Privacy-Preserving and Secure Communication Scheme for Power Injection in Smart Grid

Prem Akula*, Mohamed Mahmoud*, Kemal Akkaya†, and Min Song‡

*Department of Electrical & Computer Engineering, Tennessee Tech University, Cookeville, TN, USA
†Department of Electrical and Computer Engineering, Florida International University, Miami, FL, USA 33174
‡Department of Computer Science, Michigan Tech University, Houghton, MI, USA

Abstract—**In this paper, we propose a secure and privacy-preserving communication scheme for power injection for the smart grid storage units. The idea is based on collecting masked bids from storage units at an aggregator and send aggregated bids to the utility rather than the individual ones. The utility company can ensure the integrity and authenticity of the aggregated bid without accessing the individual bids. In this way, no party will have access to the storage units' data and make use of it to achieve unfair financial gains. Our evaluations have demonstrated that the proposed scheme is secure and can achieve the privacy requirements. Moreover, the scheme requires acceptable communication and computation overhead.**

Keywords: Privacy preservation, secure communications, power injection, smart grid, energy storage units, and PHEVs.

I. Introduction

The smart grid uses two-way communication technologies and computational intelligence in an integrated fashion across electricity generation, transmission, distribution and consumption to achieve a system that is clean, secure, resilient, and efficient [1], [2]. One of the main objectives of the smart grid is the reduction of the greenhouse gas emissions by reducing the dependency on fossil fuels. This can be done by facilitating the large-scale deployment of renewable energy generators, such as photovoltaic or wind power generators, and promoting full (or hybrid) Plug-in Electric Vehicles (PHEVs) [3].

It is expected that renewable energy will play an important role in the future power grid. However, the output power fluctuation of renewable energy generators may cause excess variations of the grid's voltage and frequency. In recent years, storage units have been used with renewable generators to provide an energy reserve with less fluctuating output power [3]. The power storage units can store the excess power at the period of strong sun or wind and inject power to the grid during peak hours. The same outcome can be achieved via the use of PHEVs' batteries as storage units [4].

These developments hint that the smart grid will have a large number of storage units in the very near future. The storage unit owners and the utility company will mutually benefit from this process. Utilities can meet the peak hour demand, thereby flattening the daily load curve and reducing both generation and investment in power generation, whereas the the storage unit owners can make revenue by participating effectively. In order to achieve such an outcome, the utility companies should have the ability to communicate with the storage unit owners. This means, from the technical perspective, appropriate communication protocols are needed to exchange information among the involved parties and enable them to conduct fair transactions.

However, such communications with the storage units can reveal sensitive information about the owners (e.g., the location of an PHEV, the amount of power injected, and other information about the activities of the people). For instance, if a house's battery injects too much power, this means the dwellers do not spend much time at home and they may be on travel. This poses a privacy threat for the owners that may make them reluctant to get involved. More importantly, the leaked information can be used for monetary gains as the utility company wants to buy energy with minimum price and the storage units want to maximize their revenues. This can motivate the parties involved in the power selling transactions to misbehave. For example, if a storage unit knows that the other units in the community have low power storage, they can refuse selling power to force the utility to offer high prices. Without hiding such information, the storage unit owners or the utility company may suffer from financial losses and the market will not be fair, which may discourage buying renewable resource generators.

In this paper, we tackle the above problem and propose a privacy-preserving and secure communication scheme for power injection in the smart grid. Our objective is to preserve the sensitive information that can be misused by some parties to maximize their revenue by using security constructs and aggregation. In our scheme, when the utility company wants to buy energy, it sends the proposed power prices in different time slots to the storage units. Each storage unit prepares its bid that should indicate the amount of power it can inject in each time slot with the announced prices. Each storage unit masks its bid using a one-time key shared with the utility. An aggregator is assumed for each neighborhood/community that aggregates all masked bids of the storage units and sends to the utility. In this way, the utility can only know the total amount of power that can be injected from the community, but it cannot know whether a specific storage unit will inject power or the amount of power that will be injected by each unit. The utility company can ensure that the aggregated bid is sent from the intended storage units without accessing the individual bids and also to ensure that the bids have not been modified in transit.

