2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)

Privacy-Preserving Energy Theft Detection in Smart Grids

Sergio Salinas, Ming Li, and Pan Li
Department of Electrical and Computer Engineering,
Mississippi State University, Mississippi State, MS 39762
Email: sergio.salinas@ieee.org, ml845@msstate.edu, li@ece.msstate.edu

***Abstract*—In the U.S., energy theft causes 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 the users to send their private information, e.g., load files or meter readings at certain times, to the UCs which invades users' privacy and raises serious concerns about privacy, safety, etc. As far as we know, we are the first to investigate the energy theft detection problem considering users' privacy issues. In this paper, we propose to solve in a distributed fashion a linear system of equations (LSE) for the users' "*honesty coefficients*", which indicate the users are honest when equal to 1 and are fraudulent when larger than 1. In particular, we develop two distributed privacy-preserving energy theft detection algorithms based on LU decomposition, called LUD and LUPD, respectively, which can identify fraudulent users without invading any user's privacy. Compared to LUD, LUPD requires higher execution time but is stable even in large-size systems. Moreover, the LUD and LUPD algorithms are proposed in the case that users commit energy theft at a constant rate, i.e., with constant honesty coefficients. We also propose adaptive LUD/LUPD algorithms to account for the scenarios where the users have variable 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.**

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 [1]. 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 power 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.

This work was supported by the U.S. National Science Foundation under grants CNS-1149786, ECCS-1128768, and CNS-1147851.

Some research has been conducted to detect energy theft in traditional power grids. Nizar et al. [2] 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. [3] propose a similar approach but choose to use genetic algorithms and Support Vector Machine (SVM) instead of ELM. Depuru et al. [4] 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. [5] 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 costumers. Moreover, criminals may utilize such private information to commit crimes. For instance, the robbers may analyze the energy consumption pattern of the potential victims to deduce their daily behavior. They can even know if a robbery alarm has been set at their target location [6]. Many researchers, such as Quinn [7], 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.

Unfortunately, there is currently a lack of research on

978-1-4673-1905-8/12/$31.00 ©2012 IEEE

privacy in smart grids. Li et al. [8] design a privacy-preserving aggregation protocol to collect the total energy consumption of a group of users at a distribution station in smart grids. To the best of our knowledge, we are the first to investigate privacy issues in energy theft detection.

Intuitively and as in previous works, we need to know about a user's electric power 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, we propose to solve in a distributed fashion a linear system of equations (LSE), which involves the respective energy consumption of n users in a neighborhood at n time instances and the total energy consumption of the n users measured at a local data collector, 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 decomposition to solve our LSE. Although some distributed algorithms for LU decomposition have been proposed in the literature, e.g., [9]–[11], they cannot preserve each node's private information. In this paper, we first develop a distributed privacy-preserving energy theft detection algorithm by 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 by LU decomposition with partial pivoting, called LUPD, which can find all the energy thieves even in large size networks but has higher execution time. Besides, the LUD and LUPD algorithms are proposed in the case that users commit energy theft at a constant rate, i.e., with constant honesty coefficients. We also propose adaptive LUD/LUPD algorithms to account for the scenarios where the users have variable honesty coefficients.

The rest of this paper is organized as follows. Section II introduces the considered network architecture, possible attacks on smart meters by energy thieves, and how we can implement the proposed algorithms in the smart meters without being tampered. Section III presents the linear system of equations for energy theft detection, and Section IV details the proposed distributed algorithms for solving the LSE. Simulation results are presented in Section V. Finally, we conclude this paper in Section VI.

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.

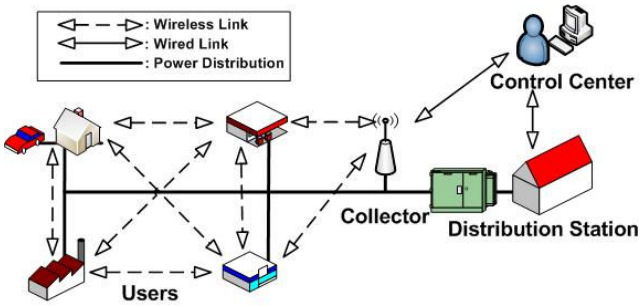


Fig. 1. A typical architecture of Neighborhood Area Network (NAN).

