This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2938875, IEEE Transactions on Information Forensics and Security

SecGrid: A Secure and Efficient SGX-enabled Smart Grid System with Rich Functionalities

Shaohua Li, Kaiping Xue, *Senior Member, IEEE,* David S.L. Wei, *Senior Member, IEEE,* Hao Yue, *Member, IEEE,* Nenghai Yu and Peilin Hong

***Abstract*—Smart grid adopts two-way communication and rich functionalities to gain a positive impact on the sustainability and efficiency of power usage, but on the other hand, also poses serious challenges to customers' privacy. Existing solutions in smart grid usually use cryptographic tools, such as homomorphic encryption, to protect individual privacy, which, however, can only support limited and simple functionalities. Moreover, the resource-constrained smart meters need to perform heavy asymmetric cryptography in these solutions, and thus unnecessarily increases load on smart grid. In this paper, we present a practical and secure SGX-enabled smart grid system, named SecGrid. Our system leverages trusted hardware SGX to ensure that grid utilities can efficiently execute rich functionalities on customers' private data, while guaranteeing their privacy. With our well-devised security protocols in SecGrid, only the smart meters need to perform AES encryption. To validate the superiority of our design, we conduct security analysis and experimentation. Security analysis shows that SecGrid can thwart various attacks from malicious adversaries, and the experimental results show that SecGrid is much faster than the existing privacy-preserving schemes in smart grid.**

***Index Terms*—Smart Grid, Intel SGX, Data Aggregation, Rich Functionalities, Security, Privacy.**

I. Introduction

SMART grid integrates various information and communication technologies to achieve efficient and reliable power generation, transmission, distribution, and control [1]–[3]. Each house will be equipped with a smart meter, which collects customers' interval data (typically minute-level or second-level power usage profile) for billing or analyzing purpose. On the one hand, these fine-grained data are used to enable real time analysis, such as dynamic pricing [4], [5] and load forecasting [6], [7]. On the other hand, this information raises privacy concerns because it reveals important personal information and can lead to various cyberattacks [8], [9]. For example, attackers can derive the appliance usage patterns of the householders from fine-grained energy usage profile [10].

To prevent customers' fine-grained data from disclosure, secure data aggregation schemes [11]–[13] have been proposed to aggregate overall power usage data. In these schemes, each smart meter encrypts data using homomorphic cryptography, such as Paillier [14] and BGN [15], and then reports ciphertext to gateway. The gateway will compute the aggregation result on ciphertext and then report the result to control center for further analysis. Data aggregation guarantees that only overall power usage data will be known by others, thereby protecting customers' privacy. However, some important tools, such as dynamic pricing and load forecasting, which can be used to ensure grid system's stability and reliability, require the grid utilities to compute on customers' fine-grained data. Dynamic pricing is used to charge customers with dynamic prices based on their real time usage. To realize it in a privacy-preserving way, the existing schemes, like the one in [4], use homomorphic encryption and sophisticated design. SecureCloud project [16] explored sensitive data processing including load forecasting with Intel SGX. This work partially interacts with ours in the section of function implementations. However, it is limited in single task while we aim at a general framework.

Although utilizing homomorphic encryption can realize data aggregation and dynamic pricing in a privacy-preserving way, it brings in a heavy computation overhead to the grid utilities, especially for resource-constrained smart meters [17], [18]. In addition, many useful tools, like load forecasting, are less likely to be implemented efficiently with privacy protection in the same way as they always involve computations of high complexity. Furthermore, if we want to realize multiple tools in one system, the computation overhead will be even higher. All these factors make crypto-based schemes impractical to modern smart grid systems.

We refer to these tools (i.e., data aggregation, dynamic pricing, load forecasting, etc.) as functionalities. In general, it is a trade-off between rich functionalities and strong privacy protection, as it is very hard to achieve both of the features simultaneously. However, in this paper, we accomplish both by our novel design. We present SecGrid, a secure and efficient smart grid system that possesses the properties of privacy preservation and rich functionalities. Our security model considers malicious adversaries who may control the software and even the OS of the whole grid utilities (including gateways and control center) except for the certified physical processors involved in the computation. In SecGrid, the resource-constrained smart meters only need to perform AES encryption, and the gateways can perform rich functionalities with high efficiency in a privacy-preserving way. In fact, our system can be treated as a framework, as any functionality that can be implemented obliviously is compatible with SecGrid.

Our main contribution is the design, implementation, and evaluation of this practical smart grid system. We use SGX

S. Li, K. Xue, N. Yu and P. Hong are with Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China (Email: kpxue@ustc.edu.cn).