Our evaluations have demonstrated that the proposed

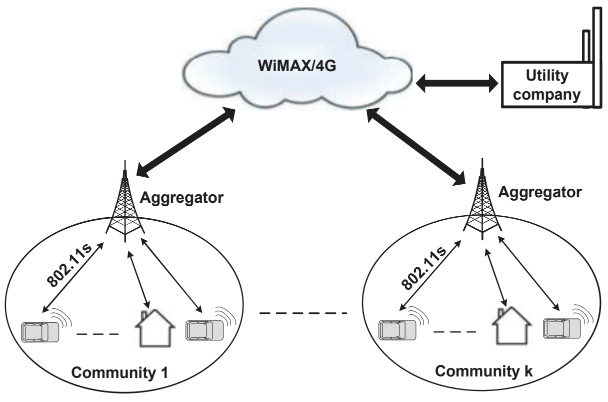

Fig. 1: The considered network model.

scheme is secure and can achieve the privacy requirements. Moreover, the scheme requires acceptable communication and computation overhead.

The remainder of this paper is organized as follows. Section II discusses the network and threat models. The proposed scheme is presented in Section III. Security/privacy analysis and overhead evaluation are presented in Section IV. The related works are summarized in V. Finally, conclusions are drawn in Section VI.

II. Network and Threat Models

A. Network Model

As illustrated in Fig. 1, the considered network model has a number of communities and a utility company. Each community can be one neighborhood, all loads connected to the same electric bus, etc. It has a group of energy storage units and one aggregator. In the following, we elaborate on the the network entities.

- **Storage Units**: The storage units can be PHEVs or batteries installed in homes. They may store energy from renewable energy sources or from the grid. Actually, power can be bought from the grid at low-price periods and injected to the grid at high prices. The decision of buying or selling energy will be taken individually by each storage unit. The storage units can communicate with the utility company via an aggregator. Purchasing power from the grid is out of scope of this paper that only focuses on power injection to the grid.
- **Aggregator**: Each community has one aggregator that forwards the storage units' packets to the utility and forwards the utility's packets to the storage units. As illustrated in Fig. 1, the communications between the aggregator and the storage units can be implemented by IEEE 802.11s-based wireless mesh technology and the communications between the aggregator and the utility can be implemented by WIMAX or 4G technology [5].
- **Utility Company**: Utility company can calculate the grid state and know when it can ask the communities to inject power. It cannot communicate with the storage units directly, but this has to be via the aggregators.

B. Adversary and Threat Model

The utility company is assumed to be honest-but-curious. It does not aim to disrupt the communications or manipulate the data, but it is curious to know which storage unit will inject power and the amount and the time of the power injection. It aims to predict the level of the power availability in the community to propose low prices that can be accepted by the storage units. The aggregator is also curious to know the amount and time of the power injection of each storage unit.

External attackers and storage units can eavesdrop on the communications to gain information about the bids of the storage units. We do not assume that the utility company colludes with the storage units of one community because they have conflicting interests.

III. Privacy-preserving Power Injection Scheme

In the following, we first describe how the cryptographic credentials are distributed to the involved parties and then describe the message formats between storage units and utility.

A. Bootstrapping

Before the communications start, an off-line trusted third party (TTP) bootstraps the system to setup some parameters and keys. TTP creates a finite field Z_q^*, an additive group G_a, and a multiplicative group G_m, where q is a large prime number. The order of Z_q^*, G_a, and G_m is q. Q is the generator of the G_a. ê is a bilinear pairing function that maps elements in G_a to elements in G_m, where $\forall$ P_1 and $P_2 \in G_a$, $\hat{e}(P_1, P_2) \rightarrow G_m$ [6].

Let $h_1()$ and $h_2()$ be two one-way hash functions, where h_1: $\{0,1\}^* \rightarrow G_a$ and h_2: $\{0,1\}^* \rightarrow Z_q^*$. The TTP chooses a random element $sk_t \in Z_q^*$ and computes PK_t = sk_tQ, where sk_t and PK_t are its private and public keys, respectively. For each community aggregator, the TTP chooses a random element $sk_a \in Z_q^*$ and computes PK_a = sk_aQ, where sk_a and PK_a are the private and public keys, respectively. Similarly, for the utility company, the TTP chooses a random element $sk_u \in Z_q^*$ and computes PK_u = sk_uQ, where sk_u and PK_u are the private and public keys, respectively. For each storage unit with identity id_i, the TTP chooses a random element sk_i $\in Z_q^*$ and computes PK_i = sk_iQ, where sk_i and PK_i are the private and public keys, respectively.