A. Network Architecture

In the smart grid, communications and electricity networks overlay each other. Utility companies (UCs) deploy control centers (CC) to monitor their distribution stations (DS) and distribution networks, and deploy SMs at users's premises to measure their individual energy consumption. Since the CC are usually physically far away from the users, a communication entity that can facilitate the communication between the SMs 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 Figure 1. In a serviced area, the users' SMs together with the collector form a Neighborhood Area Network (NAN). The communications between 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 [12] 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 from 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 power consumption. In this case, the illegal user may tamper all or some of the functionalities of the SMs. Network attack may be easier to launch and more difficult to detect.

In addition to attacking the 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 by 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. [13] 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 power 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 neighborhood 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 by user j from time instant $t_i - SP$ to time instant t_i. Since the sum of all the recorded energy consumption at time t_i must be equal to the total energy consumption of the neighborhood measured at the collector at time t_i, we have

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

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 smart meter may be malfunctioning. In order to solve for k_j's, we need n independent linear equations like the one shown above[3].

With n linear equations, we can thus have a linear system of equations (LSE) as follows:

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

This LSE can also be formulated in matrix form:

$$\boldsymbol{P}\boldsymbol{k} = \overline{\boldsymbol{\mathcal{P}}}. \quad (2)$$

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[4].

Besides, 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. In the next section, we will present our privacy-preserving energy theft detection algorithms, which can compute the unknown honesty coefficient vector $\boldsymbol{k}$ in a distributed way.

IV. Distributed Computation of Honesty Coefficients

In what follows, we develop two 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 stability by exchanging rows of the matrix $\boldsymbol{P}$ during LU decomposition, i.e., LUD with partial pivoting (LUPD).

Notice that although some other methods have been proposed in the literature, which can solve LSEs with lower computational complexity than LU decomposition, they usually

[2]Note that although we assume a secure firmware for SMs, the upper layer software for SMs can still be compromised. We only make this assumption to allow the proposed algorithms to be correctly executed.

[3]For example, the collector can choose the n time instances when $\overline{\mathcal{P}}_{t_i}$'s all have different values. In this case, it is highly likely that the n linear equations are independent of each other, especially when n is large.

[4]Note that our model does not take into account power dissipation, or technical losses (TLs), which can be calculated using measurements at the collector and the knowledge of the distribution network without compromising users' privacy [14].

consider special matrices. For example, the Thomas algorithm can decompose the matrix with computational complexity of $O(n)$ if the matrix under study is a tridiagonal matrix. Besides, Cholesky decomposition can improve the computational complexity of LU decomposition by a constant factor, but requires the matrix to be positive and definite [15]. Since the LSE studied in this paper has a dense matrix $\boldsymbol{P}$ (there are always some electrical devices working in each house/apartment, e.g., refrigerator, air conditioner), we choose to only employ LU decomposition to solve the LSE.

In the following, we first detail these two 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

1) Algorithm Description: The LU decomposition is to factorize the power consumption data matrix $\boldsymbol{P}$ into two triangular matrices: a lower triangular matrix $\boldsymbol{L}$ and an upper triangular matrix $\boldsymbol{U}$, i.e.,

$$\boldsymbol{P} = \boldsymbol{L}\boldsymbol{U}. \tag{3}$$

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

$$\begin{aligned} &u_{i,j} = 0, \quad i > j \\ &u_{1,j} = p_{t_1,j}, \quad j = 1, 2, ..., n \\ &u_{r,j} = p_{t_r,j} - \sum_{k=1}^{r-1} l_{r,k} u_{k,j}, \quad r = 2, ..., n,\ j = r, ..., n \end{aligned} \tag{4}$$

