ and identify all the fraudulent users. This procedure is detailed in Procedure 6.

Procedure 6 Backward Substitution

```
1: if $j = n$ or $SM_j$ receives packet from $SM_{j+1}$ then
2:     Compute $k_j$ as described in (14) using $s_{j+1}$ if necessary
3:     Compute $E(k_j)$ and transmit $E(k_j)$ to the collector
4:     Compute $r_{q,j}k_j$ for $q = j - 1, j - 2, ..., 1$
5:     if $j \neq 1$ then
6:         Compute $s_j = \sum_{q=j}^{n} r_{j-1,q}k_q$
7:         Transmit $s_j$ to $SM_{j-1}$
8:         Transmit $r_{1,j}k_j$, ..., $r_{j-2,j}k_j$, $r_{1,j+1}k_{j+1}$, ..., $r_{j-2,j+1}k_{j+1}$, ..., $r_{1,n}k_n$, ..., $r_{j-2,n}k_n$ to $SM_{j-1}$
9:     end if
10: end if
```

Notice that in QRD, although the collector can recover $\boldsymbol{Q}$ by knowing the rotation matrices $\boldsymbol{G_{i,j}}$'s, it does not know $\boldsymbol{R}$ and hence cannot recover $\boldsymbol{P}$. Moreover, similar to that in LUD and LUDP, we can enable each neighboring two smart meters, i.e., SM_j and SM_{j+1} where $1 \leq j \leq n-1$, to establish a symmetric security key on their own and encrypt the data transmitted in Procedure 6 (line 7 and line 8) to protect users' privacy, if the collector can eavesdrop on all the transmissions in the network.

D. Variable Honesty Coefficients

In the above LUD, LUDP, and QRD algorithms, we have only considered that the honesty coefficient vector $\boldsymbol{k}$ is a constant[5]. However, when an illegal user commits energy theft, it is possible that the rate at which he/she steals energy is variable. In other words, an illegal user can alter the smart meter in such a way that it steals energy at different rates at different times. Unfortunately, if $\boldsymbol{k}$ changes in an LSE, the proposed algorithms may not work well. Next, we design adaptive algorithms to address this problem.

We notice that when $\boldsymbol{k}$ changes in an LSE, the LUD, LUDP, and QRD algorithms will result in an honesty coefficient vector, many of whose elements are not equal to 1. Thus, when the collector gets the honesty coefficient vector $\boldsymbol{k}$ and counts the number of elements that are not equal to 1, it can infer by statistics whether it is possible to have this many energy thieves in the network. If it is unlikely for this event to happen, the collector can reduce the sampling period SP and run the algorithms again, until the possibility of that event is high and $\boldsymbol{k}$ does not change any more.

We give a mathematical model as follows. Assume there are n users in a serviced area and each of them commits energy theft independently with the same probability p. Let X denote the total number of energy thieves in the neighborhood. Then, X is a random variable, which has a Binomial distribution.

[5]Note that we can enable the SMs to report to the collector if they are disconnected from the loads.

Thus, when the collector runs LUD/LUDP/QRD and obtains the honesty coefficient vector $\boldsymbol{k}$, it can find the number of elements that are not equal to 1, which we denote by Y. Then, the collector can calculate the probability that this event happens as follows:

$$P(X=Y)=\binom{n}{Y}p^Y(1-p)^{n-Y}. \qquad (15)$$

In addition, in the case that each user j commits energy theft independently with a different probability p_j, X is also a random variable, but its expectation becomes $\mathbb{E}[X]=\sum_{j=1}^{n}p_j$. Recall the Chernoff bounds [21]:

- For any $\delta>0$,

$$P(X>(1+\delta)\mathbb{E}[X])<e^{-\mathbb{E}[X]f(\delta)}$$

where $f(\delta)=(1+\delta)\log(1+\delta)-\delta$.