The TTP generates a certificate for each entity in the system to bind its public key to its identity [7], [8]. A typical certificate should have the entity's identity, public key, expiration date, and the TTP's signature. With using the certificates and signatures, the storage units can prove that they are legitimate members in the system. The TTP should issue, renew and also revoke the certificates if necessary. For every storage unit that joins the system, it should contact the TTP to receive the necessary credentials. Finally, the TTP publishes $\{Z_q^*$, G_a, G_m, q, Q, PK_t, $h_1()$, $h_2()$, ê$\}$ and keeps sk_t secret.

B. The Proposed Scheme

When the utility company needs to buy power, it sends *Power Purchase Request* packet (*Power_Req*) to the aggregator

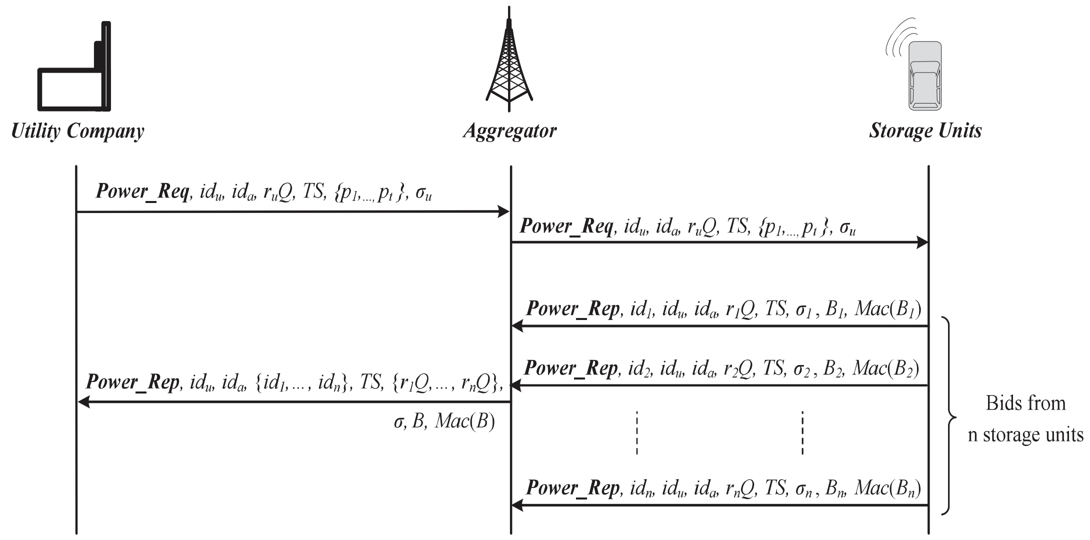
Utility Company
Aggregator
Storage Units
Power_Req, id_u, id_a, r_uQ, TS, $\{p_1,...,p_t\}$, σ_u
Power_Req, id_u, id_a, r_uQ, TS, $\{p_1,...,p_t\}$, σ_u
Power_Rep, id_1, id_u, id_a, r_1Q, TS, σ_1, B_1, Mac(B_1)
Power_Rep, id_2, id_u, id_a, r_2Q, TS, σ_2, B_2, Mac(B_2)
Power_Rep, id_u, id_a, $\{id_1,..., id_n\}$, TS, $\{r_1Q,..., r_nQ\}$, σ, B, Mac(B)
Power_Rep, id_n, id_u, id_a, r_nQ, TS, σ_n, B_n, Mac(B_n)
Bids from n storage units

Fig. 2: Bids submission.

that forwards the request to the storage units in its community. The requests are made based on assumed time slots. Then, the storage units send their bids in *Power Reply* packets (*Power_Rep*) to the aggregator that aggregates them and sends an aggregated bid to the utility. Fig. 2 summarizes the whole process of message exchanges among different parties but the details are explained in the following subsections.

1) Power Requests from Utility: To start the process of purchasing power, the utility company should send power request packet to the community's aggregator. The packet has the identities of the utility company and aggregator, id_u and id_a. It also has the list of power purchase prices per unit in each time slot $\{p_1, p_2, ..., p_t\}$, where t is the number of time slots. The utility selects a random element $r_u \in Z_q^*$ and attaches r_uQ to the packet. It also attaches a timestamp (TS) and a signature (σ_u), where, $\sigma_u = sk_u h_1(id_u, id_a, r_uQ, TS, \{p_1, p_2, ..., p_t\})$.

