2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications

Secure Distributed Solution for Optimal Energy Consumption Scheduling in Smart Grid

Mohammad Ashiqur Rahman, Libin Bai, Mohamed Shehab, and Ehab Al-Shaer
University of North Carolina at Charlotte
Emails: {mrahman4, lbai2, mshehab, ealshaer}@uncc.edu

***Abstract*—The demand-side energy management is crucial to optimize the energy usage with its production cost, so that the price paid by the users is minimized, while it also satisfies the demand. The recent proposed solutions leverage the two-way communication infrastructure provided by modern smart-meters. The demand management problem assumes that users can shift their energy usage from peak hours to off-peak hours with the goal of balancing the energy usage. The scheduling of the energy consumption is often formulated as a game-theoretic problem, where the players are the users and their strategies are the load schedules of their household appliances. The Nash equilibrium of the formulated game provides the global optimal performance (i.e., the minimum energy costs). To provide a distributed solution the users require to share their usage information with the other users to converge to the Nash equilibrium. Hence, this open sharing among users introduces potential privacy and security issues. In addition, the existing solutions assume that all the users are rational and truthful. In this paper, we first highlight the privacy and security issues involved in the distributed demand management protocols. Secondly, we propose an efficient clustering based multi-party computation (MPC) distributed protocol that enables users to share their usage schedules and at the same time preserve their privacy and confidentiality. To identify untruthful users, we propose a mechanism based on a third party verifier. Through simulation experiments we have demonstrated the scalability and efficiency of our proposed solution.**

Keywords: Smart Grid, Energy Consumption Schedule, Privacy.

I. 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 [2] 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 (between 6pm to 7pm in U.S.) 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 improving the power plant capacity. This will also result in the decrease of the price of electricity. Controlling the energy usage at the customer side of smart meters has received a lot of attention. Some research (e.g., [3], [8]) 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, especially considering varying energy prices, giving incentive for using energy at off-peak hours.

Smart grids provide innovative and efficient energy management services that offer operational reliability and value-added advantages to both users and energy providers. The potential market for smart grids show that it will be the most widely deployed critical infrastructure in the 21st century. The popularity of smart grids is followed mainly from the advent of smart meters. Smart meters give the opportunity of two-way communication between the meters and the utility servers through the intelligent collectors [4]. This opportunity gives the researchers an opportunity to rethink for the optimal demand side management. In [7] and [5], researchers propose electricity scheduling methods 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 works focus on the household users, particularly the household appliances, which are flexible in their usage time, and hence one can shift the usage time from peak time to off-peak time to reduce the energy cost.

Mohsenian-Rad et al. in [1] proposed an autonomous and distributed demand-side energy management system among users that takes advantage of the communication infrastructure among the smart meters. A game-theoretic model is applied to formulate the energy consumption scheduling problem. The Nash equilibrium of the game gives the maximum payoffs to the users. As a by-product of this algorithm, the peak-to-average ratio of the energy demand is reduced. However, there are several security problems in this optimization algorithm. In this paper, we propose a solution to solve these problems.

A. Motivation

The distributed algorithm for the optimization of the energy consumption schedule requires a user to broadcast its hourly usage information to the other participating users. This interferes with the *privacy* problem. Each user's energy consumption behavior is revealed to participants, which might be considered private. 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

978-0-7695-4745-9/12 $26.00 © 2012 IEEE
DOI 10.1109/TrustCom.2012.252

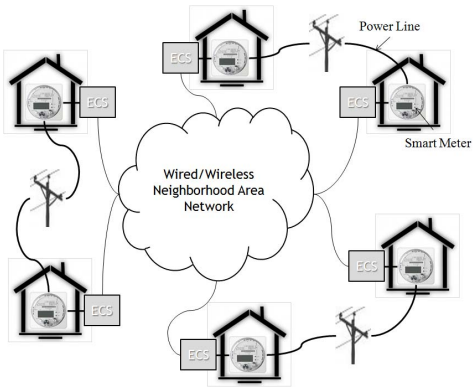

Fig. 1. The communication network between the smart meters.