- For any $0<\delta<1$,

$$P(X<(1-\delta)\mathbb{E}[X])<e^{-\frac{1}{2}\delta^2\mathbb{E}[X]}.$$

Then, the collector can infer whether the obtained $\boldsymbol{k}$ is true or not by calculating

$$P(X\geq Y)<e^{-\mathbb{E}[X]f(\delta)} \text{ with } \delta=Y/\mathbb{E}[X]-1 \qquad (16)$$

when $Y>\mathbb{E}[X]$, and

$$P(X\leq Y)<e^{-\frac{1}{2}\delta^2\mathbb{E}[X]} \text{ with } \delta=1-Y/\mathbb{E}[X] \qquad (17)$$

when $Y<E[X]$. Besides, when $Y=E[X]$, we set $P(X=Y)=1$[6].

Thus, if the estimated probability P is lower than a threshold th, we reduce the sampling period SP by g ($g>0$), which is a step variable, and run the LUD/LUDP/QRD algorithm again to obtain another $\boldsymbol{k}$. This process repeats until P is no less than the threshold and the obtained $\boldsymbol{k}$ is the same as the previous one. By then we consider the $\boldsymbol{k}$ is true, i.e., the real honesty coefficient vector in the network.

We finally present the adaptive LUD/LUDP/QRD algorithm in Procedure 7, which can detect illegal users with variable honesty coefficients.

V. Computational and Communication Complexity Analysis

In this section we analyze the computational and communication complexities of LUDP and QRD, the two stable algorithms. We define the computational complexity as the number of elementary arithmetic operations (additions, subtractions, multiplications, divisions, and square roots), plus the number of comparisons and row exchanges needed to find vector $\boldsymbol{k}$. We define the communication complexity as the total traffic demand in the network, i.e., the total number of quantities that need to be transmitted in the network.

[6]As shown in Procedure 7, by setting $P(X=Y)=1$ in this case, we will run the LUD/LUDP/QRD algorithm again with a reduced sampling period. If the obtained $\boldsymbol{k}$ does not change any more, we consider it is the true honesty coefficient vector we are looking for.

Procedure 7 **Adaptive LUD/LUDP/QRD Algorithm**

```
1:  repeat
2:      The collector instructs all SMs to take $n$ samples
        with a initial sampling period $SP$
3:      Run the LUD/LUDP/QRD algorithm
4:      if The collector receives all elements in $\boldsymbol{k}$ then
5:          $Y$ = the number of elements in $\boldsymbol{k}$ unequal to 1,
            i.e., the number of illegal SMs
6:      end if
7:      The collector calculates the probability that there
        are $Y$ illegal users according to (15), (16) or (17),
        denoted by $P$.
8:      if $P < th$ (a threshold) or $P = 1$ then
9:          $SP = SP - g$ ($g > 0$ is a step variable)
10:     end if
11: until $P \geq th$ and $\boldsymbol{k}$ does not change any more
```

A. The LUDP Algorithm

1) Computational Complexity: To determine LUDP's computational cost, we need look into the operations in Procedures 3, 4, and 2 as follows.

In Procedure 3, lines 6, 8, 14, 15 and 17 conduct computations. Specifically, line 6 performs one row exchange and repeats $(j-1)$ times at SM_j, where $2\leq j\leq n$ and n is the number of users in the network. Line 15 also performs one row exchange and repeats $(n-1)$ times. Thus, the total number of row exchanges in Procedure 3 is $\sum_{j=2}^{n}(j-1)+(n-1)=\frac{n^2+n-2}{2}$.

Line 8 performs two elementary operations and lies inside a nested "for" loop. To find the total number of times that line 8 is executed, we first consider the nested "for" loops only, then consider the number of times the procedure is executed. In particular, the inner "for" loop iterates $(n-f)$ times and the outer "for" loop iterates $(j-1)$ times. Therefore, we have that line 8 executes $\sum_{f=1}^{j-1}(n-f)=n(j-1)-\frac{j(j-1)}{2}$ times for each $2\leq j\leq n$. Then, the total number of elementary operations contributed by this line is $2\sum_{j=2}^{n}\big(n(j-1)-\frac{j(j-1)}{2}\big)=\frac{2n^3-3n^2+n}{3}$.