After receiving the packet, the aggregator first verifies the freshness of the packet by making sure that the timestamp is within acceptable range. Then, it verifies the signature by checking that $\hat{e}(\sigma_u, Q) \overset{?}{=} \hat{e}(h_1(id_u, id_a, r_uQ, TS, \{p_1, p_2, ..., p_t\}), PK_u)$. The proof of this verification is as follows.

$$\begin{aligned}\hat{e}(\sigma_u, Q) &= \hat{e}(sk_u h_1(id_u, id_a, r_uQ, TS, \{p_1, ..., p_t\}), Q)\\ &= \hat{e}(h_1(id_u, id_a, r_uQ, TS, \{p_1, ..., p_t\}), sk_uQ)\\ &= \hat{e}(h_1(id_u, id_a, r_uQ, TS, \{p_1, ..., p_t\}), PK_u)\end{aligned}$$

Finally, the aggregator broadcasts the utility's request to the storage units in its community.

2) Bids from Storage Units to Aggregetor: Each storage unit should prepare its bid that has the the amount of power it can inject in each time slot. The format of the bids is given in Fig. 3. Each group of bits represents the amount of power the storage unit can inject in one time slot. If a storage unit does not want to inject power, either because it does not have excess power or the proposed purchase prices are low, it should send a bid that has zeroes in all time slots.

The storage units should send back *Power Request Reply* packet (*Power_Rep*). The packet has its identity (id_i), the identities of the aggregator and utility, r_iQ, timestamp, σi, masked bid (B_i) and $Mac(B_i)$. The signature σ_i = $sk_i h_1(id_i, id_u, id_a, TS, r_iQ)$. The storage unit chooses a random element $r_i \in Z_q^*$ and computes r_iQ. Each storage unit should calculate a shared key with the utility as follows $k_i = h_2(\hat{e}(PK_u,\ sk_i r_i r_u Q))$. This key is used to mask the storage unit's bids and enable the utility company to ensure the aggregated bid's integrity and authenticity without accessing the individual bids. The storage unit's masked bid B_i = $(b_i + k_i)Q$, where b_i is the bid. Each storage unit computes $Mac(B_i)$ = ê$((b_i + h_2(k_i))Q, Q)$.

3) Aggregated Bid from Aggregetor to Utility: After receiving all the *Power_Rep* packets from all the storage units, the aggregator sends an aggregated *Power_Rep* packet to utility. The aggregator aggregates the storage units' signatures, bids, and $Mac(B_i)$ to produce an aggregated signature (σ), an aggregated masked bid (B) and $Mac(B)$. The bids are aggregated for the following reasons: 1) preserving the privacy of the storage units' bids; and 2) reducing the required bandwidth to send the data to the utility by reducing the data size. Moreover, the utility company does not need the individual storage units' power injection data, but it needs the total power that can be injected from the community in each time slot. The aggregated *Power_Rep* packet has the identities of the storage units and the list of r_iQ, where $1 \leq i \leq n$ and n is the number of storage units. It also has the aggregated signature σ, the aggregated masked bids (B), and $Mac(B)$.

The aggregator aggregates the bids to calculate the aggregated bid B, where $B = \sum_{i=1}^{n} B_i = \sum_{i=1}^{n} (b_i + k_i)Q$. In b_i, the number of bits assigned to each time slot depends on the maximum amount of power a storage unit can inject and the number of storage units in the community. To avoid making mistakes, the number of bits should be selected so that the total summation of the power injection of the storage units in

b_1	0000110	0011000	………	0000000
b_2	0000111	0010100	………	0001000
b_3	0001000	0010100	………	0010001
b_4	0010000	0000000	………	0011010
$\sum_{i=1}^{4} b_i$	0100101	1000000	………	0110011

Fig. 3: Format of power injection bids. b_i represents a group of time slots. In each slot, each storage unit puts their power amount which is shown as a group of 7 bits in the figure.

one time slot does not cause a carry added to the next time slot. For instance, in Fig. 3, if 4 bits are assigned to the amount of power injection, each slot should have at least 7 bits in b_i when n = 4. This can guarantee that the summation of the bids does not produce a carry from one time slot that is added to the next time slot. In order to enable the utility company to ensure that the bids are sent from the intended storage units and they have not been modified in transit without accessing the individual bids to preserve privacy, the aggregator sends $Mac(B)$. $Mac(B)$ is calculated by using point multiplication operations over G_m, where $Mac(B) = \prod_{i=1}^{n} Mac(B_i)$, where $Mac(B_i) \in G_m$.