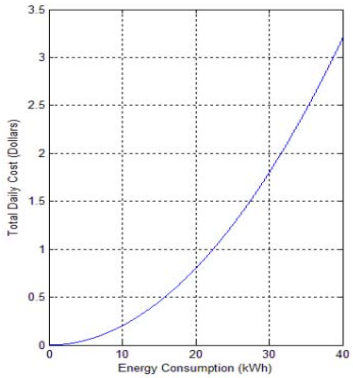

Fig. 2. A quadratic convex and increasing cost function.

these attacks, the optimization algorithm can come up with a result, which is different from the actual optimal result. As a result, the participating users after optimization may not get the expected benefit, rather they can end up paying much more than the regular price.

There are some more problems with the optimization algorithm. Firstly, 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 do this if he can find better (individual) incentive than being truthful. Second, 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.

B. Contribution

We propose a mechanism for optimizing energy cost that meets the security challenges and performs efficiently. The solution is developed on top of the typical energy consumption scheduling model, which offers the following contributions:

- 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 [1] as a use case.
- We enhance the efficiency of the distributed demand management protocol by clustering the participants and executing the optimization protocol over the clusters.
- We show with a scenario that a participating user can benefit by telling lies about its usage. We propose an assumption at which there is no incentive from defecting. We devise an adjudicator (i.e., a truthful third party verifier) based solution in order to ensure the truthfulness of the participating users.
- We demonstrate the scalability and efficiency of our proposed 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 II. We present our proposed solution in Section III. In the following section, we present the simulation experiments and evaluation results. In Section V, we describe the related work. We conclude the paper in Section VI.

II. Energy Cost Minimization Model

We begin the section by describing the smart power system and the electric consumption price model. Then we briefly introduce the energy consumption scheduling optimization and the corresponding distributed algorithm accordingly.

A. Energy Consumption Model

The smart power system that we assumed in this paper is shown in Fig. 1. 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 though the power line communication media (or Wi-Fi) and it forms a Local Area Network (LAN). The meters communicate with each other by appropriate communication protocol. The energy provider is also connected to the LAN. Each smart meter has an *energy consumption scheduling* (ECS) unit, which is capable of doing some arithmetic computation/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 unit cost can be different at different time which depends on the total electrical usage of all the users. Generally, the unit electrical cost at night is lower than the electrical cost during the day time. The cost equation is a strictly increasing function as follows:

$$C_h(b) < C_h(c), \forall b < c \tag{1}$$

The cost function is strictly convex.

$$\begin{gathered} C_h(\theta b + (1-\theta)c) < \theta C_h(b) + (1-\theta)C_h(c), \\ \forall b < c, 0 < \theta < 1 \end{gathered} \tag{2}$$

A simple cost function is shown in Fig. 2. This is a quadratic cost function as defined in the following equation:

$$C_h^{L_h} = a_h L_h^2 + b_h L_h + c_h \tag{3}$$

In Equation 3, $a_h > 0$, $b_h > 0$, and $c_h > 0$ are the constants.

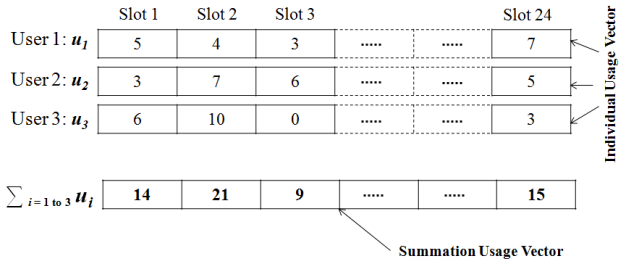

Fig. 3. An example of individual usage vectors of three users and corresponding summation usage vector.

The set of users are denoted as $\mathbb{N}$. The number of users is defined as N (i.e., $N = |\mathbb{N}|$). The vector $u_x = \left[u_x^1, u_x^2, ..., u_x^{24}\right]$ 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 denote the hourly consumption of x at the time h. Hence, the summation of the usage vectors of all the users is denoted as follows:

$$u = \left[u^1, u^2, ..., u^{24}\right] = \left[\sum_{x\in\mathbb{N}} u_x^1, \sum_{x\in\mathbb{N}} u_x^2, ..., \sum_{x\in\mathbb{N}} u_x^{24}\right] \quad (4)$$

The summation is also a vector, which is named as the *summation usage vector*). Fig. 3 shows an example of individual usage vectors and corresponding summation usage vector. The total cost of all the users is defined as follows:

$$S = \sum_{h\in\mathbb{H}} C_h(u^h) \quad (5)$$

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:

$$u_{x,a} = \left[u_{x,a}^1, u_{x,a}^2, ..., u_{x,a}^{24}\right] \quad (6)$$

The usage vector of a user $x \in \mathbb{N}$ is the summation of the usage vectors of all appliances:

$$u_x = \left[u_x^1, u_x^2, ..., u_x^{24}\right] = \left[\sum_{a\in\mathbb{A}} u_{x,a}^1, ..., \sum_{a\in\mathbb{A}} u_{x,a}^{24}\right] \quad (7)$$