Line 14 contributes one search for the highest absolute value among $(n-j+1)$ elements in column j, where $1\leq j\leq n-1$. In the worst case scenario, each search needs $(n-j)$ comparisons to determine the pivot row. Thus, the total number of elementary operations by line 14 is $\sum_{j=1}^{n-1}(n-j)=\frac{n^2-n}{2}$.

In addition, line 17 performs one elementary operation and repeats $(n-j)$ times for $1\leq j\leq n-1$. Therefore, the total number of elementary operations performed by line 17 is $\sum_{j=1}^{n-1}(n-j)=\frac{n^2-n}{2}$.

In Procedure 4, line 6 computes $(j-1)$ multiplications and as many additions or subtractions, which are the computations in lines 8 and 9 carried out at the previous node. Line 11 also conducts $j(n-j-1)$ multiplications for $1\leq j\leq n-1$. Therefore, the total number of elementary arithmetic operations in the Forward Substitution procedure is given by $2\sum_{j=1}^{n}(j-1)=n^2-n$.

The computational complexity of Procedure 2, i.e., Backward Substitution, is similar to that of Procedure 4, i.e., Forward Substitution, with the exception of n additional divisions. So Procedure 2's computational complexity is

$$2\sum_{j=1}^{n}(j-1)+n=n^2. \tag{18}$$

As a result, adding the above computational complexity results together, we can find that the total computational complexity of LUDP, denoted by PC_{LUDP}, is

$$PC_{LUDP}=\frac{4n^3+15n^2-7n-6}{6}.$$

2) Communication Complexity: The total communication complexity of LUDP is also determined by the traffic demand of Procedures 3, 4, and 2.

In Procedure 3, only lines 20 and 23 account for communications. According to line 20, SM_j transmits to SM_{j+1} the first j columns of $\boldsymbol{L}$ ($n-f+1$ elements in column f) and j pivot row indexes. Besides, in line 23, SM_n transmits all the n pivot row indices to SM_0, i.e., n quantities. Thus, the traffic demand of Procedure 3 is $\sum_{j=1}^{n-1}\big(\sum_{f=1}^{j}(n-f+1)+j\big)+n=\frac{4n^3+6n^2-2n}{12}$.

In Procedure 4, lines 3, 7, 10, 11, and 14 carry out transmissions. Lines 3 and 14 are executed only once and transmit one quantity each, while lines 7 and 10 repeat $(n-1)$ times, each of which transmits one quantity. Line 11 transmits $j(n-j-1)$ quantities for $1\le j\le n-1$. Consequently, the traffic demand of Procedure 4 is $\sum_{j=1}^{n-1}j(n-j-1)+2(n-1)+2=\frac{n^3-3n^2+14n}{6}$.

Similarly, in Procedure 2, lines 3, 7, and 8 carry out transmissions. Particularly, lines 3 and 7 transmit one quantity each and repeat n and $n-1$ times, respectively. Line 8 transmits $(j-2)(n-j+1)$ quantities for $2\le j\le n$. Therefore, the traffic demand of Procedure 2 is

$$\sum_{j=2}^{n}(j-2)(n-j+1)+2n-1=\frac{n^3-3n^2+14n-6}{6}. \tag{19}$$

Thus, from the above results, we can find that the total communication complexity of LUDP, denoted by MC_{LUDP}, is

$$MC_{LUDP}=\frac{8n^3-6n^2+54n-12}{12}.$$

B. The QRD Algorithm

1) Computational Complexity: We need examine Procedures 5 and 6 to analyze QRD's computational complexity.

In Procedure 5, line 4 performs six elementary operations $(n-f)$ times for $1\le f\le j-1$, where $2\le j\le n$. Thus, the total number of elementary operations by line 4 is $6\sum_{j=2}^{n}\sum_{f=1}^{j}(n-f)=2n^3-8n+6$.

