Secure Multiparty Computation Based Privacy Preserving Smart Metering System

Cory Thoma
Information Technology Leadership Department
Washington & Jefferson College
60 S. Lincoln St. Washington, PA 15301
Email: cmt69@pitt.edu

Tao Cui and Franz Franchetti
Department of ECE
Carnegie Mellon University
5000 Forbes Ave. Pittsburgh, PA 15213
Email: {tcui,franzf}@ece.cmu.edu

***Abstract*—Smart metering systems provide high resolution, realtime end user power consumption data for utilities to better monitor and control the system, and for end users to better manage their energy usage and bills. However, the high resolution realtime power consumption data can also be used to extract end user activity details, which could pose a great threat to user privacy. In this work, we propose a secure multi-party computation (SMC) based privacy preserving protocol for smart meter based load management. Using SMC and a proper designed electricity plan, the utility is able to perform real time demand management with individual users, without knowing the actual value of each user's consumption data. Using homomorphic encryption, the billing is secure and verifiable. We have further implemented a demonstration system which includes a graphical user interface and simulates network communication. The demonstration shows that the proposed privacy preserving protocol is feasible for implementation on commodity IT systems.**

I. Introduction

A smart meter is an energy meter (mostly an electric meter) that measures end user power consumption and communicates the information back to utility for monitoring and billing purposes. Smart meters usually sample at a time interval of less than one hour (i.e. 15 mins) [1]. Smart meters also utilize two way communication, which enables utilities to send pricing and/or direct control signals back to end users. With near realtime high resolution user consumption data and prompt price and control signals, the utility is able to monitor and control the loads in near realtime. The end users could also use the smart meter data to better manage their energy usage and reduce the energy bills. Smart metering systems enable new smart grid technologies such as flexible tariff, real time pricing, demand response, etc.

However, improper smart meter deployment can pose severe threats to end user privacy. Researchers have shown that fine grained power consumption data can be used to extract detailed information on user activities, e.g. if they are at home or on vacation, turn on or turn off a specific appliance, etc. [2] [3]. One such example is the "Nonintrusive Appliance Load Monitoring" (NALM) technique that uses smart meter measurement to identify which appliance in a household is switched on and off by detail analysis of total load [4] [5].

This work was supported by Semiconductor Research Corporation Smart Grid Research Center under Task No. 2111.006.

Fig.1 from [3] shows that using total consumption data, the detailed appliance status can be identified and further allow the user's activities being inferred, and the user's privacy to be violated. As a result, privacy issues have become one of the most significant barriers to large scale deployment of smart meters in some countries and regions [6] [7].

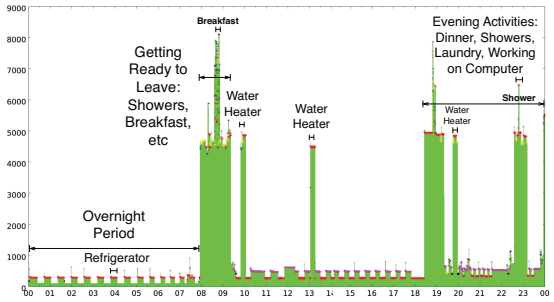

Fig. 1. Using consumption data to infer user activity [3]

Related work. In order to address the privacy issue, several privacy preserving smart metering schemes have been proposed. Based on homomorphic encryption techniques, a privacy friendly energy metering system has been proposed by Garcia and Jacob [8]. A similar method for securely aggregating neighborhood data has been proposed by Li., et.al. [9]. An anonymization process is proposed by Efthymiou and Kalogridis, which requires a third party anonymization process to remove the user's identity [10]. Another solution for future smart grid household proposed by Efthymiou et. al. uses batteries and other energy storage devices to make the load signature undetectable [11]. Microsoft researchers proposed a secured protocol for billing computation [12]. Cornell researcher proposed a Trust-Platform-Module based architecture [2], which aggregates the data of a user group over a certain time period. However, most of the previous works do not take real time control and management into consideration, and some methods mathematically or physically obfuscate the real time data which unnecessarily sacrifices the data resolution and limits the usability of the data. Some methods do not provide verifiable mechanism for billing. Some methods may still need a trusted third party.

In order to fully enable the advantage of smart metering

system and address its privacy concern, we believe an ideal privacy preserving smart meter framework should meet the following requirements: 1) fully protect the end user's privacy, 2) without sacrificing the resolution of smart meter data for actual load management usage, and 3) provide a verifiable billing method, 4) without using a trusted third party.

