Secure and Private Data Aggregation for Energy Consumption Scheduling in Smart Grids

Mohammad Ashiqur Rahman, Mohammad Hossein Manshaei, Ehab Al-Shaer, and Mohamed Shehab

Abstract—The recent proposed solutions for demand side energy management leverage the two-way communication infrastructure provided by modern smart-meters and sharing the usage information with the other users. In this paper, we first highlight the privacy and security issues involved in the distributed demand management protocols. We propose a novel protocol to share required information among users providing privacy, confidentiality, and integrity. We also propose a new clustering-based, distributed multi-party computation (MPC) protocol. Through simulation experiments we demonstrate the efficiency of our proposed solution. The existing solutions typically usually thwart selfish and malicious behavior of consumers by deploying billing mechanisms based on total consumption during a few time slots. However, the billing is typically based on the total usage in each time slot in smart grids. In the second part of this paper, we formally prove that under the per-slot based charging policy, users have incentive to deviate from the proposed protocols. We also propose a protocol to identify untruthful users in these networks. Finally, considering a repeated interaction among honest and dishonest users, we derive the conditions under which the smart grid can enforce cooperation among users and prevents dishonest declaration of consumption.

Index Terms—Smart Grid, Energy Consumption Schedule, Security and Privacy, Game Theory.

1 Introduction

Energy is critically important for residences and factories. With the booming of the population and the need of electrical energy, increasing efficiency becomes an important issue. A recent report from U.S. Department of Energy [1] states that, in the Unites States, almost two-fifths of the total electricity is consumed in households. However, the energy use is not efficient; the distribution of energy consumption rate in different hours of the day is not even. The peak usage of electricity is much higher than the off-peak periods. The peak value of electricity consumption data is extremely important for electric companies as the generation capacity of their power plants must be higher than the peak value. If some loads from the peak-periods can be shifted to the off-peak periods, the power company would be benefited by the reduction of the cost of power generation. Controlling the energy usage at the customer side of smart meters has received a lot of attention. Some research (e.g., [2], [3], [4]) has been made to minimize the cost of production with the indirect interaction between the energy users (i.e., the customers) and the energy provider giving incentive for using energy at off-peak hours.

The advent of smart meters gives the opportunity of two-way communication between the meters and the utility servers through the intelligent collectors [5]. This opportunity allows the researchers to rethink the optimal demand side management, which is also known as *demand response*. A direct load control solution for demand response is proposed in [6], where the utility remotely controls energy consumption for high-load household appliances like air-conditioners and water heaters. In [7] and [8], electricity scheduling methods are proposed to reduce the peak-to-average ratio (PAR) of the energy usage by introducing some flexible electricity price functions. These methods depend on the response of the users to the time-differentiated prices by shifting their load from the peak hours to the off-peak hours. These research works focus on the household users, particularly the household appliances, which are flexible in their usage time; hence one can shift the usage time from peak time to off-peak time to reduce the cost.

Mohsenian-Rad et al. in [9] proposed an autonomous and distributed demand-side energy management system among users that takes advantage of the communication infrastructure among the smart meters. The game-theory [10] is applied to formulate the energy consumption scheduling problem, solution to which gives the maximum payoff to the users. Similar automated demand side management systems are also proposed in [11], [12], [13], [14], [15]. In these works, demand response solutions typically optimize the overall cost of power generation and, thereby, the usage cost of the customers.

The distributed algorithm for the optimization of the energy consumption schedule requires a user to broadcast his hourly usage information to the other participating users; this algorithm interferes with the *privacy* problem. As the participating users possess various characteristics and they are mostly unknown to each other, privacy is an important matter. The algorithm is also susceptible to false data injection and replay attacks. Due to these attacks, the optimization algorithm can come up with a result which

- *M. A. Rahman, E. Al-Shaer, and M. Shehab are with the Department of Software and Information Systems, University of North Carolina at Charlotte, North Carolina, USA. Emails: {mrahman4,ealshaer,mshehab}@uncc.edu*
- *M. H. Manshaei is with the Department of Electrical and Computer Engineering, Isfahan University of Technology (IUT), Isfahan, 84156-83111, Iran. Email: manshaei@cc.iut.ac.ir*

Manuscript received July 9, 2014; revised February 20, 2015; accepted May 22, 2015.

is different from the actual optimal result. In this case, the participating users after optimization may not get the expected benefit; rather, they can end up paying much more than the regular price.

Moreover, some participating users may lie (i.e., defect) about their energy consumption behaviors. Though such defection will not give the global optimal benefit, one might be motivated to defect if he can get better payoff than being truthful. In terms of complexity, the algorithm is not time-efficient, as the algorithm requires a user to communicate all other users repeatedly until the global optimization is reached. Therefore, with the number of users, the time required for the convergence increases rapidly. In this paper, we propose some novel mechanisms for the distributed customer-side demand management in order to thwart security and privacy attacks by malicious outsiders or insiders in smart grids. We propose a mechanism for optimizing energy cost that meets the security challenges and performs efficiently that is developed on top of the typical energy consumption scheduling model. Our contributions in this paper are fivefold[1]:

- First, we propose an efficient *secure multi-party computing* (MPC) solution to preserve the privacy and security of the usage schedules. We have adopted the energy consumption scheduling model and management protocol proposed in [9] as a use case.
- Second, we enhance the efficiency of the distributed demand management protocol by clustering the participants and executing the optimization protocol over the clusters.
- Third, we demonstrate a scenario wherein a participating user can benefit by telling lies about its usage, if the price of electricity is computed based on consumers' usage in each slot (we call it *per-slot billing* mechanism in this paper). We also formally prove the incentive to deviate in such a scenario. Then we discuss assumptions for which there is no incentive for defecting. We also devise a truthful verifier-based solution in order to ensure the truthfulness of the participating users, where the verifier could be one of the users.
- Fourth, considering repeated interactions among users, we derive the conditions under which we can enforce cooperation among users and prevents dishonest declaration of consumption by a simple tit-for-tat mechanism.
- Finally, we evaluate the scalability and efficiency of our solution by executing simulation experiments.

The rest of this paper is organized as follows. We briefly discuss the demand management problem and its formalization in Section 2. We present our proposed solution for privacy and secrecy in Section 3. In Section 4, we present an incentive design to make scheduling protocol robust again selfish behaviors. We present the simulation experiments and evaluation results in Section 5. In the following section, we discuss some points regarding our proposed solution. We briefly present related work in Section 7 and conclude the paper in Section 8.

1. An early versiion of this work has been presented in IEEE TRUSTCOM 2012 [16], including the first and second contributions.

2 System Model

The main objective of this work is to formulate a secure and private mechanism for distributed protocols, in which the participating users need to share their private information to execute the protocols, while the integrity of the information is vulnerable to internal and external adversaries. To achieve this objective, in this work we consider the energy consumption scheduling model, in which users participate in a distributed protocol to optimize the energy usage cost. We believe that our proposed solution can easily be deployed within other distributed scheduling protocols in smart grids, where the scheduler needs the cooperation of smart meters.

2.1 Energy Consumption Model

In our model, the energy source provides the energy to the users by power lines. Each user is equipped with a smart meter. The smart meters are connected through the power line or Wi-Fi communication media, which forms a Local Area Network (LAN). The energy provider is connected to this LAN through the Wide Area Network (WAN). Each smart meter has an *energy consumption scheduling* (ECS) unit, which is capable of doing some arithmetic computation or processing.

The electricity cost function is defined as $C_h(L_h)$, where L_h is the hourly consumption of electricity and $C_h(.)$ returns the cost of consumed electricity at time $h \in \mathbb{H}$. $\mathbb{H}$ denotes the set of the time slots in a day. The cost per unit of electricity usage can be different at different times, which depends on the total electrical usage of all the users. The cost equation is a strictly increasing function as follows:

$$C_h(b) < C_h(c),\ \forall\, b < c$$

The power generation cost is often represented using convex functions [17]. In the case of thermal power generators, a quadratic cost function is considered. In order to reflect the generation cost accurately while charging the customers for their energy usage, the cost function is often considered as strictly convex [9], [18], [19]. Such functions indirectly motivate the customers to use electricity during off-peak hours. The convex function is represented as follows:

$$C_h(\theta b + (1-\theta)c) < \theta C_h(b) + (1-\theta)C_h(c),$$
$$\forall\, b < c,\ and\ \forall\theta \in (0,1)$$

The set of users are denoted as $\mathbb{N}$. The number of users is denoted by N (i.e., $N = |\mathbb{N}|$). The vector $u_x = [u_x^1, u_x^2, ..., u_x^{24}]$ denotes the usage vector for a user $x \in \mathbb{N}$ in 24 hours (i.e., $\mathbb{H} = \{1, 2, ..., 24\}$), where u_x^h denotes the hourly consumption of x at the time h. Hence, the summation of the usage vectors of all the users is denoted as follows:

$$u = [u^1, u^2, ..., u^{24}] = \left[\sum_{x\in\mathbb{N}} u_x^1, \sum_{x\in\mathbb{N}} u_x^2, ..., \sum_{x\in\mathbb{N}} u_x^{24}\right]$$

The summation is also a vector, which is named as the *summation usage vector*. The total cost of all the users is defined as $\sum_{h\in\mathbb{H}} C_h(u^h)$, where the u^h is the hourly usage of all users at time $h \in \mathbb{H}$.