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

$$\begin{aligned} &l_{i,j} = 0, \quad i < j \\ &l_{i,1} = \frac{p_{t_{i,1}}}{p_{t_1,1}}, \quad i = 1, 2, ..., n \\ &l_{i,q} = \frac{p_{t_{i,q}} - \sum_{k=1}^{q-1} l_{i,k} u_{k,q}}{u_{q,q}}, \quad q = 2, ..., n,\ i = q, .., n \end{aligned} \tag{5}$$

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:

$$\boldsymbol{L}\boldsymbol{y} = \overline{\boldsymbol{\mathcal{P}}}, \tag{6}$$

$$\boldsymbol{U}\boldsymbol{k} = \boldsymbol{y}. \tag{7}$$

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

$$y_j = \overline{\mathcal{P}}_{t_j} - \sum_{q=1}^{j-1} l_{j,q} y_q \tag{8}$$

for $j > 1$. 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: $k_n = \frac{y_n}{u_{n,n}}$, and

$$k_j = \frac{y_j - \sum_{p=j+1}^{n} u_{j,p} k_p}{u_{j,j}}. \tag{9}$$

for $j < n$.

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. Before the algorithm can begin, the collector must number all the SMs from 1 to n. The corresponding index, j, along with $\overline{\mathcal{P}}_{t_{j+1}}$, will be transmitted to each SM_j. This information will allow the users 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 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: **for** $q = 1$ **to** j **do**
3: Compute $u_{q,j}$ using (4)
4: **end for**
5: **for** $q = j + 1$ **to** n **do**
6: Compute $l_{q,j}$ using equation (5)
7: **end for**
8: **if** $j < n$ **then**
9: Compute y_{j+1} using (8)
10: Transmit columns $1, 2, ..., j$ of $\boldsymbol{L}$ and all known elements of $y_1, ..., y_{j+1}$ only to SM_{j+1}
11: **end if**
12: **if** $j = n$ **then**
13: Notify the collector that $\boldsymbol{L}$, $\boldsymbol{U}$, and $\boldsymbol{y}$ are available
14: **end if**
15: **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. At SM_j ($1 < j < n$), it receives y_1 to 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 to 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 (8), starting from SM_n. In particular,

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 (9) 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**

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 (9), 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 power 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 power 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.

TABLE I
RECEIVED AND COMPUTED VALUES BY EACH SM

SM_i	Received	Computed
SM_1	$\overline{\mathcal{P}}_{t_2}, y_1$	$u_{1,1}, l_{1,1}, l_{2,1}, l_{3,1}, l_{4,1}, y_2$
SM_2	$\overline{\mathcal{P}}_{t_3}, l_{1,1}, l_{2,1}, l_{3,1}, l_{4,1}, y_1, y_2$	$u_{1,2}, u_{2,2}, l_{2,2}, l_{3,2}, l_{4,2}, y_3$
SM_3	$\overline{\mathcal{P}}_{t_4}, l_{1,1}, l_{2,1}, l_{3,1}, l_{4,1}, l_{2,2}, l_{3,2}, l_{4,2}, y_1, y_2, y_3$	$u_{1,3}, u_{2,3}, u_{3,3}, l_{3,3}, l_{4,3}, y_4$
SM_4	$l_{1,1}, l_{2,1}, l_{3,1}, l_{4,1}, l_{2,2}, l_{3,2}, l_{4,2}, l_{3,3}, l_{4,3}, y_4$	$u_{1,4}, u_{2,4}, u_{3,4}, u_{4,4}$

2) One Example for LUD: Next, we take a simple example to illustrate how to find the honesty coefficient vector, $\boldsymbol{k}$, using the LUD algorithm.