Besides, line 14 carries out six elementary operations and repeats $(n-j)$ times for $1\le j\le n-1$. Therefore, the total number of elementary operations by line 14 is $6\sum_{j=1}^{n-1}(n-j)=3n^2-3n$.

Moreover, the computational complexity of Procedure 6 is the same as that of Procedure 2 shown in (18). As a result, from the above results and (18), we can have that the computational complexity of QRD, denoted by PC_{QRD}, is

$$PC_{QRD}=2n^3+4n^2-11n+6.$$

2) QRD Communication Complexity: The total communication complexity of QRD is also determined by the traffic demand of Procedures 5 and 6.

In Procedure 5, lines 19 and 21 carry out transmissions of $\mathbb{B}_1\ldots\mathbb{B}_j$ for $1\le j\le n$. Since $\mathbb{B}_k$ $(1\le k\le j)$ contains $n-k$ $\mathbb{G}_{p,q}$ sets, each of which contains two quantities, the traffic demand of Procedure 5 is $2\sum_{j=1}^{n}\sum_{k=1}^{j}(n-k)=\frac{2n^3-2n}{3}$.

Moreover, the traffic demand of Procedure 6 is the same as that of Procedure 2 shown in (19). Consequently, the communication complexity of QRD, denoted by MC_{QRD}, is

$$MC_{QRD}=\frac{5n^3-3n^2+10n-6}{6}.$$

VI. Simulation Results

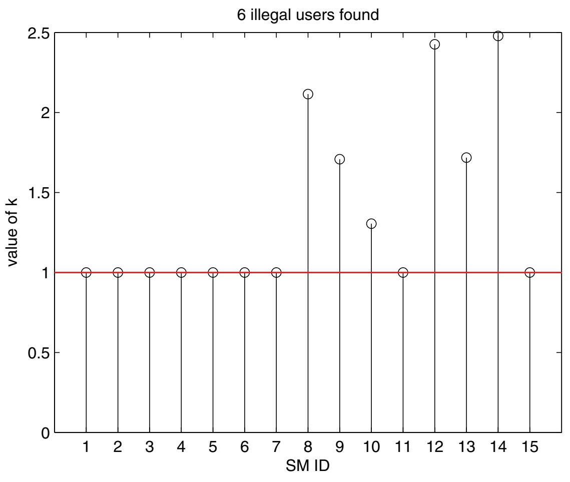
Here, we perform two series of simulations to evaluate the performance of our privacy-preserving energy theft detection algorithms LUD, LUDP, and QRD. In the first part, we assume that illegal users steal energy at a constant rate and thus have constant honesty coefficients. In the second part, we consider that illegal users have variable honesty coefficients. We conduct simulations in Matlab R2010a. The simulation results in the above two cases are presented in Section VI-A and Section VI-B, respectively.

Besides, we generate users' energy consumption data, $\boldsymbol{P}$, based on a set of data from [22] and [23]. These two studies conduct experiments in which both commercial and residential users are metered every hour and every half-hour, respectively. With these measurements, both studies provide typical daily user load profiles for different days of the week and different seasons of the year.

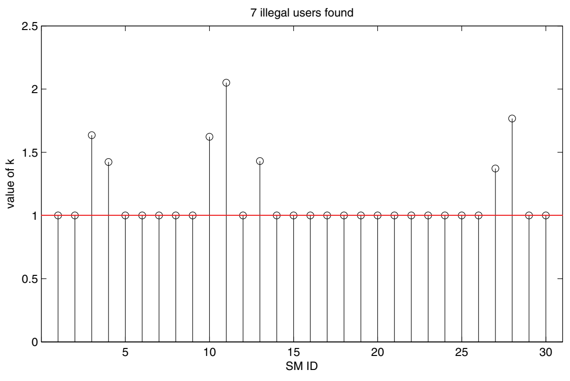
A. Constant Honesty Coefficients

We first perform simulations when illegal users steal energy at a constant rate. In other words, each illegal SM chooses a rate to steal energy and never changes this rate or stops stealing, thus having a constant honesty coefficient.