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

$$PAR = \frac{max_{h\in\mathbb{H}} u^h}{\sum_{h\in\mathbb{H}} u^h/|\mathbb{H}|} \quad (8)$$

B. Energy Consumption Scheduling Game

The *energy consumption scheduling game* is defined below:

- Players: The registered users in set $\mathbb{N}$.
- Strategies: Each user $x \in \mathbb{N}$ finds its energy usage vector u_x to maximize its payoff considering the usage of other users (u_{-x}).
- Payoff: The payoff function is defined as:

$$P_x(u_x; u_{-x}) = -\frac{u_x}{\sum_{x'\in\mathbb{N}} u_{x'}} \sum_{h\in\mathbb{H}} C_h(u^h) \quad (9)$$

The overall usage vector changes with a change in u_x. It has been proved that once the game has no change, the game reaches a fixed point, which is the Nash equilibrium [1]. At this point, no user will get benefit by choosing other scheduling.

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 unit. Each node x executes the local optimization randomly. Each execution process consists of the following three steps:

1) The node x solves local optimization by solving Equation 9 by applying some known optimization algorithm.
2) If x finds that the optimization algorithm results a consumption schedule different than its earlier consumption schedule u_x, it updates u_x and broadcasts a control message to announce the updated u_x to other nodes.
3) When a node $\bar{x}$ ($\bar{x} \neq x$) receives the control message, it updates the summation usage vector accordingly.

A node continues to execute the process at different random times until there is no new control message for a while. At this point, the algorithm converges to the optimal point.

We have found several problems in this algorithm. Firstly, the distributed algorithm requires the broadcasting of the usage vector by each node. This broadcasting message is also sent as a plain text. This introduces the privacy problem along with the following security problems: (i) the usage vector of a user can be eavesdropped by listening to the communication media, and (ii) an adversary can inject false data in order to fail the optimization. Secondly, 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 is the large number of messages that are required to exchange in the network back and forth till the convergence (as one node's optimizing decision, which is local to it, affects the decision of the other nodes). 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 to be implemented in a large scale. Another problem of this method is that the truthfulness of the participating nodes in the optimization are not ensured. A participating node can cheat about its usage vector to get advantage. An example of this is presented in Section III-C.

III. Proposed Solution

In this section, we describe our solution in two steps. First we describe how we address the security issues, i.e., the privacy preservation and the protection against false data injection. Next we present the mechanism we apply to increase the scalability of the solution. We end the section by discussing the truthfulness issue of the nodes and by proposing a mechanism for ensuring their truthfulness.

A. Security Incorporation

We propose a secure multi-party computing (MPC) algorithm to resolve the privacy problem of sharing user data. In

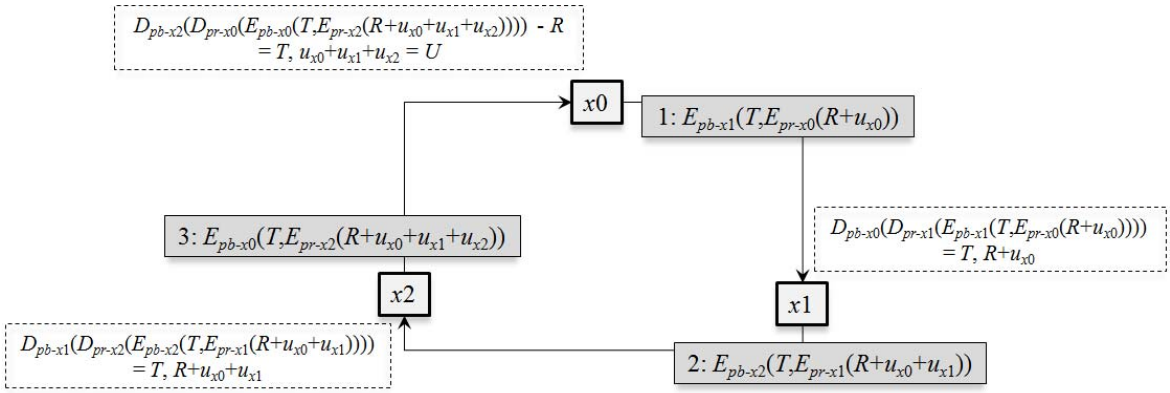

Fig. 4. The MPC technique for summing the usage vectors of the participating nodes that includes authentication and confidentiality security measures also.