For each user $x \in \mathbb{N}$, we define the set of household appliances as $\mathbb{A}_x$. Each appliance $a \in \mathbb{A}_x$ has an individual daily usage vector: $u_{x,a} = [u^1_{x,a}, u^2_{x,a}, ..., u^{24}_{x,a}]$. Note that an appliance needs a specific period of operation. Some appliances have fixed time slots of operation, while many others are flexible within some boundaries. The usage vector of a user x is the summation of the usage vectors of all appliances:

$$u_x = [u^1_x, u^2_x, ..., u^{24}_x] = \left[\sum_{a \in \mathbb{A}} u^1_{x,a}, ..., \sum_{a \in \mathbb{A}} u^{24}_{x,a}\right]$$

The peak-to average ratio (PAR) is defined as follows:

$$PAR = \frac{max_{h \in \mathbb{H}} u^h}{\sum_{h \in \mathbb{H}} u^h / |\mathbb{H}|}$$

2.2 Energy Consumption Scheduling

Our *energy consumption scheduling* problem is similar to the game presented in [9], where players are the registered users in set $\mathbb{N}$. The strategy of each user x is its energy usage vector u_x to maximize its payoff considering the other users strategies (i.e., u_{-x}). The payoff function is defined as:

$$P_x(u_x; u_{-x}) = -\frac{\sum_{h \in \mathbb{H}} u^h_x}{\sum_{x' \in \mathbb{N}} \sum_{h \in \mathbb{H}} u^h_{x'}} \sum_{h \in \mathbb{H}} C_h(\sum_{x' \in \mathbb{N}} u^h_{x'}) \quad (1)$$

It has been proved that once the game has no change, the game reaches to the Nash equilibrium point [9], where no user will get benefit by choosing other scheduling. It is worth mentioning that, while maximizing the payoff, the appliances are scheduled such that their operational needs are satisfied within the bounded time frames.

A distributed algorithm is executed between the participating users by the corresponding ECS units. In the rest of the paper, we will use the word *node* to represent a user, specifically the ECS unit corresponding to the user. Each node x randomly executes the local optimization (i.e., solving Equation (1)). To execute the local optimization, a node needs the summation of all nodes' usages. In order to have the summation, each node (x) needs to broadcast its usage vector (u_x) to others in the beginning and when any changes made to the usage vector. The algorithm converges to the optimal point, until there is no more broadcasting of usage vectors for a while. The nodes are supposed to be honest and declare their required amount of consumption.

The users are charged according to Equation (1). This means that the billing is a function of total consumption of users. However, since the electricity price is different at different time slots, the *per-slot* based billing mechanism provides fairness among users, where the payoff function of a user x is defined as follows [19]:

$$P_x = -\sum_{h \in \mathbb{H}} \frac{u^h_x}{\sum_{x' \in \mathbb{N}} u^h_{x'}} C_h(\sum_{x' \in \mathbb{N}} u^h_{x'}) \quad (2)$$

In the rest of the paper, we refer to these above mentioned billing mechanisms (i.e., Equation (1) and Equation (2)) as *total-consumption based mechanism* and *per-slot based mechanism*, respectively. We believe that the latter mechanism is more acceptable in smart grid billing systems, since with a *per-slot based mechanism*, the smart grid can charge users based on their real consumption in any given slot. In the next section, we address the security and privacy threats in the above algorithm and discuss potential solutions.

3 Data Privacy and Secrecy in Smart Grid

The distributed algorithm requires the broadcasting of the usage vector by each node. This broadcasting message is also sent as a plain text. The plain text introduces the privacy problem along with the following security problems: (i) the usage vector of a user can be eavesdropped on by listening to the communication media, and (ii) a malicious person can inject false data in order to cause the optimization to fail. In this particular work, we primarily assume external malicious attackers who can eavesdrop or inject false data. Within the participating nodes, we assume a semi-honest adversary model [20]. That is, the participating nodes follow the protocol, although some of them may be interested to know about others' usage pattern.

We propose a secure multi-party computing (MPC) algorithm to resolve the privacy problem of sharing user data. In our algorithm, a computing node runs the MPC algorithm to receive the summation of the usage vectors of all other nodes before performing each local optimization. Although any suitable MPC solution [21] could be adopted, we emphasize the computational efficiency of the algorithm by proposing a simple and light-weight protocol for secure MPC, which is very important for the low processing power devices like smart meters. In our approach, we mainly apply randomization to maintain privacy. In order to prevent eavesdropping (i.e., confidentiality) as well as the injection of false data (i.e., authenticity), we use cryptography.

A node requires the aggregation of the usage vectors of all the participating nodes for running local optimization. In our solution, we run the MPC first to get the summation of the usage vectors. An example execution of the technique is shown in Fig. 1 for three nodes. The execution of MPC algorithm is sequential like a ring: all the participating nodes create a logical ring starting from the computing node and ending to the same node. For a particular node, the logical ring is usually different at different executions of local optimization, because it is randomly selected at the time of execution.

The computing node (e.g., $x0$ as in the figure) starts MPC by launching a message containing its usage vector u_{x0} added with a random usage vector R and its identity. The message is sent to an arbitrarily selected node $x1$. The message is encrypted using the private key of $x0$ (i.e., *pr-x0*), which follows by the encryption using the public key of $x1$ (i.e., *pb-x1*).

When the node $x1$ receives the message from $x0$, it decrypts the message. It receives the summed usage vector (here $R + u_{x0}$) and the nodes (here only $x0$) which have added their usage vectors to the summation. Now it adds its usage vector with $R + u_{x0}$. Then $x1$ encrypts the resultant usage vector first using its private key (i.e., *pr-x1*) and second using the public key of $x2$ (i.e., *pb-x2*). It sends a message comprised of the encrypted usage vector and the identities of the nodes, which have added their usage to the node $x2$ that is arbitrarily selected from the remaining nodes (i.e., whose usage vectors are yet to be added). In this way, the message is received by a node after which there is no node remaining to add to the summation. This node (here $x2$ according to the figure) adds its usage vector to the summation and sends the message to the MPC initiating

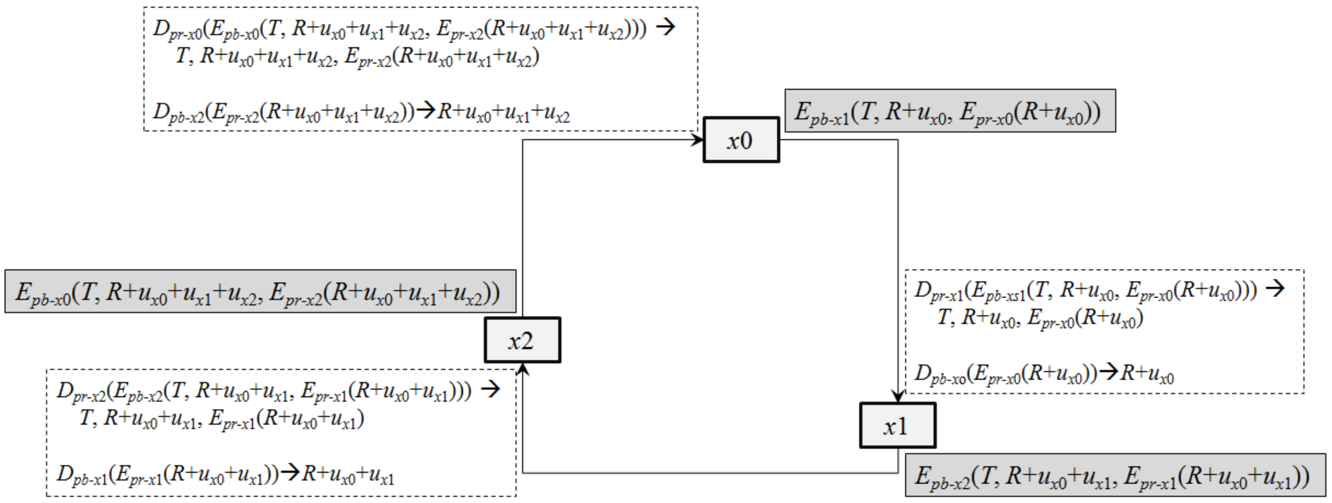

$D_{pr\text{-}x0}(E_{pb\text{-}x0}(T, R+u_{x0}+u_{x1}+u_{x2}, E_{pr\text{-}x2}(R+u_{x0}+u_{x1}+u_{x2}))) \rightarrow T, R+u_{x0}+u_{x1}+u_{x2}, E_{pr\text{-}x2}(R+u_{x0}+u_{x1}+u_{x2})$
$D_{pb\text{-}x2}(E_{pr\text{-}x2}(R+u_{x0}+u_{x1}+u_{x2})) \rightarrow R+u_{x0}+u_{x1}+u_{x2}$
$E_{pb\text{-}x1}(T, R+u_{x0}, E_{pr\text{-}x0}(R+u_{x0}))$
$x0$
$E_{pb\text{-}x0}(T, R+u_{x0}+u_{x1}+u_{x2}, E_{pr\text{-}x2}(R+u_{x0}+u_{x1}+u_{x2}))$
$x2$
$D_{pr\text{-}x1}(E_{pb\text{-}xs1}(T, R+u_{x0}, E_{pr\text{-}x0}(R+u_{x0}))) \rightarrow T, R+u_{x0}, E_{pr\text{-}x0}(R+u_{x0})$
$D_{pb\text{-}xo}(E_{pr\text{-}x0}(R+u_{x0})) \rightarrow R+u_{x0}$
$D_{pr\text{-}x2}(E_{pb\text{-}x2}(T, R+u_{x0}+u_{x1}, E_{pr\text{-}x1}(R+u_{x0}+u_{x1}))) \rightarrow T, R+u_{x0}+u_{x1}, E_{pr\text{-}x1}(R+u_{x0}+u_{x1})$
$D_{pb\text{-}x1}(E_{pr\text{-}x1}(R+u_{x0}+u_{x1})) \rightarrow R+u_{x0}+u_{x1}$
$x1$
$E_{pb\text{-}x2}(T, R+u_{x0}+u_{x1}, E_{pr\text{-}x1}(R+u_{x0}+u_{x1}))$

Fig. 1. The MPC technique for summing the usage vectors of the participating nodes that includes authentication and confidentiality security measures also.