The aggregator should verify the storage units' signatures to ensure the packets' authenticity and integrity. This can be done by verifying that $\hat{e}(\sigma_i, Q) \overset{?}{=} \hat{e}(h_1(id_i, id_u, id_a, TS, r_iQ), PK_i)$. However, the computation overhead can be reduced by using a batch verification technique [9] instead of verifying the individual signatures. Batch verification requires $n+1$ pairing operations while individual signature verifications require $2n$ pairing operations, where n is the number of signatures. First, the aggregator computes an aggregated signature $\sigma = \sum_{i=1}^{n} \sigma_i$. Then, it verifies the aggregated signature by checking whether $\hat{e}(\sigma, Q) \overset{?}{=} \prod_{i=1}^{n} \hat{e}(h_1(id_i, id_u, id_a, TS, r_iQ), PK_i)$. The proof of this verification is as follows.

$$\begin{aligned}
\hat{e}(\sigma, Q) &= \hat{e}(\sum_{i=1}^{n} \sigma_i, Q) \\
&= \prod_{i=1}^{n} \hat{e}(\sigma_i, Q) \\
&= \prod_{i=1}^{n} \hat{e}(sk_i h_1(id_i, id_u, id_a, TS, r_iQ), Q) \\
&= \prod_{i=1}^{n} \hat{e}(h_1(id_i, id_u, id_a, TS, r_iQ), sk_iQ) \\
&= \prod_{i=1}^{n} \hat{e}(h_1(id_i, id_u, id_a, TS, r_iQ), PK_i)
\end{aligned}$$

4) Aggregated Bid Calculation and Verification at Utility: After receiving the community power response packet, the utility first verifies the aggregated signature σ. Then, it uses the storage units' r_iQs to calculate a shared key with each unit as follows: $h_2(\hat{e}(PK_i, sk_u r_u r_i Q))$. The proof that this key is similar to the one computed by the storage unit ($k_i = h_2(\hat{e}(PK_u,\ sk_i r_i r_u Q))$) is as follows:-

$$\begin{aligned}
h_2(\hat{e}(PK_u, sk_i r_i r_u Q)) &= h_2(\hat{e}(sk_uQ, sk_i r_i r_u Q)) \\
&= h_2(\hat{e}(sk_iQ, sk_u r_i r_u Q)) \\
&= h_2(\hat{e}(PK_i, sk_u r_u r_i Q))
\end{aligned}$$

Using these keys, the utility company can calculate the aggregated bid from the masked bid. The total power that can be injected from the community in each time slot can be calculated as follows: $B - \sum_{i=1}^{n} k_iQ = \sum_{i=1}^{n} (b_i + k_i)Q - \sum_{i=1}^{n} k_iQ = \sum_{i=1}^{n} b_iQ + \sum_{i=1}^{n} k_iQ - \sum_{i=1}^{n} k_iQ = \sum_{i=1}^{n} b_iQ$.

Then, using $\sum_{i=1}^{n} b_iQ$, the utility computes $\sum_{i=1}^{n} b_i$ by exhaustively computing jQ at different values of j until $jQ = \sum_{i=1}^{n} b_iQ$. The other way to compute $\sum_{i=1}^{n} b_i$ is by pre-computing jQ for different values of j and searching the table to find $\sum_{i=1}^{n} b_i$.

In order to verify that the bids are sent from the intended storage units and they have not been modified in transition, the utility company verifies whether $\frac{Mac(B)}{\hat{e}(Q,Q)^{\sum_{i=1}^{n} h_2(k_i)}} \overset{?}{=} \hat{e}(Q, Q)^{\sum_{i=1}^{n} b_i}$. The proof of this is as follows:-

$$\begin{aligned}
Mac(B) &= \prod_{i=1}^{n} Mac(B_i) \\
&= \prod_{i=1}^{n} \hat{e}(Q, (b_i + h_2(k_i))Q) \\
&= \prod_{i=1}^{n} \hat{e}(Q, Q)^{(b_i + h_2(k_i))} \\
&= \hat{e}(Q, Q)^{\sum_{i=1}^{n}(b_i + h_2(k_i))} \\
&= \hat{e}(Q, Q)^{\sum_{i=1}^{n} b_i} \hat{e}(Q, Q)^{\sum_{i=1}^{n} h_2(k_i)}
\end{aligned}$$

IV. Evaluations

In this section, we first investigate the security and privacy preservation of the proposed scheme against well-known attacks. Then, we evaluate the communication and computation overhead.