our algorithm, a computing node runs the MPC algorithm to know the summation of the usage vectors of all other nodes before doing each local optimization. Although any suitable MPC solution [12] 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. 4 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 execution of local optimization. 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 it encrypts the resultant usage vector firstly using its private key (i.e., pr_{x1}) and secondly 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 add). 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 node (i.e., $x0$) along with the list of participating nodes applying the same encryption steps on the message. Note that at least three nodes are required to participate in the MPC technique. In 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.

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 9 of u_x.
  end if
until There is no significant update in u_x for a number of last consecutive rounds.

Importance of applying both encryptions. In this 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 first encryption is not done, any adversary can inject false data. If false data is injected, the summation will not be the actual one, and thus the optimal value will be incorrect. As a result, the nodes will not get benefit from participating in the optimizing process. This encryption is known as *signing* the message. In the case of second encryption, if the message was not encrypted by the receiving node's public key, anyone can eavesdrop the message, which breaks the confidentiality. Though the message content (i.e., the summation usage vector and the participating nodes) does not help an adversary to launch an attack on the protocol,

the MPC initiating node $x0$ can eavesdrop the messages, and decrypt them to know the usage vectors of the other nodes. In Fig 4, for example, $x0$ eavesdrops the message sent by $x1$ to $x2$ and decrypts the message using the public key of $x1$ (as it was not encrypted using the $x2$'s public key). Hence, it can easily figure out the usage vector of $x1$. Similarly, from the message sent by $x2$ to $x0$, it get 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 all messages between the nodes. The communication is also susceptible to replay attacks. The random usage vector (R) also helps to get rid of such attacks.

Protection against sandwich attack. As a node selects the next node after it in the logical sequence randomly at each MPC execution, in 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 is done because, otherwise, a node can be sandwiched by the two nodes before and after the node in the sequence in order to learn its usage vector.

Importance of executing MPC 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 to broadcast the updated summation usage vector to all other participating nodes. At the time of executing each local optimization, a node requires to use the latest summation usage vector. But it 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 vector $\triangle U = U_A - U_B = [\triangle X_1, \triangle X_2, \ldots, \triangle X_N]$ gives some hints about the usage vector of the computing node.

B. Efficient Computing

Usually a large number of rounds of local optimizations are required to reach the global optimal point. While reaching the global optimal is very time consuming, 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. 5. A number of clusters of nodes are formed among the participating nodes, which are mutually exclusive, that is, the intersection of any two clusters is an empty set. A node executes MPC in its cluster only. Hence, the node requires to get 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

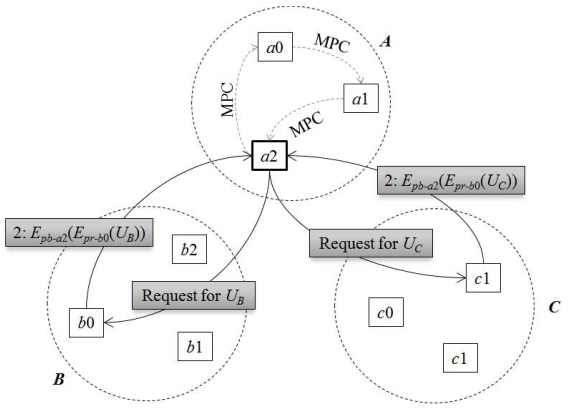

Fig. 5. The cluster based MPC technique

be the most updated one, as the selected node might not be the last that has executed the optimization (so it has done with the MPC to get the summation of the usage vectors of its cluster members). Hence, the summation found in this way might not be accurate, and as a result the optimization cannot be perfect. However, the possible error in the optimized value removes as the algorithm converges. Algorithm 1 is modified for the cluster based solution in Algorithm 2. The 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.

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 (9) 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.