We consider a network of four SMs and one collector. Before Procedure 1 can start running at each SM, two things must be done. First, the collector, with itself being SM_0, assigns an integer index j in the range [1, 4] to each of the SMs in the neighborhood, and transmits $\overline{\mathcal{P}}_{t_{j+1}}$ to SM_j where $1 \leq j \leq 3$. Second, each SM_j, including SM_0, records the necessary energy consumption according to a starting time and a sampling period SP specified by the collector. After the collector chooses four time instances, at which the data collected will be used to form an LSE, it then instructs the SMs to start running Procedure 1.

Specifically, SM_0 (the collector) first calculates $y_1 = \overline{\mathcal{P}}_{t_1}$, and transmits this value to SM_1, which computes $u_{1,1}$, $l_{1,1}$, $l_{2,1}$, $l_{3,1}$, $l_{4,1}$, and y_2. SM_1 then transmits the first column of $\boldsymbol{L}$, y_1, and y_2 to SM_2. With these data, SM_2 is able to compute $u_{1,2}$, $u_{2,2}$, $l_{2,2}$, $l_{3,2}$, $l_{4,2}$, and y_3, and then transmits the first two columns of $\boldsymbol{L}$ and y_1 to y_3 to SM_3. SM_3 can thus compute $u_{1,3}$, $u_{2,3}$, $u_{3,3}$, $l_{3,3}$, $l_{4,3}$, and y_4, and transmit the first three columns of $\boldsymbol{L}$ and y_4 to SM_4. Finally, SM_4 can compute $u_{1,4}$, $u_{2,4}$, $u_{3,4}$, and $u_{4,4}$. Table I shows the information that each node receives and computes in order to carry out the LU decomposition. Note that SM_j records column j of the energy consumption data matrix $\boldsymbol{P}$ and no energy consumption recorded at other SMs are needed.

Moreover, once $\boldsymbol{U}$ is computed, all that is left is to solve the system $\boldsymbol{Uk} = \boldsymbol{y}$ using Backward Substitution. To this end, the collector instructs the SMs to start running Procedure 2. In particular, SM_4 starts the backward substitution process by calculating $k_4 = \frac{y_4}{u_{4,4}}$. Then, SM_4 computes based on k_4 the values needed by the other SMs, i.e., $u_{3,4}k_4$, $u_{2,4}k_4$, and $u_{1,4}k_4$, and transmits them to SM_3. SM_4 will also compute $E(k_4)$, its encrypted honesty coefficient, and send it to the collector. Similarly, the other three SMs calculate their own honesty coefficients based on the received quantities, and then encrypt and send them to the collector as well.

B. The LUPD Algorithm

As mentioned before, LUD may not be numerically stable

Procedure 3 LU Decomposition with Partial Pivoting

Input: $j \rightarrow SM_j$

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 jth element of row j with the jth 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**

when n is large. Here, we propose another algorithm, i.e., LUD with partial pivoting (LUPD), 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, LUPD decomposition has the following form:

$$\boldsymbol{EP} = \boldsymbol{LU}, \tag{10}$$

where $\boldsymbol{E}$ is the permutation matrix.

The LUPD 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 computes the first column of $\boldsymbol{U}$. SM_1 also computes the first column of $\boldsymbol{L}$, and transmits the first column together with the pivot index of column 1 to SM_2. Note that in LUPD, 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

Procedure 4 Forward Substitution

Input: $j \rightarrow SM_j$

1: **if** $j = 0$ or SM_j receives packets from SM_{j-1} **then**
2: **if** $j = 0$ **then**
3: Compute $y_1 = \overline{\mathcal{P}}_{t_1}$ and transmit y_1 to SM_1
4: **end if**
5: **if** $1 \leq j \leq n - 1$ **then**
6: Compute y_{j+1} as described in (8) 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**

[16]. 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 computes 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 $1 \leq j \leq n$.

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 sent 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 (6) and (7), respectively. In particular, since $\boldsymbol{y}$ is computed according to (8), $\boldsymbol{y}$ 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 (8), 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 $\boldsymbol{k}$.