A. Security and Privacy Preservation Analysis

Our scheme uses two well-known mathematical difficulties, called discrete logarithm and decisional Diffie-Hellman problems. In discrete logarithm problem, if Q and sk_iQ are two points $\in G_a$, it is infeasible to compute x such that $sk_iQ = xQ$. In decisional Diffie-Hellman problem, given Q, r_iQ, and r_uQ, it is infeasible to compute $r_i r_u Q$.

To preserve the privacy of the bids, each storage unit masks its bid with a one-time secret key. Given $(b_i + k_i)Q$ and Q, it is infeasible to calculate the bid b_i without knowing k_i. Eavesdroppers and the aggregator cannot figure out the storage unit's bids because they do not know the secret keys. The use of one-time key to mask the bids can boost the privacy preservation. If the same key is used in two bids, the attacker can calculate the difference between the two bids. That means if one bid is exposed, all the bids that use the same key can also be exposed. Specifically, given the two bids sent at two different times $B_1 = (b_1 + k_1)Q$ and $B_1^{'} = (b_1^{'} + k_1)Q$, the attacker can calculate $B_1 - B_1^{'} = (b_1 - b_1^{'})Q$. Using this value, the attacker can calculate $b_1 - b_1^{'}$. In addition, if a storage unit submits the same bid at different occasions, the packets look different and the attacker cannot know whether the bids are similar, less, or even greater. This is because the bids are masked with random one-time key that has a random r_i used for only one time, i.e., each time the utility requests power injection, the storage units calculate different keys.

The utility can only know the community's total power injection but it cannot know the individual storage unit's bids because it receives only aggregated bids and it is infeasible to decompose them to the individual bids. The aggregator and the utility cannot also know if a storage unit will inject power or not because the units that do not inject power sends packets with zero bid. They do not even know the number of storage units that will inject power. Hiding this information is important to prevent them from drawing a conclusion about the future power availability in the community and thus predicting the future prices. If a PEV is not parked, the meter that measures the power injection sends zero bids so that no one can know whether the PEV is at home or not. The aggregator cannot even know the total amount of power injected from the community. If the attackers can know the scheduled slots for power injection from a PEV, they can know the location of the PEV during these times. In this sense, our scheme does not only aim to protect the power injection data privacy but also location privacy.

In our scheme, the utility can ensure the integrity of the aggregated bid without accessing the individual bids to preserve privacy. The verification of $Mac(B)$ fails if the bids are manipulated or are not sent from the intended storage units. Moreover, if a malicious storage unit tries to use its credentials to send packets in a different community, the aggregator and the utility can identify the attack because they record the identities and certificates of the storage units in the community. In addition, we have used a key management procedure to enable the utility to share keys with the storage units. Computing the shared key between a storage unit and the utility is infeasible because it requires retrieving r_i from r_iQ. Even if the aggregator and some storage units collude, they cannot reveal the shared key between a victim storage unit and utility. The key computation is not controlled only by one side, but the two sides that execute the key agreement procedure contribute to the key.

B. Communication and Computation Overhead

The communication overhead is measured by the amount of data (in Bytes) that should be transmitted. We assign two byte for each identity, half byte for each price (P_i), five bytes for TS, 160 bits for q, and 40 bytes for each group point. Using these, the size of the *Power_Req* packet sent from the utility is $89 + 0.5t$ bytes, where t is the number of time slots. The size of the *Power_Rep* packet sent from the storage units is 171 bytes. The size of the *Power_Rep* packet sent from the aggregator is $129 + 42n$ bytes, where n is the number of storage units. Instead of sending n signatures with $40n$ bytes, the aggregated signature needs only 40 bytes for any number of storage units. Similarly, the sizes of B and $Mac(B)$ are the same for any number of storage units. The size of b_i that can prevent a carry problem is $t \times log_2(16 \times n)$, assuming 4 bits are assigned for the amount of power injection in each slot and n is the number of storage units.

TABLE I: Communication and computation overhead.