Algorithm 1 Executed by each node $x \in \mathbb{N}$.

```
repeat
    if a random time instance to compute optimization then
        Execute Secure MPC to get the summation $S_{-x}$ of all the users' usage vectors.
        Solve local optimization Equation (1) of $u_x$.
    end if
until There is no significant update in $u_x$ for a number of last consecutive rounds.
```

node (i.e., $x0$) along with the list of participating nodes applying the same encryption steps onto the message. Note that at least three nodes are required to participate in the MPC technique. In the case of two nodes, the MPC initiating node can easily figure out the other node's usage vector. The distributed algorithm for local optimization of the usage vector executed by each node is shown in Algorithm 1. Each node continuously executes the local optimization at random time intervals. At each run, the MPC algorithm is executed to get the summation of the usage vectors of all users. A node stops executing any further round of the local optimization process when there is no significant update in u_x for a number of consecutive rounds.

3.1 Properties of the Proposed MPC Algorithm

In the following subsections, we briefly review the main features of the proposed secure and private MPC algorithm for energy consumption in smart grids.

3.1.1 *Importance of Applying Both Encryptions*

In our proposed MPC process, each node (e.g., $x0$) at first encrypts the message using its private key (e.g., *pr-x0*) and then encrypts the same using the public key of the receiving node (e.g., *pb-x1*). These two encryptions are not required for secure MPC execution, i.e., privacy preservation. If the first encryption is not performed, any malicious person can inject false data. If false data is injected, the summation will not be the actual data, and thus the optimal value will be incorrect. As a result, the nodes will not benefit from participating in the optimizing process. This encryption is known as *signing* the message.

In the case of second encryption, if the message is not encrypted by the receiving node's public key, anyone can eavesdrop on the message breaking the confidentiality. Though the message content (i.e., the summation usage vector and the participating nodes) does not allow an attack on the protocol, the MPC initiating node $x0$ can eavesdrop on the messages and decrypt them to find the usage vectors of the other nodes. In Fig. 1, $x0$ eavesdrops on the message sent by $x1$ to $x2$ and decrypts the message using the public key of $x1$ (as it is not encrypted using the $x2$'s public key). Hence, the node can easily figure out the usage vector of $x1$. Similarly, from the message sent by $x2$ to $x0$, the node gets the usage vector of $x2$ by subtracting the usage vector of $x1$. If there are more nodes, it can get the usage vectors of all the participating nodes in MPC by eavesdropping on all messages between nodes. The communication is also susceptible to replay attacks. The random usage vector (R) also helps to get rid of such attacks.

3.1.2 *Protection against Sandwich Attack*

One node randomly selects the next node in the logical sequence at each MPC execution; in the case of a particular node, the node sequence in the ring is usually different than that of a different MPC execution. Even the ring is not known when the execution runs. This random selection is done because, otherwise, a node can easily be sandwiched by the preceding node and by the subsequent node in the sequence in order to learn its usage vector.

The sandwich attack cannot be fully avoided by applying randomness; rather this randomness only makes the collusive attack less potential. In fact, the probability of successful sandwich attack would be $1/N$ (N is the number of participating users), when one of the colluding nodes always choosing the victim node as its next node. However, the victim node can easily detect this malicious behavior.

Moreover, increasing the number of participants, i.e., larger values of N, would decrease the chance of successful attack.

3.1.3 *MPC Executing at Each Local Optimization*

The execution of MPC at each optimization process increases the execution cost (i.e., time complexity) of our algorithm. However, it is possible to execute the MPC only at the beginning of the whole process, not at each local optimization process. In this case, the computing node requires broadcast of the updated summation usage vector to all other participating nodes. At the time of executing each local optimization, a node requires use of the latest summation usage vector, though this raises a privacy issue again. Comparing the broadcasted usage vector with the earlier vector, one node can understand the changes done by the computing node. That is, for example, if the summation usage vectors before and after the optimization done by a node are $U_A = [X_1, X_2, \ldots, X_N]$ and $U_B = [\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_N]$ respectively, then the difference of the two vectors, $\triangle U$ (where, $\triangle U = U_A - U_B = [\triangle X_1, \triangle X_2, \ldots, \triangle X_N]$) gives some hints about the usage vector of the computing node.

3.1.4 *Protect the List of Participants*

Here we consider a special scenario in which an external attacker compromises a participating node. Then, it makes the compromised node interrupt the MPC scheme by bypassing the next node, or simply returning the partially aggregated result to the MPC initiating node. It manipulates the list of participating nodes showing that each node has participated in this particular execution. As a result, the MPC scheme is corrupted without being detected by any other node leading to the optimization failure.

We devise a solution against such a list-manipulation attack. Along with the list, each node creates a signed message containing a sequence of three nodes: (i) the node itself (e.g., $x1$ in Fig. 1), (ii) the previous node in the ring ($x0$, from which it has received the accumulated result), and (iii) the next node in the ring ($x2$, to which it will send the updated accumulated result). It then sends the message directly to the initiating node or adds the message along with the list and the aggregated result during the MPC algorithm. When the MPC initiating node receives these messages, it can easily verify if all nodes have participated in the scheme. The node order in the signed message restricts the chance of replay attacks later, as this order is supposed to be random. However, a timestamp can replace this order in the message, if all nodes share a globally synchronized timing system.

3.2 Efficient Computing

The convergence cost of the optimization algorithm is high. The average time of the algorithm is $O(NM)$, where N is the number of participating nodes and M is the number of rounds required for the convergence. For a large N, M also becomes very high. As a result, the implementation of this algorithm is infeasible for large number of users. The main reason behind this cost is a large number of messages that are required to exchange until the convergence. The low computational capability of a smart meter is also important to consider. Each optimization round is expected to take a significant processing time. Therefore, this algorithm is not suitable for implementation on a large scale.

Usually numerous rounds of local optimizations are required to reach the global optimal point. Running MPC algorithm at the beginning of each local optimization process must increase the execution cost. The overhead of our local optimization algorithm is simply the summation of the cost of running MPC and the cost of optimization. In order to reduce the overhead we apply clustering, which reduces the number of participating nodes in an MPC execution. In our solution, we use mutually exclusive clusters as shown in Fig. 2. A number of clusters are formed among the participating nodes, where the intersection of any two clusters is an empty set. In this solution approach, a node executes MPC in its cluster only. Hence, the node is required to receive the summation of the usage vectors of the members of each of the other clusters. For the collection, the computing node chooses an arbitrary node from each cluster. The sum may not be the most updated one, as the selected node might not be the last that has executed the optimization. Hence, the summation found in this way may not be accurate, and as a result the optimization cannot be perfect. However, the possible error in the optimized value decreases as the algorithm converges. Algorithm 1 is modified for the cluster based solution in Algorithm 2. The

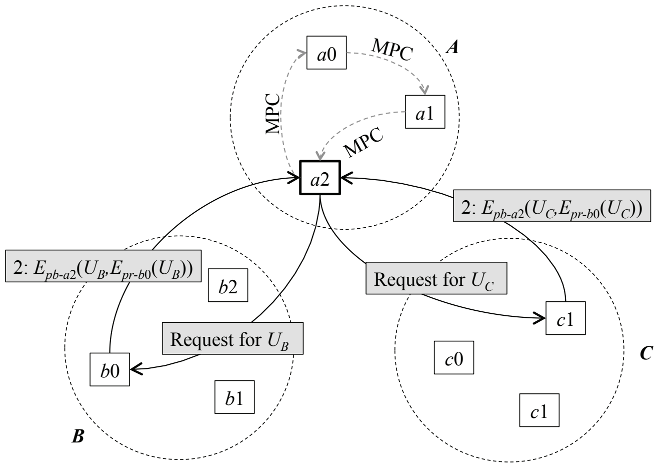

a0
MPC
A
MPC
a1
MPC
a2
2: $E_{pb\text{-}a2}(U_C, E_{pr\text{-}b0}(U_C))$
2: $E_{pb\text{-}a2}(U_B, E_{pr\text{-}b0}(U_B))$
b2
Request for U_C
c1
Request for U_B
c0
C
b0
b1
c1
B

Fig. 2. The cluster based MPC technique.

Algorithm 2 Executed by each node $x \in \mathbb{N}$.

```
repeat
    if a random time instance to compute optimization then
        Execute MPC to get the summation of usage vectors $S_C$ in its cluster $C$.
        Collect the summation of usage vectors $S_{-C}$ from all other clusters
        Add $S_C$ and $S_{-C}$ to get the total $S$
        Solve local optimization (following Equation (1)) of $u_x$.
        Update $S_C$ according to current $u_x$.
    end if
until There is no significant update in $u_x$ for a number of last consecutive rounds.
```

time complexity of this algorithm is basically $O((N/C)M)$, where N is the number of users, C is the number of clusters, and M is the number of iterations for the convergence.

4 Truthful Mechanism Design

In the case of the per-slot based billing mechanism, the distributed demand management protocol in [9] is vulnerable to another security problem, which is because of the untruthfulness of the participating nodes. A node can cheat about its usage vector to get advantage. In this section, we first illustrate this problem with an example. Then, we present a solution to detect an untruthful user. Finally, we also provide a game-theoretic solution that ensures the participating users to follow the protocol rightfully.