Furthermore, the same as that in LUD, we can enable each neighboring two smart meters, i.e., SM_j and SM_{j+1} where $1 \leq j \leq n - 1$, 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, LUPD takes more time to complete. This is because in LUPD 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, LUPD is numerically stable while LUD is not.

C. Variable Honesty Coefficients

In the above LUD and LUPD algorithms, we have only considered that the honesty coefficient vector $\boldsymbol{k}$ is a constant. 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 an adaptive algorithm to address this problem.

We notice that when $\boldsymbol{k}$ changes in an LSE, the proposed LUD and LUPD 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. Thus, when the collector runs LUD/LUPD 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:

$$P(X = Y) = \binom{n}{Y} p^Y (1-p)^{n-Y}. \tag{11}$$

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 [17]:

- 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

$$P(X \geq Y) < e^{-\mathbb{E}[X] f(\delta)} \text{ with } \delta = Y/\mathbb{E}[X] - 1 \tag{12}$$

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

$$P(X \leq Y) < e^{-\frac{1}{2}\delta^2 \mathbb{E}[X]} \text{ with } \delta = 1 - Y/\mathbb{E}[X] \tag{13}$$

when $Y < E[X]$. Besides, when $Y = E[X]$, we set $P(X = Y) = 1$[5].

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/LUPD 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/LUPD algorithm in Procedure 5, which can detect illegal users with variable honesty coefficients.

Procedure 5 Adaptive LUD/LUPD Algorithm

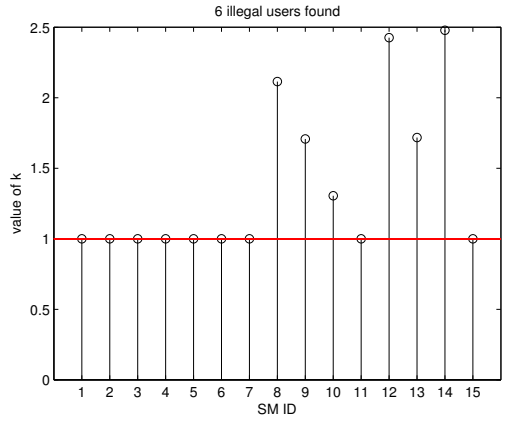
1: **repeat**
2: The collector instructs all SMs to take n samples with a initial sampling period SP
3: Run the LUD/LUPD algorithm
4: **if** The collector receives all elements in $\boldsymbol{k}$ **then**
5: $Y =$ the number of elements in $\boldsymbol{k}$ unequal to 1, i.e., the number of illegal SMs
6: **end if**
7: The collector calculates the probability that there are Y illegal users according to (11) or (12) or (13), 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

V. Simulation Results

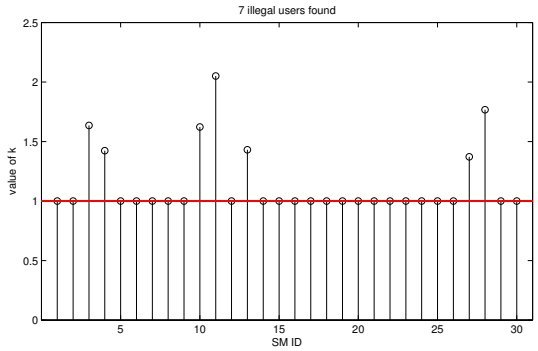
We perform two series of simulations to evaluate the performance of our privacy-preserving energy theft detection algorithms LUD and LUPD. In the first part, we assume that illegal users steal energy at a constant rate and thus have constant honesty coefficients. In the second part, we consider that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

Besides, we generate users' power consumption data, $\boldsymbol{P}$, based on a set of data from [18] and [19]. 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.

[5]As shown in Procedure 5, by setting $P(X = Y) = 1$ in this case, we will run the LUD/LUPD/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.