We evaluate the performance of LUD, LUDP, and QRD, when every user commits energy theft with a probability of 0.3 and there are totally 15, 30, and 50 users, respectively. Each energy thief chooses an honesty coefficient uniformly and randomly in [1.1, 10]. As shown in Fig. 2, the LUD algorithm can work well when there are 15 and 30 users in total. In particular, in Fig. 2(a) we can see that 6 users have an honesty coefficient larger than 1. It means that these 6 SMs only record a fraction of their consumed energy. Using these results, the collector can easily identify the energy thieves and how much less they have paid in their monthly bills. We can also observe that the legal users have an honesty coefficient equal to 1 and can be easily identified as well. Similar results are shown in Fig. 2(b) when there are 30 users. Besides, we can also find that LUDP and QRD can obtain the same results as LUD in these two cases. Moreover, the results of LUD, LUDP, and QRD when the number of users is 50 are presented in Fig. 3. In this case, the LUD algorithm is not stable. It finds 49 illegal users while in practice there are only 17 energy thefts. In contrast, the LUDP and QRD algorithms can successfully identify all the 17 illegal users.

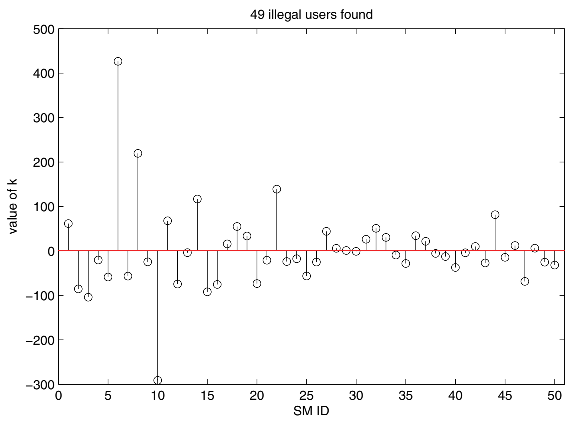


(a) 15 users

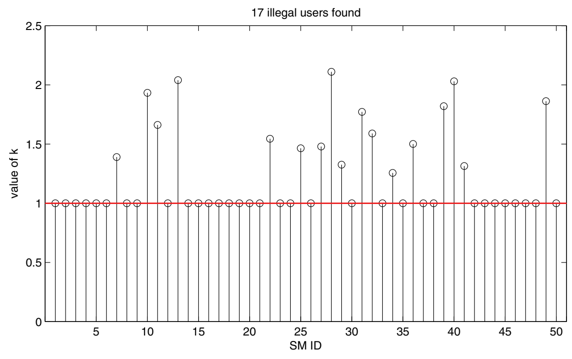


(b) 30 users

Fig. 2. Elements of $\boldsymbol{k}$ obtained by the LUD algorithm.



(a) LUD



(b) LUDP and QRD

Fig. 3. Elements of $\boldsymbol{k}$ obtained by the LUD, LUDP, and QRD algorithms in a network of 50 users.

B. Variable Honesty Coefficients

We then conduct simulations when illegal users steal energy at variable rates. We consider that each energy thief chooses a new honesty coefficient uniformly and randomly in [1.1, 10] each time after a certain period. we first consider that all the users commit energy theft with the same probability $p = 0.3$, and then consider that each user commits energy theft with a probability independently and randomly chosen between 0.3 and 0.7.

In particular, when all the users have the same cheating probability $p = 0.3$, we find that the adaptive LUD algorithm is not stable when there are more than 25 users in the network. The results are omitted due to space limitation. Besides, we show the results of the adaptive LUDP/QRD algorithm in Fig. 4, when the number of users is equal to 100, 200, and 300, respectively. We can see that all the energy thieves can be found. Moreover, when each user commits energy theft with a probability independently and randomly chosen between 0.3 and 0.7, we show the results of the adaptive LUDP/QRD algorithm in Fig. 5, in the cases that the number of users is equal to 100, 200, and 300, respectively. We can find that in these cases, the adaptive LUDP/QRD algorithm can also successfully and efficiently identify those fraudulent users.