4.1 Benefit by Deviating from the Protocol

As we have already mentioned, we assume a semi-honest user model [20], i.e., a node may cheat passively (tell a lie about its information), but it always follows the protocol (i.e., the MPC and optimization procedure). Fig. 3 and 4 show an example of how a participating node can get benefit from lying. We consider three nodes (A, B, and C), along with their arbitrary usage vectors (three time slots in each vector) in the example. In Fig. 3, we show the expected scenario, where each participating node is truthful about its usage. In this case, we see that B pays \$336 for its electricity usage whether we apply the per-slot or the total-consumption based charging method. We follow a simple convex price function, $C(L) = 2L^2$ for each time slot, where L is the total consumption at a given time slot. In Fig. 4, we show the case when node B lies about its usage and pretends it would use 12kWh in the third time slot, though its actual intention is different (i.e., 6kWh). After the optimization, B is supposed to use 12kWh in the third slot. However, in practice, B uses 6kWh. As a result, when the smart grid charges with per-slot basis, the price at that slot reduces and B pays \$312 for its usage. Hence, B gets benefit from cheating. Note that the other nodes (i.e., A and C) will need to pay more than that in the case of the truthful scenario, whether they are charged based on per-slot (i.e., \$456) or total-consumption (i.e., \$437.14).

However, if the smart grid employs the total-consumption based billing mechanism (as it is shown in [9]), there would be no motivation for node B to deviate from the protocol as it should pay more (i.e., \$349.71) comparing to the case where the node declares the true value (i.e., \$336). Moreover, this false declaration causes a significant increase in the honest nodes' bills. We now formally prove that a malicious insider node can increase its payoff by lying about its consumption. Note that similar examples can be shown by other convex functions. We consider a simple convex function in order to avoid complex computations, so that it can be easy to grasp the presented example.

Theorem 1. *In an energy consumption scheduling algorithm for smart grid with per-slot based billing mechanism, a node can increase its benefit (get electricity with lower price) by declaring false information about its consumption while others' payoff will be decreased.*

	Usage Declaration			*Optimization Results*			*Billing [C(L)=2L²]*	
Time Slots	**Slot 1**	**Slot 2**	**Slot 3**	**Slot 1**	**Slot 2**	**Slot 3**	**Per Slot**	**Total**
User A	5	6	4	6	4	5	336	336
User B	2	4	6	3	3	6	336	336
User C	5	5	5	5	7	3	336	336

Fig. 3. Cost (price) of usage for 3 nodes during 3 slots when they are honest in usage declarations.

	Usage Declaration			*Optimization Results*			*Billing [C(L)=2L²]*	
Time Slots	**Slot 1**	**Slot 2**	**Slot 3**	**Slot 1**	**Slot 2**	**Slot 3**	**Per Slot**	**Total**
User A	5	6	4	7	6	2	456	437.14
User B *(Cheater)*	2	4	12	3	3	12	--	--
User C	5	5	5	6	7	2	456	437.14

User B *(Actual Usage)*	**3**	**3**	**6**	**Per Slot Bill= 312**	**Total Bill: 349.71**

Fig. 4. Node B lies about its consumption during the advertisement at the third slot and declares 12 instead of 6. In the per-slot based billing mechanism, this cheating gives benefits (312) to the node with respect to the honest case (336).

Proof. In this proof we assume that none of the nodes has restriction on the usage time slots of their appliances. That is, a node has freedom to use its appliance at different time slots. Therefore, if the total usage of all nodes can be distributed evenly over the time slots, each node's payoff will be maximized. Hence, we assume that the total usage is evenly distributed over the time slots after the optimization. Let L_T be the total load throughout all of the time slots and L_A be the load at each time slot:

$$L_A = \frac{L_T}{|\mathbb{H}|} = \frac{\sum_{h\in\mathbb{H}}\sum_{x'\in\mathbb{N}} u_{x'}^h}{|\mathbb{H}|}$$

In the per-slot based billing mechanism, payoff is computed using Equation (2), where $C_h(.)$ is a strictly increasing convex function. The total cost at each time slot is $C_h(L_A)$. Therefore, c_h, the cost of each unit of usage, is $c_h = C_h(L_A)/L_A$. Let us rewrite the payoff, given the definition of c_h:

$$P_x = -\sum_{h\in\mathbb{H}} u_x^h c_h$$

Now we assume that node $\bar{x}$ is a dishonest node and it pretends to use $\Delta U_{\bar{x}}^{\bar{h}}$ more (i.e., its total usage is $U_{\bar{x}}^{\bar{h}} + \Delta U_{\bar{x}}^{\bar{h}}$) during time slot $\bar{h}$ ($\bar{h} \in \mathbb{H}$), though its actual usage is still $U_{\bar{x}}^{\bar{h}}$. Hence, after the optimization process, the usage at each slot would be changed to $\bar{L}_A$:

$$\bar{L}_A = \frac{L_T + \Delta U_{\bar{x}}^{\bar{h}}}{|\mathbb{H}|} = L_A + \frac{\Delta U_{\bar{x}}^{\bar{h}}}{|\mathbb{H}|}$$

Since $\bar{L}_A > L_A$ and $C_h(.)$ is a strictly increasing function for each $h \in \mathbb{H}$, then we conclude that $C_h(L_A) < C_h(\bar{L}_A)$. If $\bar{c}_h$ is the per unit usage cost in this dishonest scenario, then

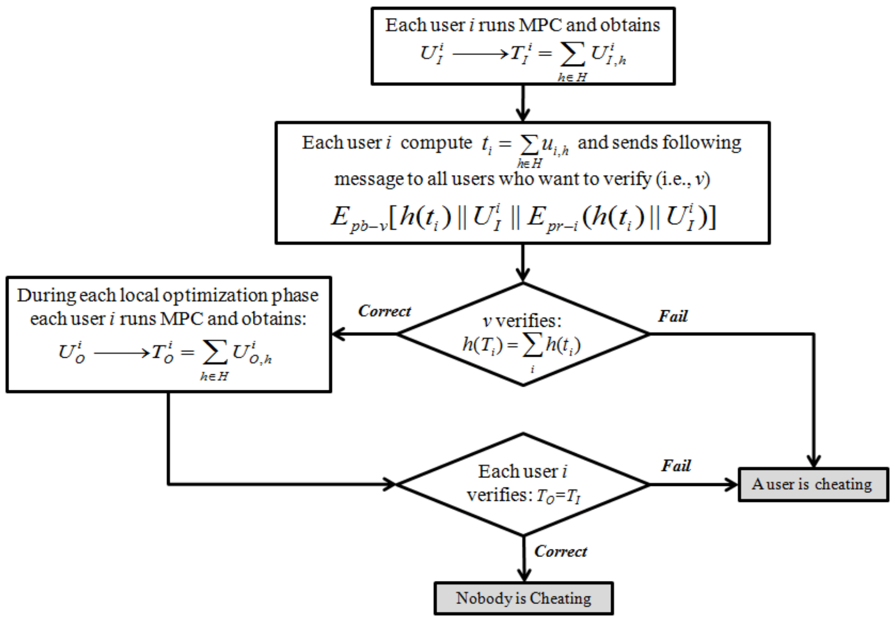

Each user i runs MPC and obtains $U_I^i \longrightarrow T_I^i = \sum_{h\in H} U_{I,h}^i$
Each user i compute $t_i = \sum_{h\in H} u_{i,h}$ and sends following message to all users who want to verify (i.e., v)
$E_{pb-v}[h(t_i) \,||\, U_I^i \,||\, E_{pr-i}(h(t_i)\,||\,U_I^i)]$
During each local optimization phase each user i runs MPC and obtains: $U_O^i \longrightarrow T_O^i = \sum_{h\in H} U_{O,h}^i$
Correct
v verifies: $h(T_I) = \sum_i h(t_i)$
Fail
Each user i verifies: $T_O = T_I$
Fail
A user is cheating
Correct
Nobody is Cheating

Fig. 5. The verification protocol to detect any deviation from the protocol, i.e., whether some node is lying during the execution of the optimal energy consumption protocol.

$\bar{c}_h > c_h$ for all $h \in \mathbb{H}$. Similarly, we can calculate the cost during the slot $\bar{h}$, considering the actual usage of dishonest node. Let us designate this by $\hat{c}_{\bar{h}}$:

$$\hat{c}_{\bar{h}} = \frac{C_h(\bar{L}_A - \Delta U_{\bar{x}}^{\bar{h}})}{\bar{L}_A - \Delta U_{\bar{x}}^{\bar{h}}}$$

Given the properties of the cost function $C_h(.)$, we can conclude that $\hat{c}_{\bar{h}} < c_{\bar{h}}$. Moreover, we know that $\bar{L}_A - \Delta U_{\bar{x}}^{\bar{h}} < L_A$ (easily derivable from the equation of $\bar{L}_A$). Hence we conclude that $\hat{c}_{\bar{h}} < c_h$. Now we calculate the new payoff of node x who has lied in declaring its consumption (i.e., $\bar{P}_{\bar{x}}$):

$$\bar{P}_{\bar{x}} = -(u_{\bar{x}}^{\bar{h}}\hat{c}_{\bar{h}} + \sum_{h\in\mathbb{H},h\neq\bar{h}} u_{\bar{x}}^{h}\bar{c}_h)$$

The dishonest node $\bar{x}$ is benefited from the lie, if $\bar{P}_{\bar{x}} > P_{\bar{x}}$. It is true, when the ratio of the usage at $\bar{h}$ and the usage at the rest of the slots is greater than the ratio of the increase of the cost at the slots other than $\bar{h}$ (i.e., $\bar{c}_h - c_h$) and the decrease of the cost at slot $\bar{h}$ (i.e., $\hat{c}_{\bar{h}}$), with respect to the honest scenario. That is:

$$\frac{u_{\bar{x}}^{\bar{h}}}{\sum_{h\in\mathbb{H},h\neq\bar{h}} u_{\bar{x}}^{h}} > \frac{\bar{c}_h - c_h}{c_h - \hat{c}_{\bar{h}}}$$

□

In Theorem 1, we consider the assumption of having no restriction on the usage time slots for an appliance. This assumption helps us understand the idea of distributing the energy usage within the (permitted) time slots for optimizing the energy cost.