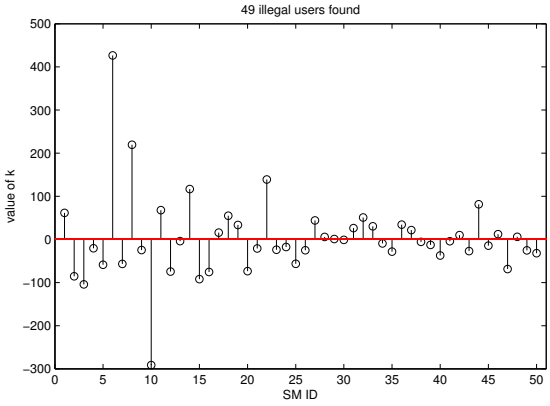


(a) 15 users

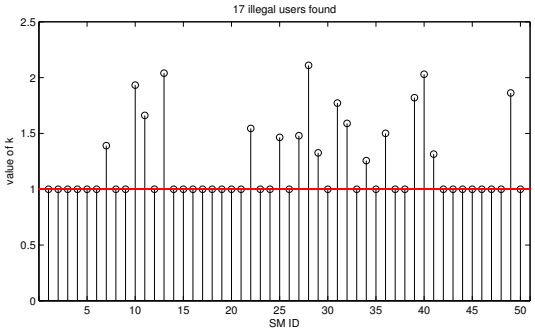


(b) 30 users

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



(a) LUD



(b) LUPD

Fig. 3. Elements of $\boldsymbol{k}$ obtained by the LUD and LUPD algorithms.

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 a 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.

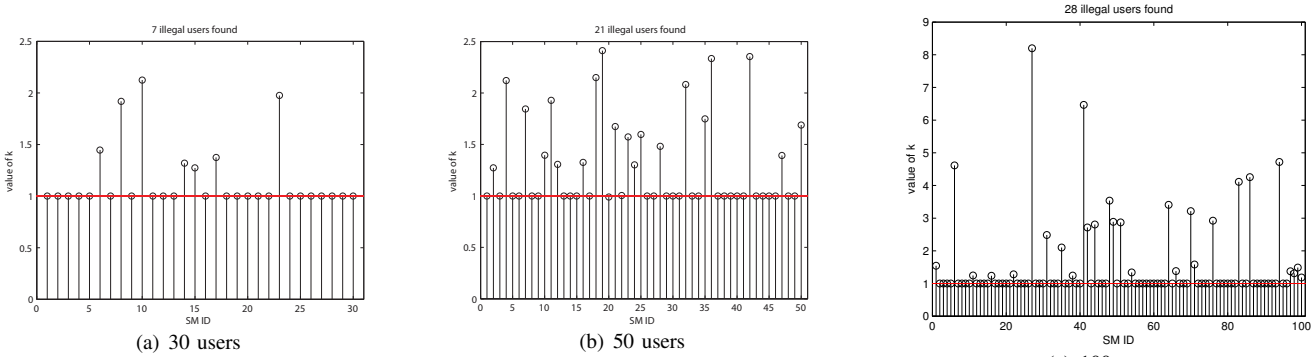
B. Variable Honesty Coefficients

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 algorithm in Fig. 4, when the number of users is equal to 30, 50, and 100, 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 algorithm in Fig. 5, in the cases that the number of users is equal to 30, 50, and 100, respectively. We can find that in these cases, the adaptive LUDP algorithm can also successfully and efficiently identify those fraudulent users.