Contribution. In this work, we propose a secured architecture to address above issues. Secure multiparty computation (SMC) is a computation framework which enables multiple parties to secretly compute some joint value using a secured protocol without revealing the private data to anyone. The related homomorphic encryption enables the computation to be performed on encrypted data. By using SMC with homomorphic encryption properties, we design a smart meter based load management and billing framework that achieves all the above requirements. We further develop a demonstration system which includes the graphical user interface and runs on our campus network. The demonstration shows the feasibility of the proposed scheme.

Synopsis. This paper is organized as following: the concepts of secure multiparty computation and homomorphic encryption are reviewed in Section II. A smart meter based tariff and the SMC based protocol are detailed in Section III. The demonstration project is described in Section IV. Section V concludes the paper.

II. Background: Secure Multiparty Computation

Our proposed smart meter load management scheme is built upon several different but related concepts from cryptography, which we review in this section.

A. Public Key Encryption

Public key encryption [13] is an asymmetric encryption scheme. It uses key-pairs: a public key and a corresponding private key to securely transmit messages. The public key is used to encrypt a message (from plaintext to ciphertext). The corresponding private key is used to decrypt a message (from ciphertext to plaintext). The two keys are related mathematically, but parameters are chosen so that determining the private key from the public key is prohibitively expensive. Therefore, the public key can be publicly distributed to all the senders who want to send encrypted messages to the recipient while the private key is only possessed by the recipient, so that the recipient can decrypt the message encrypted by senders with corresponding public key.

B. Secure Multiparty Computation

Secure multi-party computation (SMC) is a cryptographic problem in which multiple parties jointly compute a value based on individually held private data, without sharing the data. The concept is closely related to the idea of *zero-knowledgeness* used in public key cryptosystems (for instance, RSA) and zero-knowledge authentication (for instance, Feige-Fiat-Shamir Identification Scheme) [13]. Security is often derived from one-way functions like integer multiplication/factorization that is easy in one way (polynomial time algorithm to multiply two prime numbers) but hard to invert (exponential time to find the original integers from the product). Historically, the first example of SMC was the *millionaire problem*: millionaires Alice and Bob are interested in knowing which of them is richer without revealing their actual wealth [14]. One of our secure primitives is built on the millionaire problem and will be discuss this in next section. A more general formulation of SMC is: for a number of players $P_1, \ldots, P_n$, each has initial inputs $x_1, \ldots, x_n$, and SMC securely computes some function f on these inputs, where $f(x_1, ...x_n) = (y_1, ..., y_n)$. Each player P_i only obtains the output y_i. During the computation process, the actually value of each player's input is kept privately without revealing to anyone.

C. Homomorphic Encryption and SMC

One standard approach to SMC is homomorphic encryption, which enables direct arithmetic operations on encrypted values. A prominent example is the Paillier cryptosystem [15], [16]. One simple illustrative example would be: Let $n = pq$, and p and q be distinctive prime numbers of sufficient size (1,024–2,048 bits). n is the public key and (p, q) is the private key. Also let $g = n + 1$. To encrypt a value $m \in \mathbb{Z}_m$, select a random value $r \in \mathbb{Z}_m$ and compute $[m] = (mn + 1)r^n = g^m r^n \bmod n^2$. [.] denotes an encrypted value; the public key is usually omitted since it is constant. For encrypted messages $[a]$ and $[b]$, and constants c, one can easily compute $[a+b] = [a][b]$ and $[ca] = [a]^c$. The computation of $[ab]$ usually requires a cryptographic protocol, since the Paillier cryptosystem is not homomorphic with respect to multiplication.

D. SMC Framework

The theoretical and algorithmic foundations for SMC are well-researched and it has been shown that SMC has the potential to solve hard problems in application areas that require strong privacy. Various approaches based on compilers and domain-specific languages exist: The Fairplay system [17] implements two-party SMC, and the FairplayMP extension [18] implements multiparty SMC. SMCL [19] is a domain-specific language for SMC. One large scale real-world application of SMC was a sugar beet auction system in Denmark [20]. However, to the best of our knowledge, SMC for control or management of infrastructure has not been investigated. From its definition SMC framework can not only protects multiple users' privacy, but also enable complicated arithmetic and logic operations, therefore SMC shows great potential in building privacy preserving smart meter systems.