4.2 Protocol to Detect Dishonest Node

Theorem 1 shows that a malicious node can increase its payoff by declaring false information about its consumption in per-slot based charging mechanism.

In the following, we provide a solution to the above problem. The next theorem shows that if a node z cannot lie about its total usage, it cannot benefit from lying about its usage vector (i.e., the potential usage distribution in different time slots).

Theorem 2. *If the declared total usage at the time of participation in the optimization process is equal to the actual usage, there is no incentive for lying about the usage vector.*

Proof. Let us assume that node $\bar{x}$ has lied about its usage vector (usage distribution only) at the time of optimization process. After completing the optimization process, $\bar{x}$ is using a different slot s for the load l, other than the slot $\bar{s}$ that it advertised (and used) at the time of optimization process. Based on the current optimal usage vectors, Let the total load of the slot s be L. Hence, the current load $\bar{L}$ is $L + l$, which eventually increases the electricity price rate of that slot from P to $\bar{P}$. If it would use the slot s for the load l at the time of optimization, the total load of the slot s, $\hat{L}$ would not be more than $L + l$. Therefore, $\hat{L} \leq L + l$. The reason behind the possibility of $\hat{L} < L+l$ is that, at that time of optimization, other competing nodes might move from s to a different slot, since they would find one more candidate $\bar{x}$ with load l for this slot. They might move to $\bar{s}$ as $\bar{x}$ would not be a candidate of load l for this slot. Therefore, there is no way for $\bar{x}$ to receive better payoff (i.e., less cost) by lying about its usage distribution only. □

Considering the results of Theorem 2, we design the following secure protocol to ensure whether any participating node has lied about its total usage. Fig. 5 and Fig. 6 illustrate how the protocol works.

Note that in this protocol, each participating node knows the summation of the usage vectors of all the participating nodes from the beginning to the end of the optimization process (when the process converges).

From the summation usage vector, each node knows the accumulated total usage of all the participating nodes (not individual total usage). Hence, throughout the optimization process no node can tell a lie about its total usage, as it will change the overall total usage. Each node knows the expected total load according to the optimal usage vector. We also assume that the power utility company provides the nodes with the usage report of each day, if the verifier needs to find a cheater. If required, one or more nodes are randomly selected as *verifiers*. These verifier nodes receive required information from the provider. In the following, we summarize the steps that should be taken to verify whether all nodes follow the protocol:

Dishonest Node Detection Mechanism:

1) At the beginning of computing local optimization, each node executes the following steps:

   a) Each node executes MPC to know the summation usage vector U_I^i of the usage vectors of all participating nodes. It can then compute the total usage $T_I = \sum_{h \in H} U_{I,h}$.
   b) Each node computes the hash value of its total usage, i.e., $h(t_i)$. The hash function is a one-way cryptographically secure and known to all nodes. We also assume that this hash function is homomorphic in nature, i.e., $h(x+y) = h(x) + h(y)$. Along with this hashed value, the node also sends the summation usage vector U_I^i to each node. It signs the message using its private key (i.e., E_{pr-i}) and sends it to the requesting node by encrypting it with the verifier public key (i.e., E_{pb-v}). This message acts as the *commitment* made by the node about its total usage.
   c) Receiving such a message, each verifier can compare $h(T_i)$ with the summation of declared consumption, i.e., $\sum_i h(t_i)$. If the verifier detects any difference, it will ensure that there exists at least a node that is cheating.

2) At the time of each local optimization, each node is required to compute the sum of the usage vectors T_O, which should be the same as T_I. Otherwise, it can detect the cheater.
3) After the period of consumption, as shown in Fig. 6, the provider sends the total consumed energy to the verifier. If the verifier finds the difference between the expected and the reported usage, the verifier initiates a verification. The verification works as follows:

   a) The provider sends all individual summation of consumed energy, i.e., $h(t_{i,c})$.
   b) The verifier uses the famous Yao's millionaire technique [22] to verify whether the individual consumption is bounded between $t_i - \theta$ and $\theta + \theta$, given that the verifier knows $h(t_{i,c})$ and $h(t_i)$. θ can be selected by verifier, considering the acceptable amount of deviation from the requested consumption.

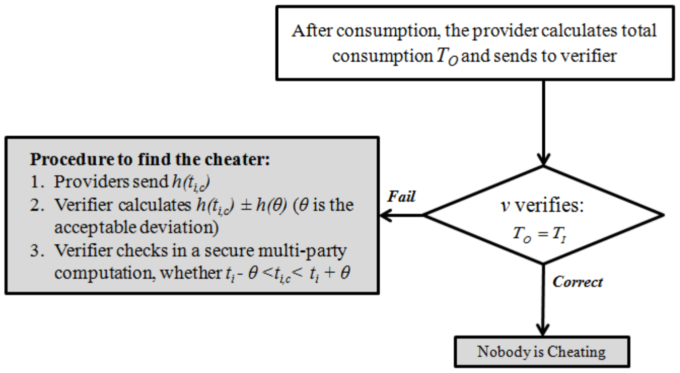

After consumption, the provider calculates total consumption T_O and sends to verifier
Procedure to find the cheater:
1. Providers send $h(t_{i,c})$
2. Verifier calculates $h(t_{i,c}) \pm h(\theta)$ (θ is the acceptable deviation)
3. Verifier checks in a secure multi-party computation, whether $t_i - \theta < t_{i,c} < t_i + \theta$
Fail
v verifies: $T_O = T_I$
Correct
Nobody is Cheating

Fig. 6. The verification protocol to find the cheating node, i.e., whether a node lied during the execution of the optimal energy consumption protocol. This protocol executes after the actual energy consumption.

The actual usage can be a little smaller or larger than the advertised usage, but the difference cannot be significantly high. Hence, as is shown in the protocol, a threshold value can be defined (i.e., θ) and the verification for equality can be done within the threshold value. There should be some penalties or punishments, (e.g., in terms of money), against the cheated nodes, as well as necessary reimbursements for the suffered nodes. In order to keep the privacy, this verification must be done by using the hash values of total consumption of each node [22].

4.3 A Game-theoretic Solution to Ensure Truthfulness

Considering the complexity of the protocol presented in the previous subsection, we propose a simple game-theoretic solution to ensure the truthfulness of participating nodes in the energy consumption scheduling protocol.

We model the interaction of honest and malicious nodes with an infinite repeated game [10], where the game will be played with a probability of $\delta \in (0, 1]$ in each time. Here, δ potentially have some impact on the period of subscription. Since, the players should interact and run optimization protocol over a long period of time, an infinite repeated game is a valid assumption. Let us assume that nodes can be divided into two main groups G_1 and G_2. We assume that the nodes in group G_1 are honest, while the nodes in group G_2 tend to be dishonest. In other words *honest* nodes in G_1 follow the optimization protocol whereas *dishonest* nodes in G_2 tends to deviate from the protocol and declare the wrong consumption and try to get more benefit by cheating. Any of the nodes in the second group can initiate an infinitely repeated game by deviating from the protocol. In such a case, the cheater node will play with all honest nodes in the second group.

Let us assume that the interaction between these two groups is modeled by a game as shown in Table 1. We designate the strategy set of nodes in G_1 by $S_1 = \{C, P\}$, where strategy C means that the nodes cooperate and follow the protocol and strategy P means that the nodes in group G_1 will punish nodes in G_2 by not declaring their consumption honestly.

Similarly the strategy set for nodes in G_2 is $S_2 = \{R, D\}$, where R means that the nodes in this group follow the protocol and would consequently be rewarded as the nodes

TABLE 1
Normal form of the game between Cooperative and Defective users.

$G_1 \backslash G_2$	R	D
C	α_1, α_2	γ_1, β_2
P	β_1, γ_2	λ_1, λ_2

in G_1 will also cooperate. Strategy D means that the nodes in G_2 defect and will not follow the protocol.

Note that when all nodes cooperate (i.e., (C, R)) they obtain better payoffs compared to the case when all nodes defect and do not follow the protocol (i.e., (P, D)). In other words, $\alpha_i > \lambda_i$, $i \in \{1, 2\}$.

As it is shown in Theorem 1, dishonest nodes in G_2 can increase their payoffs by deviating from the protocol and declaring false information (i.e., play D), i.e., $\beta_i > \alpha_i$, $i \in \{1, 2\}$. Consequently, honest nodes' payoffs decrease when other nodes do not cooperate with them, i.e., $\alpha_i > \gamma_i$, $i \in \{1, 2\}$.

In order to get an insight into the strategic behavior of honest nodes in a repeated interaction with other nodes, we define the following strategy:

Definition 1. *With the* grim trigger strategy *in repeated games, a player will cooperate, but as soon as the opponent defects (thus satisfying the trigger condition), the player using grim trigger will defect for the remainder of the iterated game.*

Let us assume that honest nodes use the *grim trigger strategy* when they interact with dishonest nodes. The following theorem shows that under certain conditions the cooperative behavior can be enforced by honest nodes.