C. Ensuring the Truthfulness of Participating Nodes

Here, we assume a semi-honest user model [11], i.e., a user may cheat passively (tell a lie about its information), but it always follow the protocol (the MPC and optimization procedure). We show an example of how a participating user can get benefit from lying in Fig. 6. We consider three users A, B, and C, along with their arbitrary usage vectors (three time slots in each vector) in the example. In Fig. 6(a), we show the expected scenario, where each participating user is truthful about its usage. In that case, we see that B pays \$304 for its electricity usage. We follow a simple price function, $C(L) = 2L^2$ for each time slot. In Fig. 6(b), we show the case when B lies about his usage. He pretends that he would use 10kWh in the third time-slot, though his actual intention

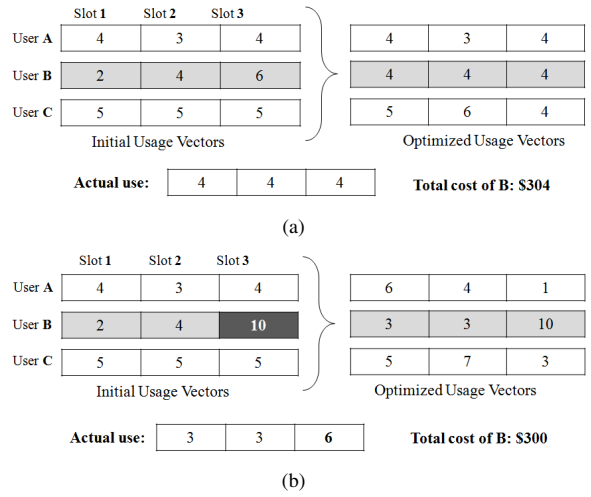

Fig. 6. Cost (price) of usage of a node (a) when it is truthful about its usage, (b) when it has lied about its usage (more than its actual requirement).

is different (6 kWh). After the optimization, B is supposed to use 10 kWh in the third slot. However, in practice he uses 6kWh. As a result, the price at that slot reduces well and B pays \$300 for his usage. Hence, he gets benefit from lying. However, other users A and C will need to pay more than that in the case of the truthful scenario. If B could not lie about his total usage, he cannot benefit from lying about its usage vector, i.e., the potential usage distribution in different time slots. The following theorem proves this.

Theorem 1: If the advertised total usage (at the time of participating in the optimization process) is equal to its actual usage, there is no incentive in lying about the usage vector.

Proof: Let $\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_N$ is the optimal usage vectors of N participating nodes with an untruthful user x. The user x has lied about its usage vector (usage distribution only) at the time of optimization process. After the finish of the process, 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. Let, based on the current optimal usage vectors, the total load of the slot s is L. Hence, the current load $\hat{L}$ is $L + l$, which eventually increases the electricity price rate of that slot from P to $\hat{P}$. If it would use the slot s for the load l at the time of optimization, the total load of the slot s L' would not be more than $L + l$. Hence, $L' \leq L + l$. The reason behind the possibility of $L' < L + l$ is that, at that time of optimization, other competing nodes might move from s to a different slot, since they found one more candidate x with load l for this slot. They might move to $\bar{s}$ as x would not be a candidate of load l for this slot. Therefore, no way x can get more payoff (less cost) by lying about its usage distribution only.

Therefore, it is important to ensure whether any participating node has lied about its total usage. We propose a solution to address this issue. Each participating node knows the summation of the usage vectors of all the participating nodes from the beginning to the end (when the process converges) of the optimization process. From the summation usage vector, each node knows the overall 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 assume that the power utility company provides the nodes with the usage report of each day that shows the total load at each slot of that day. If a node finds a significant difference between the reported load and the expected load, one can go for verification. A third party is required to play the *adjudicator* or *judge* role in this case. The following series of actions can find the cheater, if there is any:

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 of the usage vectors of all participating nodes. It computes the total usage T_I from U_I.
   b) Each node hashes its total usage. The hash function is one-way cryptographically secure and known to all. Along with this hashed value, it also sends the summation usage vector $\bar{U}_I$ to each node. It signs the message using its private key and sends it to the requesting node by encrypting it with the receiver's public key. This message acts as the *commitment* made by the node about its total usage.
   c) Receiving such a message, each node compares its U_I with the received $\bar{U}_I$, which ensures that all nodes start with the same summation.
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.
3) At the time of real use, if a node finds anomaly between the expected and the reported usage (after observing the report provided by the utility company), it initiates a verification. The verification is done by the adjudicator, which works as follows:
   a) The verification initiating node sends the energy provider the hashed value of the total usage of each node.
   b) The adjudicator collects the actual total energy usage of each participating node from the energy provider. Now it hashes the actual total usage and compares it with the corresponding hashed value provided by the initiator. If the actual is significantly different than the advertised one, the corresponding node lied about its total usage.