III. Secure Multiparty Computation Based Smart Meter Load Management System

A. SMC Primitives

Two basic SMC primitives—*secure summation* and *secure comparison*—can be used to build the proposed privacy preserving smart meter control and management system.

Secure summation. Secure summation can be built upon the Paillier cryptosystem. As mentioned in Section II, Paillier

is an additive homomorphic cryptosystem. The following steps describe the details of Paillier scheme.

1) **Key generation**: Choose two large prime p and q randomly and independently. Let $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$. Choose random $g \in \mathbb{Z}^*_{n^2}$, ensure n divides the order of g by checking the existence of $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$ where $L(x) = (x-1)/n$. The public key is (n, g). The private key is (λ, μ).
2) **Encryption**: Given the public key (n, g) and a random number $r \in \mathbb{Z}^*_n$. A plaintext M can be encrypted to ciphertext $[M]$, $[M] = g^M \cdot r^n \bmod n^2$. The encryption process is denoted as: $[M] = E(M)$.
3) **Decryption**: Given the public key (n, g) and private key (λ, μ), a ciphertext $[M]$ can be decrypted to plaintext M by $M = L([M]^\lambda \bmod n^2) \cdot \mu \bmod n$. The decryption process is denoted as: $M = D([M])$.

The two homomorphic encryption properties of Paillier cryptosystem can be written as follows.

$$D(E(m_1) \cdot E(m_2) \bmod n^2) = m_1 + m_2 \bmod n. \quad (1)$$

$$D(E(m_1)^k \bmod n^2) = km_1 \bmod n. \quad (2)$$

The additive homomorphic encryption in (1) means that given only the public-key and the encrypted $[a]$ and $[b]$, one can compute the encrypted $[a+b]$ without know actual value of a and b. The homomorphic multiplication of plaintexts in (2) means that given only the public-key, the encrypted $[a]$ and a constant plaintext b, one can compute the encrypted $[ab]$ without know actual value of a.

Based on the additive homomorphic encryption property of Paillier cryptosystem, a *secure summation* process to compute the summation of User 1 to User n's consumption data can be implemented as described below.

1) **Set-up**: Using Paillier system, the public key (n, g) is given to all users and the utility. The private key (λ, μ) is only given to utility.
2) **Start**: Starting from User 1, User 1 encrypts his/her usage P_1 to ciphertext M_1, where $[M_1] = E(P_1)$, and send $[M_1]$ to next user.
3) **Encrypted Addition**: When User i received message $[M_{i-1}]$ from User $i-1$, User i encrypts his/her consumption data P_i to $[P_i] = E(P_i)$, and send the new message $[M_i] = [P_i] \cdot [M_{i-1}] \bmod n^2$ to next User $i+1$.
4) **Stop**: When the message $[M_n]$ from the last User n finally send to the utility. The utility decrypt the message $M_n = D([M_n])$ using the private key.

   Then $M_n = \sum_{i=1}^{n} P_i$.

In this way, the utility is able to decrypt the final message to obtain the summation without know actual value of any user. Therefore, the privacy preserving *secure summation* primitive can be achieved.

Secure comparison. The secure comparison is derived from Yao's millionaire example. Millionaire Bob wants to know if he is wealthier than millionaire Alice, without either of them revealing their actual wealth. Alice and Bob use the public-key cryptosystem RSA [13]. Alice has an RSA private/public key pair. In addition, they need to know an upper bound k of their wealth. All wealth values are rounded to millions to keep k a reasonably small number. Alice is worth a millions, and Bob b millions.

- *Step 1.* Bob picks a random n-bit integer called x (Alice will later use a $n/2$ bit prime, so the length of the integer is important). Bob first calculates c as the RSA encipherment of x using Alice's public key.
- *Step 2.* Bob transmits $c - b + 1$ to Alice.
- *Step 3.* Alice generates a series of numbers $y_1, y_2, \ldots, y_k$ such that y_i is the RSA decipherment (using her private key) of $c - b + i$.
- *Step 4.* Alice now generates a random $n/2$ bit length prime p. Alice then generates $z_1, z_2, \ldots, z_k$ by calculating $z_i = y_i \bmod p$. Note that p must be chosen so that all the z_i differ by at least 2.
- *Step 5.* Alice now transmits the prime p to Bob, and then sends k numbers u_i. The first few u_i are $u_1 = z_1, u_2 = z_2, \ldots, u_a = z_a$ with a being Alice's worth in millions. Then Alice adds 1 to all the remaining $k - a$ values u_i to be sent and sends $u_{a+1} = z_{a+1} + 1, \ldots, u_k = z_k + 1$.
- *Step 6.* Bob receives p and $u_1, \ldots, u_k$. He computes $g = x \bmod p$. If the $u_b = g$ then Alice is equal or greater to Bob in wealth ($a \geq b$). If the $u_b \neq g$ then Bob is wealthier than Alice ($a < b$).