VII. Conclusion

In this paper, we have presented three P2P computing algorithms, i.e., LUD, LUDP, and QRD, which can identify the users who are committing energy theft in smart grids while preserve all users' privacy. The three algorithms are distributed algorithms and are based on LU or QR decomposition. We can observe that no private data from any user needs to be transmitted to other users or to the collector, which cannot be recovered either, thus preserving users' privacy. We have also analyzed the computational and communication complexities of the proposed algorithms, and find that QRD has higher computational complexity and higher communication complexity compared to LUDP. Moreover, extensive simulations have been conducted. The simulation results show that fraudulent users can be detected both when they commit energy theft at a constant rate, i.e., with constant honesty coefficients, and when they steal energy at variable rates, i.e., with variable honesty coefficients.

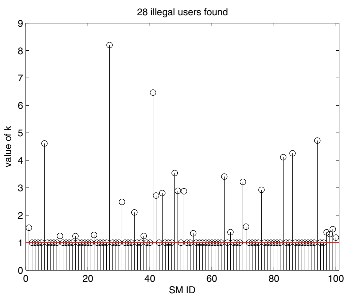
References

[1] S. Salinas, M. Li, and P. Li, "Privacy-preserving energy theft detection in smart grids," in *Proc. IEEE SECON*, June 2012.

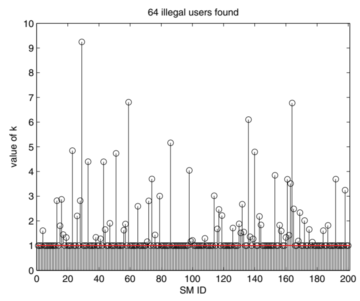
[2] P. McDaniel and S. McLaughlin, "Security and privacy challenges in the smart grid," *IEEE Security Privacy*, vol. 7, no. 3, pp. 75–77, May–June 2009.

[3] A. Nizar, Z. Dong, and Y. Wang, "Power utility nontechnical loss analysis with extreme learning machine method," *IEEE Trans. Power Syst.*, vol. 23, no. 3, pp. 946–955, Aug. 2008.

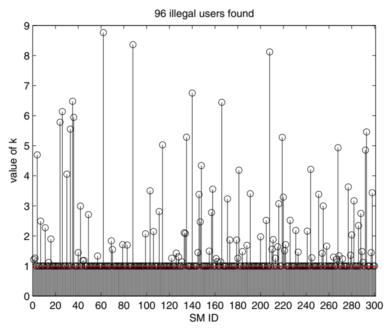
[4] J. Nagi, K. Yap, S. Tiong, S. Ahmed, and A. Mohammad, "Detection of abnormalities and electricity theft using genetic support vector machines," in *Proc. IEEE Region 10 Conf.*, Nov. 2008.



(a) 100 users

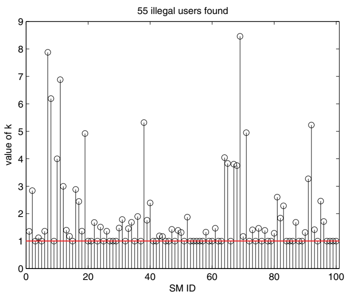


(b) 200 users

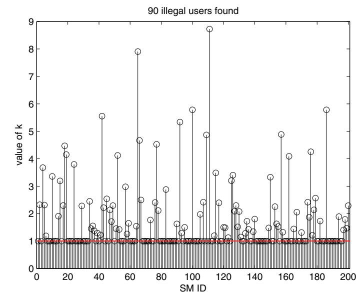


(c) 300 users

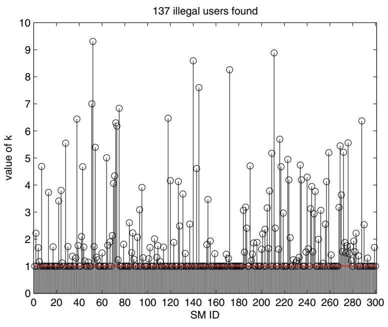
Fig. 4. Elements of $\boldsymbol{k}$ obtained by the LUDP and QRD algorithms – constant cheating probability.