Entity	Operation	Computational time (ms)
Utility	*Power_Req* composition	2.5 ms
	Power_Rep verification	$12.72n + 7.27$
Aggregator	*Power_Req* verification	10.35
	*Power_Rep*s verifications	$5.11(n + 1)$
	Aggregated *Power_Rep* composition	$1.41n - 0.9$
Storage units	*Power_Req* verification	10.35
	Power_Rep composition	17.82

The computation overhead is the required time to compose a packet or perform an operation in milliseconds (ms). We will focus on measuring the time required for performing the cryptographic operations because they are usually the main source of the packet delay. Our scheme uses batch signature verification to reduce the computation overhead. Verifying n individual signatures sent from the storage units needs $2n$ bilinear pairing operations but the batch verification needs only $n + 1$ operations.

Experiments were conducted on a computing machine with 2.9 GHz processor and 2 GB Ram with PbC and MIRACL libraries [10], [11]. The measurements indicate that a pairing, r_iQ, point addition, point multiplication, and hashing operations require 5.11 ms, 1.2 ms, 0.45 ms, 0.51 ms, 0.0082 $ms/Byte$, respectively. Using these measurements, Table I gives the computation overhead of our scheme.

As indicated in Fig. 2, in order to compose the *Power_req* packet, the utility company should compute σ_u and r_uQ, that take around 2.5 ms. The aggregator and the storage units need to verify the signature. This needs two pairing operations and one hashing operation, which take 10.35 ms. To compose the *Power_Rep* packet, the storage units need to compute k_i, r_iQ, σ_i, B_i, and $Mac(B_i)$. The total computation time is 17.82 ms. The aggregator needs to verify the signatures that need $n + 1$ pairing operations. It also needs to aggregate the bids and $Mac(B_i)$ to produce B and $Mac(B)$. To total required time to compose the *Power_Rep* packet is $1.41n - 0.9$ ms. The aggregator needs to verify the aggregated signature and $Mac(B)$. It also needs to calculate the keys shared with the storage units. The total required computational time is $12.72n + 7.27$ ms. It can be concluded from Tables I that the computation overhead on the storage units is much less than those of the aggregator or utility. This is important because the computational power of the storage units is less than those of the aggregator and the utility.

V. Related Work

Implementing PEVs in a large scale without control increases peak load significantly and the grid may be overloaded. The increase in electric demand creates new peaks in load curve, decreases the lifespan of transformer and will have power losses and voltage deviation. In [12], Rahman et. al. pointed out that the impact of PEVs charging on the power demand is determined not only by the number of PEVs in use, but also by the number of PEVs being charged at an instant. They conclude that it is not adequate to have only sufficient generation capacity during off-peak hours to assure a system's ability to absorb PEV loads without adverse effects. In [13], two algorithms to address this problem are proposed. Both are based on a forecast of future electricity prices and use dynamic programming to find the economically optimal solution for the PEV owner. Gan et al [14] propose a decentralized algorithm to optimally schedule PEVs charging. Obviously, power injection can be used to help the utility to tackle this problem. Storage units can charge at low-demand periods and inject power at high-demand periods.

Since PEVs will have a significant impact on the power grid due to the increased electricity consumption, intelligent scheduling for charging and discharging of PEVs is required. He et al [15] study two major challenges in the scheduling problem; 1) finding the globally optimal scheduling solution which can minimize the total cost; and 2) finding a distributed scheduling scheme which can handle a large population and the random arrivals of the PEVs. They propose a globally optimal scheduling scheme and a locally optimal scheduling scheme for PEV charging and discharging. Wu et al. [16] propose an operating framework for aggregators of PEVs. A minimum-cost load scheduling algorithm is designed to determine the purchase of energy in the day-ahead market based on the forecast electricity price and PEVs power demands.

However, although the smart grid has received extensive attention recently (see [17], [18], [19], [20]), to the best of our knowledge, a secure and privacy-preserving communication scheme for power injection has not investigated yet.

VI. Conclusion And Future Work

In this paper, we motivated the problem of privacy exposure during the power injections operations of storage unit owners to the utility. We presented a privacy-preserving scheme that is based on the idea of aggregation at the aggregator to hide the individual information of storage owners from the utility. The proposed scheme not only hides the storage information from the utility and aggregator, but it also enables authentication of the storage units and the integrity of their data. We evaluated the proposed schemes performance and security features and showed that the scheme can resist possible attacks and preserve storage owners' privacy without a major computational burden. In our future work, we will extend our scheme to deal with autonomous schedule charging that does not involve infrastructure such as the aggregator. This can be used in smart community with self power managing by mutually helping each other.