This has all been done without either of them transmitting their wealth value and cryptanalysis shows that the values haven't been leaked [14]. The procedure is redone with switched roles to let Alice know if or if not she is wealthier than Bob.

In the smart meter based load management system application, the above millionaire problem can be translated to the comparison of the utility's pre-set consumption threshold and the user's realtime consumption data to to find out if the user is over certain limit, without revealing user's consumption data.

B. SMC Based Demand Management

In this section, we describe the proposed electricity plan for smart meter based load management. We also show how SMC primitives can enable such realtime load management without revealing any knowledge of private user consumption data.

Customer electricity plan. We now discuss a proposed electricity plan similar to a cell phone plan. In our set-up, all customers have SMC-capable smart meters installed. Customers buy electricity plans from their utility with two service states, *normal* and *high*. Similar to cell phone plans, electricity plans in our setup also cover a certain number high demand allowance kWh (kilowatt-hours) per month which can be used in high demand state. If customers exceed their high demand kWh budget in high demand state, they pay a premium on the excess kWh. Electricity consumed during low demand is charged at the normal rate. The smart meter signals customers in which of the two states they operate at any given point in time and how many high demand kWh

they have left. Customers can choose to adjust their energy consumption, similar to how people manage their peak-hour cell phone minutes. In addition, the smart meter has a standard interface to consumer appliances to signal to them the current consumption mode and to enable or disable smart appliances. At the end of the month the smart meter sends the total number of kWh for each state (normal, high demand) to the utility for billing. This is a simple customer interface that has proven to work for phone service; the hope is that it is simple enough for electricity customers.

From customer side, the details of the electricity plan are set-up as follows:

1) Each User i has a state variable U_s^i denoting his/her demand state. U_s^i could be either: *normal* or *high* demand.
2) Each User i has a high demand allowance N_H^i kWh per month covered by the plan.
3) When $U_s^i = normal$ User i is charged with normal rate.
4) When $U_s^i = high$, but User i has high demand allowance N_H^i, User i will be charged with normal rate and his/her N_H^i begin to decrease according to usage.
5) When $U_s^i = high$, and $N_H^i = 0$. User i will be charged with high rate.

From utility side, the users are divided into groups according to the plan and their history load profile patterns. The details of the electricity plan set-up from utility side are as follows:

1) Each group has a state variable G_s denoting the *peak* or *offpeak* states: G_s =*peak* or *offpeak*
2) Utility computes the total group consumption P_Σ^G and set a threshold $Thres^G$ for the group G.
3) When for the group G, $P_\Sigma^G \geq Thres^G$, then $G_s = peak$, otherwise G_s =*offpeak*.
4) When G_s =*peak*, a group-wide user threshold $Thres_{user}^G$ is computed. It could be the average user consumption in G.
5) For the User i in G, if the consumption $P_i \geq Thres_{user}^G$, User i will enter $high$ demand state: $U_s^i = high$. Otherwise, $U_s^i = normal$.

The detailed implementation of the proposed electricity plan is summarized in Algorithm 1.

Discussion. Several features of this electricity plan:

1) Dynamical management: The utility is able to dynamically determine the system's status (peak or offpeak) based on the realtime consumption of the customer groups.

2) Customer experience: Within the same customer group, the utility selects the customers with higher consumption to enter the high demand state first. Those customers can choose to lower the consumptions or use their high demand allowance hours. With this plan, the customer with higher demand may consider to carefully plan their energy usage.

3) Security of real time data: The proposed demand management algorithm with SMC primitives preserves privacy in user consumption data: in above Algorithm 1, the variables in [.] are all encrypted by individual user. By using SMC primitives *secure summation* and *secure comparison*, the required functions for load management can be performed without revealing private user information to anyone.