(a) 100 users



(b) 200 users



(c) 300 users

Fig. 5. Elements of $\boldsymbol{k}$ obtained by the LUDP and QRD algorithms – variable cheating probability.

[5] S. Depuru, L. Wang, and V. Devabhaktuni, "Support vector machine based data classification for detection of electricity theft," in *Proc. Power Syst. Conf. Exposition (PSCE)*, Mar. 2011.

[6] C. Bandim, J. Alves, A. Pinto, F. Souza, M. Loureiro, C. Magalhges, and F. Galvez-Durand, "Identification of energy theft and tampered meters using a central observer meter: a mathematical approach," in *Proc. Transmission Distrib. Conf. Exposition*, Sep. 2003.

[7] A. Ruzzelli, C. Nicolas, A. Schoofs, and G. O'Hare, "Real-time recognition and profiling of appliances through a single electricity sensor," in *Proc. IEEE SECON*, June 2010.

[8] E. L. Quinn, "Privacy and the new energy infrastructure," *Social Science Research Netw.*, pp. 1995–2008, 2009 [Online]. Available: http://ssrn.com/paper=1370731

[9] "Privacy and the smart grid," NIST Guidelines for Smart Grid Cyber Security: vol.2, Aug. 2010 [Online]. Available: http://csrc.nist.gov/publications/nistir/ir7628/nistir-7628_vol1.pdf

[10] F. Li, B. Luo, and P. Liu, "Secure information aggregation for smart grids using homomorphic encryption," in *Proc. 1st IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Oct. 2010.

[11] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "PDA: Privacy-preserving data aggregation in wireless sensor networks," in *Proc. IEEE INFOCOM*, May 2007.

[12] X. Shen, H. Yu, J. Buford, and M. Akon, *Handbook of Peer-to-Peer Networking*. Springer Verlag, 2010.

[13] B. Noble, *Applied Linear Algebra*. Prentice Hall, 1969.

[14] S. Abdelhak, A. Abdelgawad, S. Ghosh, and M. Bayoumi, "A complete scheme for distributed LU decomposition on wireless sensor networks," in *Proc. 53rd IEEE Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2010.

[15] Q. Liang and L. Wang, "Redundancy reduction in wireless sensor networks using SVD-QR," in *Proc. IEEE Military Commun. Conf. (MILCOM)*, Oct. 2005.

[16] A. K. Zille Huma Kamal amd Ajay Gupta Leszek Lilien, "Classification using efficient LU decomposition in sensornets," in *Proc. WSN*, July 2006.

[17] G. A. Geistt and C. H. Romine, "LU factorization algorithms on distributed-memory multiprocessor architectures," *SIAM J. Scientific Statistical Comput.*, vol. 9, no. 4, pp. 639–649, July 1988.

[18] S. Depuru, L. Wang, and V. Devabhaktuni, "A conceptual design using harmonics to reduce pilfering of electricity," in *Proc. Power Energy Society General Meeting*, July 2010.

[19] M. LeMay and C. A. Gunter, "Cumulative attestation kernels for embedded systems," in *Proc. 14th European Conf. Research Comput. Security (ESORICS)*, Sep. 2009.

[20] M. Oliveira and A. Padilha-Feltrin, "A top-down approach for distribution loss evaluation," *IEEE Trans. Power Del.*, vol. 24, no. 4, pp. 2117–2124, Oct. 2009.

[21] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT Press, 2001.

[22] "California commercial end-use survey," Itron, June 2006 [Online]. Available: www.energy.ca.gov/ceus/

[23] "Electricity user load profiles by profile class," UKERC Energy Data Center, June 1997 [Online]. Available: http://data.ukedc.rl.ac.uk/browse/edc/Electricity/LoadProfile/data