References

[1] K. Akkaya, K. Rabieh, M. Mahmoud, and S. Tonyali, "Efficient generation and distribution of CRLs for IEEE 802.11s-based smart grid AMI networks," *Proc. of IEEE SmartGridComm*, Italy, November, 2014.

[2] M. Mahmoud, K. Akkaya, K. Rabieh, and S. Tonyali, "An efficient certificate revocation scheme for large-scale AMI networks," *Proc. of IEEE Performance Computing and Communications Conference (IPCCC)*, pp. 1–8, USA, 2014.

[3] M. Liserre, T. Sauter, and J. Y. Hung, "Future energy systems: Integrating renewable energy sources into the smart power grid through industrial electronics," *IEEE Industrial Electronics Magazine*, pp. 18–37, 2010.

[4] H. Lund and W. Kempton, "Integration of renewable energy into the transport and electricity sectors through V2G," *Energy policy*, vol. 36, no. 9, pp. 3578–3587, 2008.

[5] N. Saputro, K. Akkaya, and S. Uludag, "A survey of routing protocols for smart grid communications," *Computer Networks*, vol. 56, no. 11, pp. 2742–2771, 2012.

[6] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *Advances in Cryptology CRYPTO*, pp. 213–229, 2001.

[7] M. Mahmoud, J. Misic, K. Akkaya, X. Shen, "Investigating public-key certificate revocation in smart grid," *IEEE Journal on Internet of Things (IoT), to appear*.

[8] K. Akkaya, K. Rabieh, M. Mahmoud, and Samet Tonyali, "Customized certificate revocation lists for IEEE 802.11s-based smart grid AMI networks," *IEEE Transactions on smart grid, to appear*.

[9] R. Lu, X. Liang, X. Li, X. Lin, and X. Shen, "Eppa: An efficient and privacy-preserving aggregation scheme for secure smart grid communications," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1621–1631, Sept 2012.

[10] "MIRACL Crypto," https://github.com/CertiVox/MIRACL, accessed: 2015-05-09.

[11] "B. Lynn, PBC library," https://crypto.stanford.edu/pbc/download.html, accessed: 2015-05-09.

[12] S. Rahman and G. Shrestha, "An investigation into the impact of electric vehicle load on the electric utility distribution system," *IEEE Transactions on power delivery*, pp. 591–597, Apr 1993.

[13] N. Rotering and M. Ilic, "Optimal charge control of plug-in hybrid electric vehicles in deregulated electricity markets," *Power Systems, IEEE Transactions on*, vol. 26, no. 3, pp. 1021–1029, Aug 2011.

[14] L. Gan, U. Topcu, and S. Low, "Optimal decentralized protocol for electric vehicle charging," *Power Systems, IEEE Transactions on*, vol. 28, no. 2, pp. 940–951, May 2013.

[15] Y. He, B. Venkatesh, H. V. Poor, and L. Guan, "Optimal scheduling for charging and discharging of electric vehicles," *IEEE TRANSACTIONS ON SMART GRID*, pp. 1045–1105, 2012.

[16] D. Wu, D. C. Aliprantis, and L. Ying, "Load scheduling and dispatch for aggregators of plug-in electric vehicles," *IEEE TRANSACTIONS ON SMART GRID*, pp. 368 – 376, Mar 2012.

[17] K. Rabieh, M. Mahmoud, K. Akkaya, S. Tonyali, "Scalable certificate revocation schemes for smart grid AMI networks using bloom filters," *IEEE transactions on dependence and secure computing (IEEE TDSC), to appear*.

[18] A. Beussink, K. Akkaya, I. Senturk, and M. Mahmoud, "Preserving consumer privacy on IEEE 802.11 s-based smart grid AMI networks using data obfuscation," *Proc. of IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 658–663, Canada, 2014.

[19] M. Mahmoud, J. Misic, and X. Shen, "A scalable public key infrastructure for smart grid communications," *Proc. of IEEE Global Communications Conference (GLOBECOM)*, pp. 784–789, USA, 2013.

[20] ——, "Efficient public-key certificate revocation schemes for smart grid," *Proc. of IEEE Global Communications Conference (GLOBECOM)*, pp. 778–783, USA, 2013.