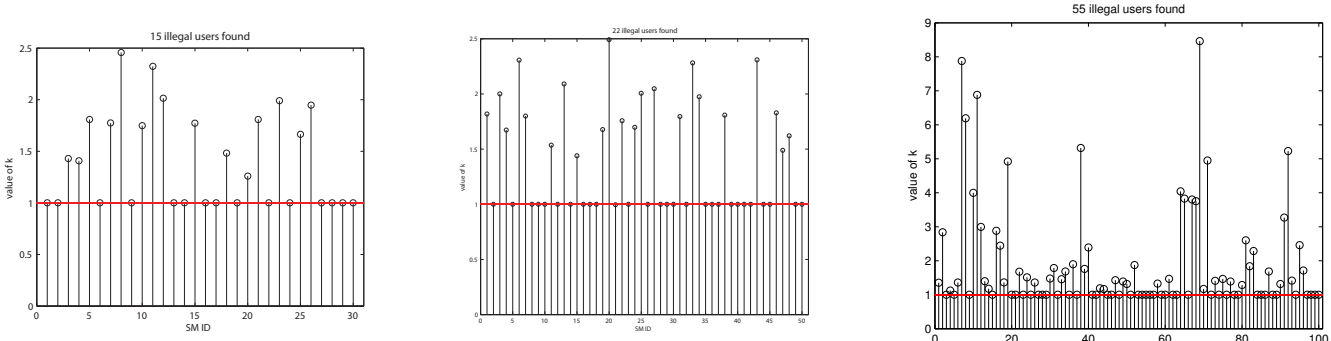
VI. Conclusion

In this paper, we have presented two algorithms, i.e., LUD and LUPD, which can identify the users who are committing energy theft in smart grids while preserve all users' privacy.



(a) 30 users (b) 50 users (c) 100 users

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



(a) 30 users (b) 50 users (c) 100 users

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

The two algorithms are distributed and based on LU decomposition. We can observe that no private data from the users needs to be transmitted to other users or to the collector, thus preserving users' privacy. We also find that LUD might not be stable in large-size networks while LUPD can. Extensive simulation results show that fraudulent users can be detected both when they commit energy theft at a constant rate and when they steal energy at variable rates.

REFERENCES

[1] 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.

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

[3] J. Nagi, K. Yap, S. Tiong, S. Ahmed, and A. Mohammad, "Detection of abnormalities and electricity theft using genetic support vector machines," in *Proceedings of the IEEE Region 10 Conference*, Hyderabad, India, November 2008.

[4] S. Depuru, L. Wang, and V. Devabhaktuni, "Support vector machine based data classification for detection of electricity theft," in *Proceedings of the Power Systems Conference and Exposition (PSCE)*, Phoenix, Arizona, USA, March 2011.

[5] 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 *Proceedings of the Transmission and Distribution Conference and Exposition*, Dallas, Texas, USA, September 2003.

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

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

[8] F. Li, B. Luo, and P. Liu, "Secure information aggregation for smart grids using homomorphic encryption," in *Proceedings of IEEE SmartGridComm*, Gaitherburg, Maryland, USA, October 2010.

[9] S. Abdelhak, A. Abdelgawad, S. Ghosh, and M. Bayoumi, "A complete scheme for distributed lu decomposition on wireless sensor networks," in *Proceedings of the 53rd IEEE International Midwest Symposium on Circuits and Systems*, Seattle, Washington, USA, August 2010.

[10] A. K. Zille Huma Kamal, Ajay Gupta Leszek Lilien, "Classification using efficient lu decomposition in sensornets," in *Proceedings of WSN*, Banff, Alberta, Canada, July 2006.

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

[12] S. Depuru, L. Wang, and V. Devabhaktuni, "A conceptual design using harmonics to reduce pilfering of electricity," in *Proceedings of the Power and Energy Society General Meeting*, Minneapolis, Minnesota, USA, July 2010.

[13] M. LeMay and C. A. Gunter, "Cumulative attestation kernels for embedded systems," in *Proceedings of the 14th European conference on Research in computer security*, Saint Malo, France, September 2009.

[14] M. Oliveira and A. Padilha-Feltrin, "A top-down approach for distribution loss evaluation," *IEEE Transactions on Power Delivery*, vol. 24, no. 4, pp. 2117–2124, October 2009.

[15] E. Kreyszig, *Advanced engineering mathematics*. Wiley, 1993.

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

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

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

[19] "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