Theorem 3. *Honest nodes can get cooperation of other nodes in the defined optimization protocol using the* ***Grim Trigger Strategy****, if the following condition holds:*

$$\frac{\beta_2 - \alpha_2}{\beta_2 - \lambda_2} \leq \delta < 1$$

Proof. The grim trigger strategy could enforce cooperation if the temptation to defect (i.e., playing D instead of playing R) in a given time slot is smaller than the value of rewards minus the value of punishment in the following stages of the game. The temptation to defect for dishonest nodes is $\beta_2 - \alpha_2$. The total reward can be expressed as follows:

$$\alpha_2 + \alpha_2\delta + \alpha_2\delta^2 + \alpha_2\delta^3 + ... = \frac{\alpha_2}{1-\delta}$$

The value of payoff after the punishment strategy will be:

$$\lambda_2 + \lambda_2\delta + \lambda_2\delta^2 + \lambda_2\delta^3 + ... = \frac{\lambda_2}{1-\delta}$$

Therefore, considering that the temptation to defect should be smaller than reward minus the punishment infinitely, we obtain the following relation:

$$\delta \geq \frac{\beta_2 - \alpha_2}{\beta_2 - \lambda_2}$$

Moreover, δ must hold the following relation: $0 < \delta < 1$. Here, $\delta < 1$ ensures that $\lambda_2 < \alpha_2$. Otherwise, the punishment strategy would not work and none would have any hesitation to deviate. On the other hand, $\delta > 0$ states the fact that a node may have the temptation to lie only if $\beta_2 > \alpha_2$. □

Theorem 3 shows that honest nodes can prevent cheating by applying the grim trigger strategy. Note that the existence of dishonest nodes can be easily detected by applying the detection protocol shown in Fig. 5. After the detection of dishonest nodes, the grim trigger strategy taken by honest nodes guarantees the cooperation among nodes without applying the algorithm (Fig. 6) to find the individual cheater. In other words, non-cooperative (dishonest) nodes will not have enough incentive to cheat, knowing that the honest nodes can retaliate by playing the grim trigger strategy (i.e., not cooperating anymore). However, this will be guaranteed if and only if the smart grid operator chooses the amount of punishments (i.e., λ_i) and rewards (i.e., α_i), such that the condition of Theorem 3 is always satisfied given the period of subscription (i.e., δ).

5 Evaluation

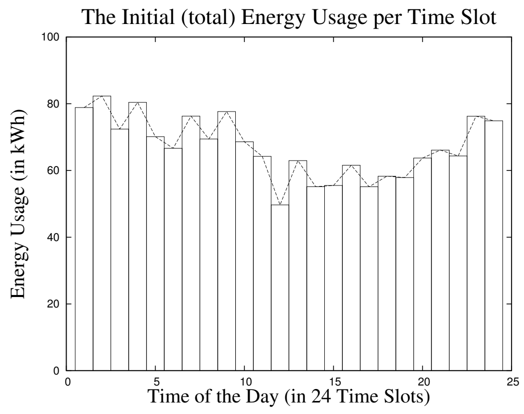
In this section, we present the evaluation results that we find by simulating our proposed solution for optimal and secured energy consumption scheduling. We also present the analytical results of applying the grim strategy with respect to the payoffs for different cases.

5.1 Simulation Results on Energy Consumption Scheduling

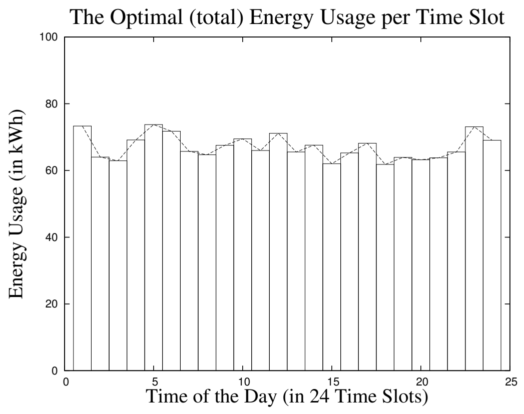
We evaluate our proposed solution, especially its scalability, by running a simulation program written in Java. We run our experiment in Intel Core2-Duo 2.2GHz Processor. Each user $x \in \mathbb{N}$ has an arbitrary number of nonshiftable household appliances and an arbitrary number of shiftable household appliances. These arbitrary numbers are taken from a range between 10 to 20.

Each nonshiftable appliance has a fixed operation schedule, while each shiftable appliance has a duration of operation and the possible time slots of operation (i.e., some consecutive time slots ranging from a starting slot to an ending slot). The number of possible time slots must be larger than the usage duration. The properties of the appliances are chosen arbitrarily. We experiment using an arbitrary number of users (i.e., nodes). For the constants of the cost function $C_h(.)$, we assume that both b_h and c_h are equal to 0 for all $h \in \mathbb{H}$, and the values of $a_h, h \in \mathbb{H}$ are an arbitrary value chosen between 0.5 to 0.6. The initial hourly cost of 15 users is depicted in Fig. 7(a). In the experiment of our proposed solution, we use fixed size mutual exclusive clusters, while the number of clusters depends on the number of participating users.

Our solution without clusters is almost similar to the algorithm of [9], except that the MPC algorithm (along with cryptographic measures) is executed by a node before each computation of the local optimization. We simulate MPC simply by the number of messages with a fixed processing overhead. The size of a cluster is taken as 5 users. We assume that the optimization process converges when the difference of cost reduction is less than 0.001 in the last consecutive rounds (e.g., 20) of local optimizations for each node.

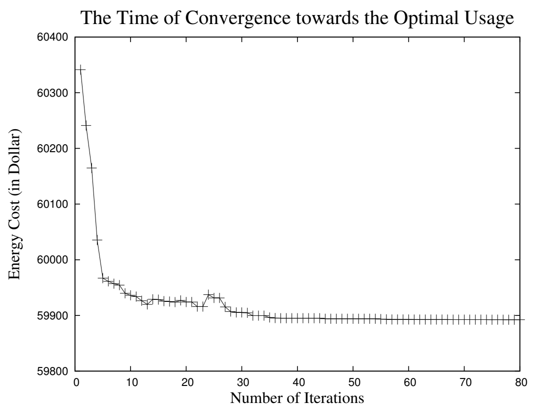


(a)

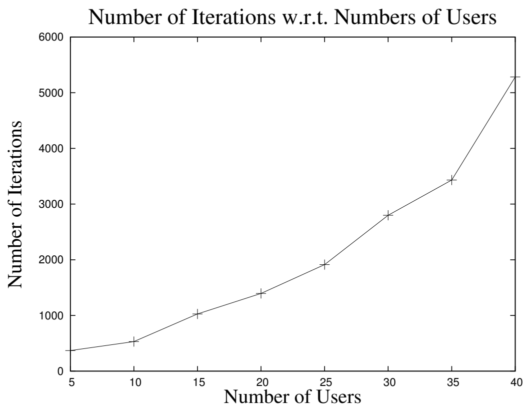


(b)

Fig. 7. (a) The initial total usage vector in different time slots (15 users), and (b) the corresponding optimal total usage vector (after executing optimization process).



(a)



(b)

Fig. 8. (a) The energy cost (reduction) trend towards the optimal value with the number of iterations of the optimization algorithm, and (b)the number of iterations that our proposed solution takes for convergence value.

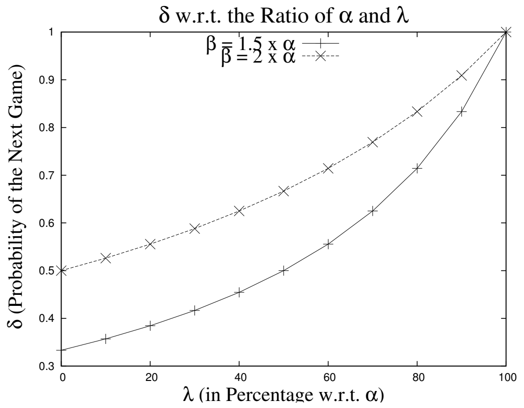
5.1.1 Convergence to Optimal Results

The summation of the usage vectors of all users prior to the optimal demand scheduling process is shown in Fig. 7(a). After the convergence, the summation of the optimal usage vectors of all users is shown in Fig. 7(b). We can see that the loads during different time slots in the case of the second figure are more balanced than those slots in the former figure. The peak hourly usage value is reduced from 84 (as in Fig. 7(a)) to 75 (as in Fig. 7(b)). Since the total load of slot increases the electricity price following a convex function, the users try to shift from the peak slots to the off-peak slots.

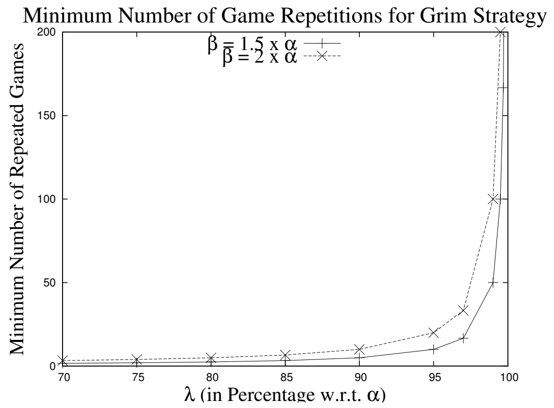
The cost reduction pattern with the execution of the algorithm (in case of 5 users) is shown in Fig. 8(a). This pattern shows that at the very beginning of the iterations the cost reduces very quickly. After the few initial iterations the algorithm starts converging slowly. We see from the figure that the overall energy cost is reduced from $\$6.04 \times 10^4$ to about $\$5.99 \times 10^4$. As we discussed before, due to the reduction of the usage in peak-hours, not only will the users save money, but so too will the power company by reducing the investment in the capacity of their power plants. Fig. 8(b) shows the number of iterations that our solution takes to converge in case of different numbers of users.

5.1.2 Comparison with the Existing Solution

In order to compare our proposed algorithm with the existing algorithm [9], we perform a number of experiments by simulating 10, 15, 20, and 25 users. In the case of our method, the experiments are done taking 2, 3, 4, and 5 clusters, where each cluster has 5 users. The running time of the simulation program is measured by the built-in timer of Netbeans, which gives the running time in seconds. The average running time of our method and that of the existing method [9] are shown in Table 2. From the table we can clearly observe that the running time of the existing method increases significantly with the number of users. We observe that if the number of users is equal or larger than 30, this algorithm takes a significantly high running time. This is why we do not run the simulation for 30 or more users for the existing method. The optimal result produced by our method is very close to that produced by the existing method, but according to the computational complexity, our



(a)



(b)

Fig. 9. (a) The value of δ (i.e., the probability of playing the next game) with respect to the values of α and λ, and (b) the minimum number of times (i.e., days) the game needs to be repeated after the deviation (in the case of grim strategy), so that the grim strategy works correctly.