Algorithm 1 Proposed Demand Management Algorithm

{Load management loop at each time interval}
loop
  {***Secure summation***}
  $P_\Sigma^G \leftarrow [\sum_{i \in G} P_i]$
  if $P_\Sigma^G > Thres^G$ **then**
    G_s = *peak*
    $Thres_{user}^G = f_G(P_\Sigma^G)$
  else
    G_s = *offpeak*
    All User Rate = Normal
  end if
  if G_s = *peak* **then**
    Broadcast G_s = *peak*
    for $i \in G$ **do**
      {***Secure comparison***}
      if $[P_i] > [Thres_{user}^G]$ **then**
        $U_s^i = high$
        $N_H^i = N_H^i - 1$
        if $N_H^i \leq 0$ **then**
          User i Rate = High
        end if
      else
        $U_s^i = normal$
        User i Rate = Normal
      end if
    end for
  end if
end loop

C. Billing and Verification

The billing and verification is achieved by *homomorphic encryption*. As described in Section II, homomorphic encryption is a form of encryption which allows specific types of computations to be carried out on ciphertext (encrypted data) and obtain an encrypted result which is the ciphertext of the result of operations performed on the plaintext.

The idea of privacy preserving billing and verification is to let both the utility and the user's smart meter calculate the user's bill in real-time as electricity is consumed in low or high demand mode. However, the bill calculation by utility is carried out on encrypted data received from the user to hide the instantaneous consumption from the utility. The bill calculation by user's smart meter is carried out on plaintext. At the end of each month, the user's plaintext bill is sent to the utility as final bill total, and can be encrypted and compared to utility's encrypted bill value to ensure the correctness of the user-calculated bill.

The set-up of the billing from user side: each user i's smart meter has two counters for each day j: $U_{normal}^i(j)$: normal price usage counter and $U_{high}^i(j)$: high price usage counter. The normal price usage counter accumulate the user's energy usage when the price is set at normal rate, while the high price usage counter accumulate the user's energy usage when price

is set at high rate. The following steps are implemented for billing and verification:

In the algorithm, we still use Paillier cryptosystem.

- *Step 0.* Using Paillier system. User i has the public key and private key. The utility only has the public key.
- *Step 1.* At the end of each day j, User i use public key to encrypt daily $U^i_{normal}(j)$ and $U^i_{high}(j)$. The encrypted $[U^i_{normal}(j)]$ and $[U^i_{high}(j)]$ are send to the utility for record.
- *Step 2.* At the end of each month, using the public key, the utility is able to compute the encrypted total usage of User i: $[U^i_{normal}] = \sum_j [U^i_{normal}(j)]$ and $[U^i_{normal}] = \sum_j [U^i_{high}(j)]$. Since the prices are given constants. The total encrypted bill can also be computed as: $[Bill_i] = [U^i_{high} P_{high} + U^i_{normal} P_{normal}]$.
- *Step 3.* At the end of each month, the user i also computes his/her total usage U^i_{normal} and U^i_{high} in plaintext. The total bill in plaintext is $Bill_i = U^i_{high} P_{high} + U^i_{normal} P_{normal}$.
- *Step 4.* Utility send the encrypted total usage and bill to User i, and User i send plaintext total usage and bill to the utility.
- *Step 4.* For verification by utility, using the public key, utility can encrypted the plaintext usage and bill received from User i to see if the data match the encrypted data of the utility.
- *Step 4.* For verification by User i, using the private key, User i can decrypted the ciphertext usage and bill received from the utility see if the data match his/her own meter's record.

Discussion. The billing and verification scheme utilizes the homomorphic addition of ciphertexts in (1) and the homomorphic multiplication of plaintexts in (2). The utility is able to keep (encrypted) records of user's daily usage and to compute the (encrypted) total bill without knowing the actual value. Both user and utility can make sure the bill is correct at the end of each month. One assumption of the verification is that the smart meter can correctly record the usage data.

IV. Implementation

In this section, we discuss a prototypical demonstration implementation of our proposed privacy preserving scheme using commodity computing and network environments. We also build a smart meter simulation environment to simulate the interaction between the users and the utility using the proposed privacy preserving scheme.