The actual usage can be reasonably a little different from the advertised one, but the difference cannot be significantly high. Hence, a threshold value can be defined and the verification for equality can be done within the threshold value. There should be some kinds of penalties (punishments), e.g., in terms of money, against the cheated nodes, as well as necessary reimbursements for the suffered nodes.

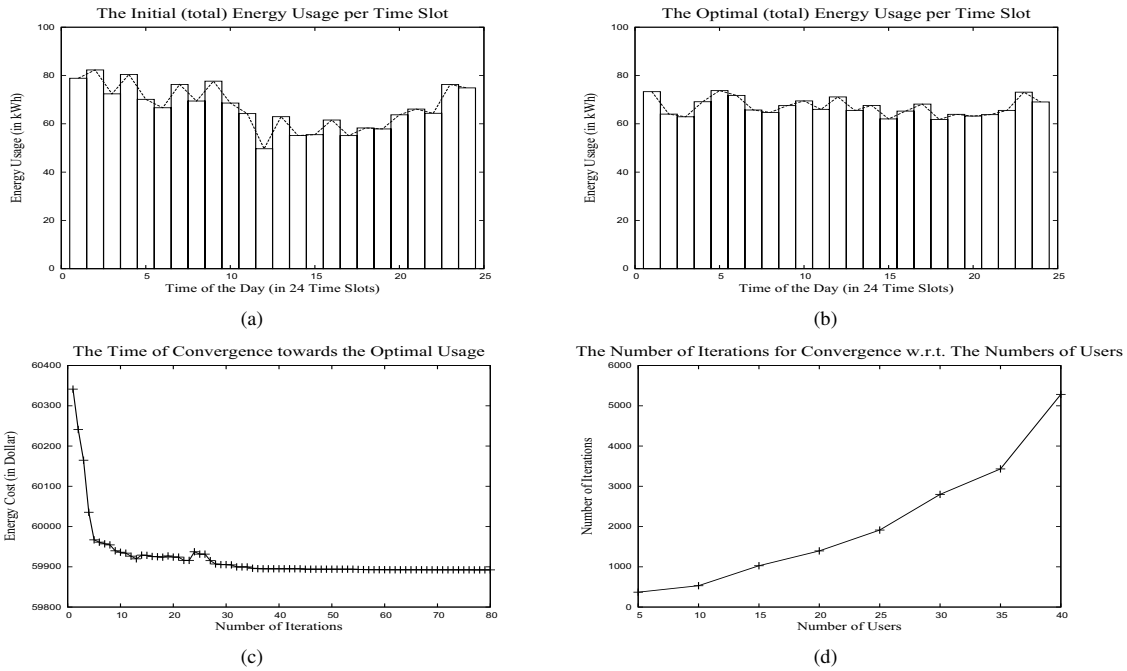

Fig. 7. (a) The initial total usage vector in different time slots (15 users), (b) the corresponding optimal total usage vector (after executing optimization process), (c) the energy cost (reduction) trend towards the optimal value with the number of iterations (rounds) of the optimization algorithm, and (d) the number of iterations that our proposed solution takes for convergence.

IV. Evaluation

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 the 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. These properties of the appliances are arbitrarily chosen. We do our experiment taking 10 to 25 users. 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 Figure 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 [1], 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. After the convergence, the summation of the optimal usage vectors of all users according to the example scenario of Fig. 7(a) is shown in Fig. 7(b). The second figure shows that the loads of the slots are more balanced than those of 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. 7(c). It 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 the users will save money, but also the power company will save a lot by reducing the investment in the capacity of their power plants. Fig. 7(d) shows the number of iterations that our solution takes to converge in case of different numbers of users.

In order to compare our proposed algorithm with the existing algorithm [1], we did a number of experiments by simulating 10, 15, 20, and 25 users. In 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 [1] are shown in Table I. From the table we can clearly observe that the running time of the existing method increases significantly with the

TABLE I
THE COMPARISON BETWEEN NON-CLUSTER METHOD AND THE CLUSTER BASED METHOD

		Non-Cluster Method		Cluster Based Method	
Users	**Initial Cost (Dollars)**	**Time (Seconds)**	**Cost (Dollars)**	**Time (Seconds)**	**Cost (Dollars)**
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

number of users. We observed that if the number of users is larger than 30, this algorithm took a significantly high running time. That is why, we did not run the simulation more than 30 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 method performs excellently well. In the case of 25 users, our solution is around 50 times faster than that of [1].