TABLE 2
Comparison between proposed and existing methods (cost in dollars and time in seconds)

		Existing Method		Proposed Method	
Users	Initial Cost	Time	Cost	Time	Cost
10	281541.3	0.099	276733.4	0.240	276730.3
20	1094853.1	0.655	1074111.0	0.420	1074105.1
25	1456517.5	27.038	1440312.5	0.526	1440307.2
30	2087363.4	-	-	0.698	2053263.8
40	3772814.5	-	-	2.753	3615380.5
50	6999563.1	-	-	10.044	6785863.3
100	16538176.50	-	-	46.68	15868284.50

TABLE 3
Execution time in relaxed convergence condition

Users	Iterations	Time (Seconds)
25	1713	0.21
50	5207	0.56
100	20831	2.02
500	471476	21.87
1000	2122834	102.56

method performs exceptionally well. In the case of 25 users, our solution is around 50 times faster than that of [9].

5.1.3 *Relaxed Convergence Condition and Execution Time*

We have defined convergence in Algorithm 1 as the condition when there is no significant update, let Δ, in the consumption schedule for a good number (e.g., equal to the number of users) of last consecutive rounds. We have used $\Delta = 0$, i.e., no update in the consumption schedule, in our experiments. However, we have already seen in Fig. 8(a) that the algorithm converges fast in the beginning, while the change becomes insignificant after some iterations (e.g., 30 iterations for 5 users). Let us relax the convergence condition by taking Δ as 0.001% of the initial energy cost, i.e., the algorithm is assumed to reach the convergence when there is no change larger than 0.001% of the initial energy cost for many consecutive rounds. Table 3 shows convergence times by varying the number of users. We see that the execution time is reduced a lot comparing to the case when $\Delta = 0$ (as seen in Table 2). Our algorithm takes around 100 seconds for 1000 users.

5.2 Analytical Results on Grim Strategy

Fig. 9(a) shows how the probability of playing the next game (i.e., δ) depends on α and λ, which are the payoffs in the cases of cooperation and deviation. We consider two different βs, which represent the payoff while the dishonest group lies and while the honest group continues to cooperate cooperates, respectively. We take β as a ratio with respect to α. In one scenario, β is 1.5α, while in the second scenario β is 2α. We also consider λ as a percentage of α. We vary λ from 0% of α to 100% of α. In Fig. 9(a), we see that δ increases exponentially with the decrease of the difference between α and λ (i.e., increase of the ratio between them). When there is no difference, i.e., λ is equal to α, the game must be (i.e., δ is 1) played infinitely. Remember that Theorem 3 shows that if $\delta \geq 1$, i.e., $\alpha \leq \lambda$, the grim strategy is not required and deviation will always be preferred to cooperate.

We also calculate the minimum number of times or repetitions, n that the game needs to be played to neutralize the benefit of dishonesty by applying the grim or punishment strategy. We calculate n by solving the equation $\beta - \alpha = n * (\alpha - \lambda)$. Here, $\beta - \alpha$ is the benefit of deviating, while $\alpha - \lambda$ is the loss due to the punishment strategy. Fig. 9(b) shows the value of n in different cases by varying λ. It shows that the closer λ is to α, the higher is n. This behavior states that the participation of a player should be ensured to be sufficiently long participation, so that after any deviation a node has to play for a number of times until the extra benefit received from deviation become zero due to the punishment strategy.

6 Discussion

In this section, we discuss about some points regarding our proposed solution for secure and private energy consumption scheduling.

6.1 A utility collector based energy consumption scheduling

Smart meters often forms a mesh network and they are connected to a data collector through this mesh network. A common framework for the communications often includes home area networks, building area networks, and neighborhood area networks, where we need many different communications between meters before sending data to the grid [23]. If we consider the usage data aggregation only, a smart meter needs to send its data to the collector through other meters, which does not preserve the privacy. Therefore, an aggregation scheme taking the help of a collector still needs to apply some privacy-preserving protocols. In addition, it is good to have privacy as well as resiliency against single node failure. Therefore, we choose distributed algorithm for optimal scheduling of energy consumptions. The similar motivation has been presented in previous works, such as [9]. Moreover, the collector is responsible for sending individual billing data and command controls to and from the meters and utility or control center. Thus, it will be overwhelming for a collector to process the optimal scheduling of energy usage. In our proposed mechanism, we employ a trusted third party only when there is a suspicion of cheating.

6.2 The existing security in the smart metering communication

A utility collector often sends collected usage data to the utility center by creating a secure channel with the utility server. However, in practice such secure channel is yet to be implemented in many places properly or fully, particularly for the communication among smart meters and collectors. For example, LonTalk is often used for communication among meters and collectors, which mainly ensures authentication of the two communicating parties [24]. After all, the privacy is not considered as a concern in this communication which is mainly taken place from a meter to a collector for reporting usage data to the utility center. On the other hand, the optimal scheduling protocol is executed among nodes (smart meters), where each node needs to know and use the received data for the optimal scheduling of energy consumptions. Therefore, the security, integrity, and privacy issues are required to be addressed together and comprehensively taking the protocol execution into consideration. In our work, we identify the potential threats of eavesdropping and false data injection with respect to the optimal energy usage scheduling protocol and thus provide a defense mechanism by ensuring data privacy along with the data authentication and integrity. We also provide security against semi-honest participating nodes, especially when they provide false information about their potential energy usage.

6.3 Scalability of the proposed energy consumption scheduling algorithm

From our simulation results, we find that the number of iterations to reach the convergence point grows rapidly in the number of users. However, we also find that the optimization process starts to converge after few iterations (e.g., only 5 to 10 iterations for 5 users), and further convergence becomes negligible after some iterations. Thus, the participating nodes can relax the convergence condition to stop the optimal scheduling process early within an acceptable time frame (as shown in Table 3). In addition, our cluster based solution keeps the size of the participating nodes in the MPC process small. Thus, the application of MPC has less impact on the protocol convergence time with the increase of the number of users. Moreover, parallel executions are also possible in such a cluster based design, which can significantly reduce the convergence time.

7 Related Work

In this section, first we briefly discuss the research done so far for optimizing the energy usage cost. Then, we present a brief discussion on the research work that address the security and privacy problems related to this area.

M. Fahrioglu et al. proposed a game [3] that models the interaction between a utility and its customers to let the customers help a utility solve a variety of problems. K. Herter in [4] proposed a mechanism to minimize the generation cost with the indirect interaction between the energy users and the energy provider by providing incentive for using energy at off-peak hours with the help of time-varying energy prices.

A. Gomes et al. discussed a load control strategy in [25]. The paper implemented the elastic electrical price. The authors pointed out that the complexity is increased by the diversity and volatility in power systems, because the individuals have different aim.

N. Ruiz et al. in [7] introduced a direct load control algorithm based on linear programming to operate the virtual power plant composed of a large number of users with load reduction capabilities. The author in [26] made an investigation on relationship between the critical-peak pricing and the households with different income and usage in California State. This work justifies the need of a good cost function which can affect the behavior of users and their usage. However, the automation of the optimal demand side management solely by the interactions among the participating users (i.e., without an involvement of the utility) can offer an open, independent, and creative solution.

Mohsenian-Rad and Leon-Garcia proposed an optimal and automatic residential energy consumption scheduling framework in [27] that minimizes the electricity payment as well as the operational waiting time of each household appliance under a real-time pricing model. The authors addressed a similar problem in [28], [9] for autonomous demand side management within a neighborhood. They considered the deployment of energy consumption scheduling devices in smart meters, which can communicate with each other. The game theory is applied to distributively find the optimal consumption scheduling for all the users. There are also other autonomous demand side management solutions presented by different researchers in [11], [12], [13], [14], [15]. In this solutions, the overall energy production or usage cost is minimized by controlling the energy consumption directly with optimal scheduling or indirectly with suitably chosen energy prices. Another demand response system is designed in [29] by minimizing the peak-to-average ratio of the aggregate load demand.

In [19], fairness in autonomous demand response is discussed based on the contribution that a user makes in achieving the system's global objective. However, none of these algorithms addresses the security problems introduced by their proposed algorithms.

Li et al. presented a distributed data aggregation process for smart meters involved in transmitting data from a set of meters to the data collector [30]. To protect user privacy, they applied homomorphic cryptography. So, the meters participating in the aggregation cannot see intermediate results. However, their solution is costly and it cannot solve the security issues other than privacy found in our problem domain. In [31], the authors showed that it is possible to extract the detail information about the household activities of a customer without any prior knowledge. In order to secure the customer's privacy, the authors also proposed a homomorphic encryption based zero-knowledge billing protocol for smart meters. Raj et al. proposed a privacy preserving approach by perturbing the usage data while meeting the utility needs [32]. Sankar et al. in [33] introduced the competitive privacy problem in distributed state estimation at the regional transmission organizations.

Unlike above works, we proposed a lightweight MPC based data aggregation protocol that solves privacy as well as data integrity problems. In addition, with respect to the data integrity, we addressed the need of the participants' truthfulness in such protocols and proposed mechanisms to ensure this truthfulness by punishing for lying. Although we focused on the optimal scheduling of customers' energy consumption, our work is applicable to other problems in smart grids that need data aggregation among the participants.