A. Network Topology

The system is a network consisting of smart meters and the utility server. The network topology of the proposed privacy preserving load management is showed in Fig. 2. There are two structures for different security primitives:

1) The outer ring structure is used for *secure summation*: at the end of each time interval, the utility starts the Paillier Cryptosystem enabled secure summation from the the first user. The first user then encrypted his/her consumption data to the next

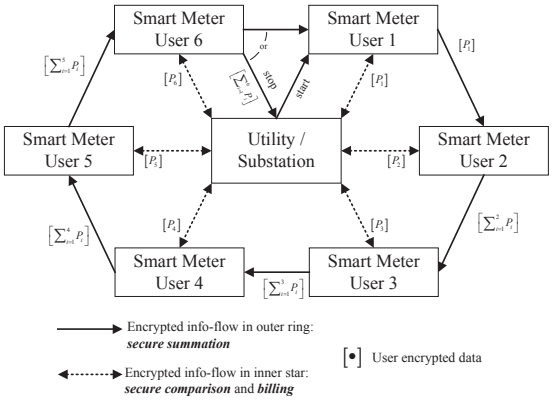

Fig. 2. Network topology of smart meters and the utility server.

user. Each user add his/her own encrypted consumption data to the encrypted data received from precious user. Then, finally, the total encrypted sum is sent to utility. Then the utility is able to decrypt the encrypted sum and obtain the summation of all user's total consumption data. Note in Fig. 2, all data in [.] are encrypted by user him/herself.

2) The inner star structure used for *secure comparison* and *billing*: at the end of each time interval, once the utility obtained the total consumption of the user group. The *secure comparison* is performed between each user and the utility. Also, at the end of each day, each user sends his/her encrypted usage data to the utility. At the end of each month, the billing calculation and verification is performed between the utility and each user.

B. User Interface

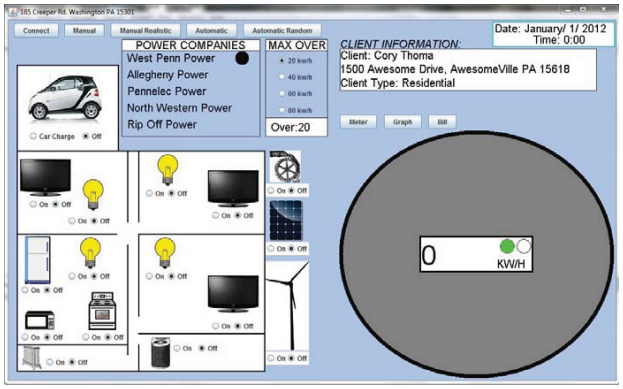

Fig. 3. The the customer smart meter interface.

The GUI customer interface simulates as the control panel of a smart house. The customer is able to monitor the household real time consumption as well as the electricity price and bill. The customer is also able to monitor and control various appliances including TV, heater, refrigerator, oven, electric vehicle charging and user owned renewable generations.

The substation interface is designed to monitor the user group's total consumption and system status (peak or offpeak). The utility can also use the interface to set the thresholds for system status and user demand status.

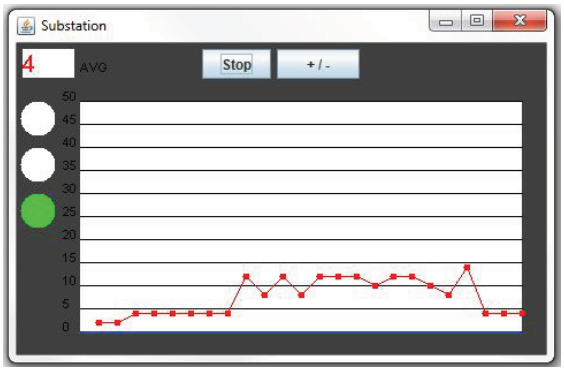

Fig. 4. The substation interface for the utility.

C. Simulation

Based on the proposed load management scheme and its privacy preserving implementation, we developed a simulation environment to simulate the system reasonably realistically.

During the simulation, the customer can choose to switch on/off specific appliance by using the customer interface in Fig. 3. There are also several simulated appliance behaviors triggered by the buttons on the upper left conned on Fig. 3. The "Manual" gives the user complete control over all appliances. The "Manual Realistic" adds some realistic restrictions to what the user can do. The "Automatic" is a fully automated simulation of a household, the household appliance will be turned on or off according to pre-programmed sequence. The "Automatic Random" adds some randomness upon the "Automatic" simulation.

D. Hardware and Software Infrastructure

The user interface and network communication system are implemented using the Java programming language. The networking is built on a library provided by Apache called Apache MINA. The demonstration and simulation is running on a desktop computer with 2.53 GHz 2 Core CPU with 4 GB memory. The computational power necessary for the simulation shows that an embedded processor in a smart meter could easily meet the performance requirements for a real-world deployment of our proposed approach.