V. RELATED WORK

In this section, we briefly discuss the researches done for the optimized energy usage, especially with respect to the cost. M. Fahrioglu et al [3] proposed a game to help the interaction between a utility and its customers to let the customer help a utility solve a variety of problems. The idea of the game is to design an incentive structure which is able to encourage the player to make the right contract and revel their true value of power. N. Ruiz et al. [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 algorithm can obtain the maximum load reduction by determining the scheduling consumption strategies for the controllable users.

Mohsenian-Rad and Leon-Garcia proposed an optimal and automatic residential energy consumption scheduling framework in [9] 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 [1] for autonomous demand side management within a neighborhood. They considered the deployment of energy consumption scheduling (ECS) devices in smart meters, which can communicate with each other. The game theory is applied to distributively find the optimal consumption schedule vectors for all the users. None of these algorithms addresses the security problems introduced by their proposed algorithms. Li et al. [10] presented a distributed data aggregation process for smart meters involved in transmitting data from a set of meters to the data collector. To protect user privacy, they applied homomorphic encryptions. 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.

VI. CONCLUSION

Efficient electrical energy usage is very important, especially with the growing need of electricity. Leveraging the two-way communication capability of smart meters, some distributed solutions are proposed for the optimal energy consumption scheduling of household appliances of the energy users. This paper has presented a mutually exclusive cluster based solution for the optimal energy management problem, which efficiently solves the security problems found in the earlier proposed solutions. Our solution provides data privacy, as well as protection from false data injection. Comparing to the existing demand-side management solutions, the proposed approach is also highly efficient. The running time of the existing solutions increases quadratically with the increase of the number of users, while that of our solution increases almost linearly. Particularly, in our experiment with 25 users, our solution is around 50 times faster than the existing solution. We have also presented an example which shows that a user participating in the optimization process can get benefit by lying about its usage. We have proved that if an user cannot be untruthful about its total usage, then he cannot get any incentive by lying about the distribution of usage in different time slots. We have proposed an adjudicator (i.e., a truthful third party verifier) based solution in order to ensure the truthfulness of the participating users.

REFERENCES

[1] Amir-Hamed Mohsenian-Rad, et al. Autonomous Demand-Side Management Based on Game-Theoretic Energy Consumption Scheduling for the Future Smart Grid *IEEE Transaction on Smart Grid*, Vol 1, No 3, 2010.

[2] U.S. Department of Energy. 2010 Buildings Energy Data Book. *Energy Efficiency and Renewable Energy*, Mar 2011.

[3] Fahrioglu, M. and Alvarado, F.L. Designing incentive compatible contracts for effective demand management *Power Systems, IEEE Transactions on*, PP. 1255-1260, Vol 15, No 4, 2000.

[4] Krishnan, R. Meters of tomorrow *IEEE Power and Energy Magazine*, PP. 92-94, 2008.

[5] C. Triki and A. Violi. Dynamic pricing of electricty in retail markets Q. J. Oper. Res., Vol. 7, No. 1, pp. 2136, Mar. 2009.

[6] Masters, G.M. and ebrary, Inc. Renewable and efficient electric power systems *Wiley Online Library*, 2004.

[7] Ruiz, N., Cobelo, I. and Oyarzabal, J. A direct load control model for virtual power plant management *Power Systems, IEEE Transactions on*, Vol 24, No 2, PP. 959-966, 2009.

[8] Herter, K. Residential implementation of critical-peak pricing of electricity *IEEE Transactions on Power Delivery*, Vol 35, No 4, 2007.

[9] Mohsenian-Rad, A.H. and Leon-Garcia, A. Optimal residential load control with price prediction in real-time electricity pricing environments *IEEE Transactions on Smart Grid*, Vol 1, No 2, 2010.

[10] Li, F., Luo, B. and Liu, P. Secure information aggregation for smart grids using homomorphic encryption *First IEEE International Conference on Smart Grid Communications*, PP. 327-332, 2010.

[11] O. Goldreich. Foundations of Cryptography: Volume II (Basic Applications). *Cambridge University Press*, 2004.

[12] Atallah, M. and Du, W. Secure multi-party computational geometry *Algorithms and Data Structures*, PP.165-179, Springer,2001.