8 Conclusion

This paper has presented a mutually exclusive cluster based solution for the optimal energy management problem, which can solve the security and privacy threats found in the earlier proposed solutions. Our solution provides data privacy, as well as protection from false data injection, i.e. data integrity. Comparing to the existing demand-side management solutions, the proposed approach is also highly efficient as in our proposed solution the running time increases almost linearly with the increase of the number of users. We have also presented an example which shows that a user participating in the optimization process can benefit by lying about his usage if the smart grid uses a per-slot based charging mechanism. We have proved formally that a dishonest node can make benefit by lying about usage with per-slot based charging mechanism. We have proved that if a user cannot be untruthful about his total usage, then he cannot get any incentive by lying about the distribution of usage in different time slots. We have proposed a verifier based solution in order to detect malicious node who declared false information about its usage. We have also analyzed a grim trigger strategy solution for ensuring the truthfulness, that can help smart grids to build incentive-based charging mechanisms. These mechanisms can be designed in future research works by choosing appropriate subscription period, reward and punishments values.

Acknowledgments

The authors are very grateful to the anonymous reviewers for constructive feedback and insightful suggestions that helped to significantly improve the quality of this work. We would like to thank Ms. Monireh Mohebbi Moghadam for her suggestions and feedbacks.

References

[1] U.S. Department of Energy. 2010 renewable energy data book, March 2011. http://www.afdc.energy.gov/pdfs/51680.pdf.

[2] C.W. Gellings and J.H. Chamberlin. *Demand-side management: concepts and methods, 2nd edition*. Fairmont Press, 1993.

[3] M. Fahrioglu and F. L. Alvarado. Designing incentive compatible contracts for effective demand management. *IEEE Transactions on Power Systems*, 15(4):1255–1260, November 2000.

[4] K. Herter. Residential implementation of critical-peak pricing of electricity. *Energy Policy*, 35(4):2121–2130, April 2007.

[5] R. Krishnan. Meters of tomorrow [in my view]. *Power and Energy Magazine, IEEE*, 6(2):96–94, March 2008.

[6] B. Ramanathan and V. Vittal. A framework for evaluation of advanced direct load control with minimum disruption. *IEEE Transactions on Power Systems*, 23(4):1681–1688, November 2008.

[7] N. Ruiz, I. Cobelo, and J. Oyarzabal. A direct load control model for virtual power plant management. *IEEE Transactions on Power Systems*, 24(2):959–966, May 2009.

[8] C. Triki and A. Violi. Dynamic pricing of electricity in retail markets. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies (4OR)*, 7(1):21–36, March 2009.

[9] A.-H. Mohsenian-Rad, V.W.S. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia. Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. *IEEE Transactions on Smart Grid*, 1(3):320–331, December 2010.

[10] R. Gibbons. *Game Theory for Applied Economics*. Princeton University Press, 1992.

[11] C. Ibars, M. Navarro, and L. Giupponi. Distributed demand management in smart grid with a congestion game. In *First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 495–500, October 2010.

[12] N. Gatsis and G.B. Giannakis. Cooperative multi-residence demand response scheduling. In *45th Annual Conference on Information Sciences and Systems*, pages 1–6, March 2011.

[13] S. K. Vuppala, K. Padmanabh, S. K. Bose, and S. Paul. Incorporating fairness within demand response programs in smart grid. In *IEEE PES, Innovative Smart Grid Technologies (ISGT)*, pages 1–9, January 2011.

[14] P. Samadi, R. Schober, and V.W.S. Wong. Optimal energy consumption scheduling using mechanism design for the future smart grid. In *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 369–374, October 2011.

[15] M. Burcea, W.-K. Hon, H.-H. Liu, P. W.H. Wong, and D. K.Y. Yau. Scheduling for electricity cost in smart grid. In *Combinatorial Optimization and Applications*, volume 8287, pages 306–317. Springer International Publishing, 2013.

[16] M. A. Rahman, L. Bai, M. Shehab, and E. Al-Shaer. Secure distributed solution for optimal energy consumption scheduling in smart grid. In *IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 279–286, June 2012.

[17] A. J. Wood and B. F. Wollenberg. *Power Generation, Operation, and Control, 2nd Edition*. Wiley, 1996.

[18] Y. Miyano and T. Namerikawa. Load leveling control by real-time dynamical pricing based on steepest descent method. In *SICE Annual Conference (SICE), 2012 Proceedings of*, pages 131–136, August 2012.

[19] Z. Baharlouei, M. Hashemi, H. Narimani, and H. Mohsenian-Rad. Achieving optimality and fairness in autonomous demand response: Benchmarks and billing mechanisms. *IEEE Transactions on Smart Grid*, 4(2):968–975, March 2013.

[20] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2009.

[21] M. J. Atallah and W. Du. Secure multi-party computational geometry. In *7th Springer-Verlag International Workshop on Algorithms and Data Structures*, pages 165–179, London, UK, 2001.

[22] A. C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, November 1982.
[23] M. M. Fouda, Z. M. Fadlullah, N. Kato, R. Lu, and X. Shen. A lightweight message authentication scheme for smart grid communications. *IEEE Transactions on Smart Grid*, 2(4):675–685, December 2011.
[24] Lontalk protocol specification. http://www.enerlon.com/JobAids/Lontalk%20Protocol%20Spec.pdf.
[25] A. Gomes, C.H. Antunes, and A.G. Martins. A multiple objective evolutionary approach for the design and selection of load control strategies. *IEEE Transactions on Power Systems*, 19(2):1173–1180, May 2004.
[26] P. Centolella. The integration of price responsive demand into regional transmission organization (RTO) wholesale power markets and system operations. *Energy*, 35(4):1568 – 1574, April 2010.
[27] A.-H. Mohsenian-Rad and A. Leon-Garcia. Optimal residential load control with price prediction in real-time electricity pricing environments. *IEEE Transactions on Smart Grid*, 1(2):120–133, September 2010.
[28] A.-H. Mohsenian-Rad, V. W. S. Wong, J. Jatskevich, and R. Schober. Optimal and autonomous incentive-based energy consumption scheduling algorithm for smart grid. In *Innovative Smart Grid Technologies (ISGT)*, pages 1–6, January 2010.
[29] H. K. Nguyen, J.B. Song, and Z. Han. Demand side management to reduce peak-to-average ratio using game theory in smart grid. In *IEEE Conference on Computer Communications (INFOCOM) Workshops*, pages 91–96, March 2012.
[30] F. Li, B. Luo, and P. Liu. Secure information aggregation for smart grids using homomorphic encryption. In *First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 327–332, October 2010.
[31] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In *2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, pages 61– 66, 2010.
[32] S. R. Rajagopalan, L. Sankar, S. Mohajer, and H. V. Poor. Smart meter privacy: A utility-privacy framework. In *2nd Annual IEEE Conference on Smart Grid Communications (SmartGridComm)*, pages 17–20, October 2011.
[33] L. Sankar, S. Kar, R. Tandon, and H. V. Poor. Competitive privacy: An information-theoretic approach. In *IEEE Conference on Smart Grid Communications (SmartGridComm)*, pages 17–20, October 2011.

Mohammad Ashiqur Rahman has received his PhD in Computing and Information Systems from the University of North Carolina at Charlotte (UNC Charlotte), USA, in 2015. Earlier, he received his BSc and MSc degrees in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET), Dhaka, in 2004 and 2007, respectively. He is currently a member of the Cyber Defense & Network Assurability (CyberDNA) Research Center at UNC Charlotte. He will be joining the Department of Computer Science at Tennessee Tech University as an Assistant Professor, starting from August 2015. His primary research interests include cyber infrastructure security analytics and automation, risk analysis and security hardening, and policy verification and optimal management. His research area covers cyber security and management for both general networks as well as cyber physical systems.

Mohammad Hossein Manshaei received the BSc degree in Electrical Engineering and the MSc degree in Communication Engineering from the Isfahan University of Technology in 1997 and 2000. He earned another MSc degree in Computer Science and his Ph.D. in Computer Science and Distributed Systems from the University of Nice Sophia-Antipolis, France, in 2002 and 2005. He did his thesis work at INRIA, Sophia Antipolis, France. He is an Assistant Professor at the Isfahan University of Technology, Iran. From 2006 to 2011, he was a senior researcher and lecturer at EPFL, Switzerland. He was a visiting researcher at the UNCC, during summer 2013. His research interests include wireless networking, wireless security and privacy, social networks, cognitive radios, and game theory.

Ehab Al-Shaer is a Professor and the Director of the Cyber Defense and Network Assurability (CyberDNA) Center in the College of Computing and Informatics at University of North Carolina Charlotte. He received his MSc and Ph.D. in Computer Science from the Northeastern University (Boston, MA) and Old Dominion University (Norfolk, VA) in 1998 and 1994 respectively. His primary research areas are network security, security management, fault diagnosis, and network assurability. Prof. Al-Shaer edited/co-edited more than 10 books and book chapters, and published about 150 refereed journals and conferences papers in his area. Prof. Al-Shaer is the General Chair of ACM CCS 2009-2010 and NSF Workshop in Assurable and Usable Security Configuration, August 2008. Prof. Al-Shaer also served as a Workshop Chair and Program Co-chair for a number of well-established conferences/workshops in his area including IM 2007, POLICY 2008, ANM-INFOCOM 2008, ACM CCS 2010. He also served as a member in the technical programs and organization committees for many IEEE and ACM conferences.

Mohamed Shehab is an Associate Professor of Software and Information Systems Department at the University of North Carolina at Charlotte. He received his PhD degree in Electrical and Computer Engineering from Purdue University in August 2007. His research and teaching interests are in the broad areas of network and information security. In particular, his research focuses on advancing the state of the art in the design and implementation of distributed access control protocols to cope with the requirements of emerging distributed, web services, social networks, database systems and peer-to-peer environments.