V. Conclusion

In this paper we propose and implement a privacy preserving smart meter based load management system. We use *secure multi-party computation* and *homomorphic encryption* as the security primitives. Our scheme fulfills four conditions that are desirable for a privacy preserving smart meter load management system: 1) it is able to fully preserve the detailed users data, 2) it does not sacrifice the data resolution for proposed smart grid control and management functionalities, 3) it has a verification process, and 4) it does not need a trusted third party. We further implemented the proposed scheme as a networked simulation on standard commodity hardware. Our system provides a GUI that allows prospective users to experience our proposed electricity plan and the smart meter load management scheme. The demonstration also shows that our proposed privacy preserving scheme is feasible on current commodity hardware and thus could be deployed on a real system.

References

[1] "Assessment of Demand Response and Advanced Metering," Federal Energy Regulatory Commission, Tech. Rep., 2008.

[2] S. Wicker and R. Thomas, "A privacy-aware architecture for demand response systems," *Hawaii International Conference on System Sciences*, vol. 0, pp. 1–9, 2011.

[3] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin, "Private memoirs of a smart meter," in *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, ser. BuildSys '10. New York, NY, USA: ACM, 2010, pp. 61–66. [Online]. Available: http://doi.acm.org/10.1145/1878431.1878446

[4] G. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[5] Y. Kim, T. Schmid, Z. Charbiwala, and M. Srivastava, "ViridiScope: design and implementation of a fine grained power monitoring system for homes," in *Proceedings of the 11th international conference on Ubiquitous computing*. ACM, 2009, pp. 245–254.

[6] P. McDaniel and S. McLaughlin, "Security and privacy challenges in the smart grid," *IEEE Security & Privacy*, pp. 75–77, 2009.

[7] H. Khurana, M. Hadley, N. Lu, and D. Frincke, "Smart-grid security issues," *Security & Privacy, IEEE*, vol. 8, no. 1, pp. 81–85, 2010.

[8] F. D. Garcia and B. Jacobs, "Privacy-friendly energy-metering via homomorphic encryption," in *In 6th Workshop on Security and Trust Management (STM 2010)*, ser. Lecture Notes in Computer Science, J. C. et al., Ed. Springer Verlag, 2010.

[9] F. Li, B. Luo, and P. Liu, "Secure information aggregation for smart grids using homomorphic encryption," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, 2010, pp. 327 –332.

[10] C. Efthymiou and G. Kalogridis, "Smart Grid Privacy via Anonymization of Smart Metering Data," in *2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2010, pp. 238–243.

[11] G. Kalogridis, C. Efthymiou, S. Denic, T. Lewis, and R. Cepeda, "Privacy for smart meters: Towards undetectable appliance load signatures," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, 2010, pp. 232 –237.

[12] A. Rial and G. Danezis, "Privacy-preserving smart metering," Microsoft technical report MSR-TR-2010-150, Tech. Rep., 2010.

[13] L. C. Washington and W. Trappe, *Introduction to Cryptography: With Coding Theory*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2002.

[14] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, ser. SFCS '82. Washington, DC, USA: IEEE Computer Society, 1982, pp. 160–164.

[15] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, 1999, pp. 223–238.

[16] M. Franz, B. Deiseroth, K. Hamacher, S. Jha, S. Katzenbeisser, and H. Schroeder, "Secure computations on non-integer values," Cryptology ePrint Archive, Report 2010/499, 2010.

[17] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay—a secure two-party computation system," in *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, ser. SSYM'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 20–20. [Online]. Available: http://portal.acm.org/citation.cfm?id=1251375.1251395

[18] A. Ben-David, N. Nisan, and B. Pinkas, "Fairplaymp: a system for secure multi-party computation," in *Proceedings of the 15th ACM Conference on Computer and Communications Security*. New York, NY, USA: ACM, 2008, pp. 257–266.

[19] J. D. Nielsen and M. I. Schwartzbach, "A domain-specific programming language for secure multiparty computation," in *Proceedings of the 2007 workshop on Programming languages and analysis for security*, ser. PLAS '07. New York, NY, USA: ACM, 2007, pp. 21–30.

[20] P. Bogetoft, D. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. Nielsen, J. Nielsen, K. Nielsen, J. Pagter *et al.*, "Secure multiparty computation goes live," *Financial Cryptography and Data Security*, pp. 325–343, 2009.