D. Wei is with the Computer and Information Science Department, Fordham University, New York, NY 10458, USA.

H. Yue is with Department of Computer Science, San Francisco State University, San Francisco, CA 94132, USA.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2938875, IEEE Transactions on Information Forensics and Security

processor, which is Intel's trusted hardware capability [19], as a building block. Indeed, SGX does not guarantee to secure everything, and we need to cope with many challenges not addressed by the hardware. The first is to establish a secret key between a smart meter and a gateway. Since smart meter is resource-constrained, it cannot perform heavy cryptographic schemes, such as Diffie-Hellman key exchange and asymmetric encryption [20], [21]. To solve this problem, a user device is introduced to participate in the initialization phase of a smart meter, which can only interact once with smart meter to complete the key exchange.

The second challenge is to guarantee data integrity for the smart meters' reports. Since SGX has no non-volatile storage, the customers' reports that need to be stored in the storage of gateway, may be tampered, removed, or rolled back by a malicious software or compromised OS [22], [23]. The existing solutions guaranteeing integrity in such case either bring in heavy overhead or are not suitable for smart grid architecture [24], [25]. We thus propose a lightweight integrity guaranteed method for SecGrid. This method is inspired by *count increment* technique of literature [26]. During the processing of gateway, every report will be encrypted together with unique *count* and *nonce*, and the monotonicity and freshness of which will be verified in our proposed periodic report protocol. A similar approach to prevent rollback attacks has also been adopted in Trusted Platform Module (TPM) [27].

The next challenge is to protect the data inside the isolated memory regions from attacks due to unsafe memory accesses. SGX provides the isolated memory regions for the programs, and thus unsafe implementation of programs can easily leak data or suffer from other attacks. By "unsafe" here means that the implemented codes may have memory access patterns or control flows that depend on the values of sensitive data. We thus provide the safe implementations of three functionalities, namely, data aggregation, dynamic pricing, and load forecasting, to show how the functionalities can be securely supported in SecGrid. Other challenges, such as time synchronization, gateway restart protection, etc., are also solved in our system. In summary we make the following contributions:

- We present SecGrid, a practical smart grid system supporting rich functionalities while guaranteeing customers' privacy. In our design, the smart meters only need to perform AES encryption instead of heavy cryptography.
- Our system is compatible with any functions that can be implemented obliviously in smart grid. To better present our system, we discuss three case studies, namely data aggregation, dynamic pricing, and load forecasting, with strong data obliviousness.
- The security analysis indicates that our design is secure against malicious adversaries. Also, the experimental results show that the proposed protocols can be completed efficiently, and the runtime of three functions has around $10^3\times$ improvement compared with existing solutions.

The rest of the paper is organized as follows: Section II enumerates the related works of Intel SGX and rich functionalities in smart grid. Then we present some preliminaries of our system in Section III. In Section IV, we introduce our system model and security model. We illustrate protocols in detail about initialization, periodic report and gateways/control center restart in Section V, followed by implementation of functions in Section VI. In Section VII and VIII, we analyze the security of our design and evaluate the performance respectively. Finally, Section IX makes a conclusion.

II. Related Work

A. Functionalities for Smart Grid

To enable rich functionalities in smart grid, customers' consumption data need to be collected for analysis. However, customers' privacy may be leaked out unconsciously during the procedures. To guarantee privacy and also enable functions carried out in smart grid at the same time, many cryptography-driven schemes have been proposed. One popular privacy-preserving mechanism is secure data aggregation [11]–[13], which aggregates customers' consumption data of a specific region through homomorphic encryption. Another widely researched topic is dynamic pricing [28]–[30], which uses flexible pricing strategy based on current market demands. However, privacy protection schemes, like the one in [4], can only handle simple pricing models with cryptography tools. Some real time pricing models [5] are not likely to realize efficiently in a privacy preserving way due to time or resource limitation. Other famous functions, such as load forecasting [6], [31], [32], that are very useful to improve the grid's performance, also suffer the same problem. SecureCloud project [16] showed several use cases such as data aggregation and load forecasting to show the possibility of implementing sensitive data processing with Intel SGX in smart grid systems.

B. Intel SGX

Intel SGX provides *isolated* execution spaces, named enclaves. Programs in enclaves can process data in plaintext. But any software or even the OS on the same platform, cannot observe the data content inside enclaves [33], [34]. There are also many works aiming at developing privacy-enhanced applications with SGX. Ohrimenko et al. [33] showed how to outsource model training to untrusted servers. To make memory accesses data-independent, which is not protected by SGX, their system uses padding and other tricks to hide access pattern. VC3 [34] implements MapReduce in distributed servers with confidentiality and verifiability. Opaque [35] is an encrypted data analytics platform over Spark SQL, which uses oblivious sorting for data processing in an encrypted database. Iron [36] implements some interesting but heavy primitives in cryptography. The SGX implementations are efficient and practical. The access pattern, not protected by SGX, is hidden by oblivious comparison functions. Town Crier [37] is a work that provides authenticated data feed from external trusted sources for smart contracts. ZeroTrace [24] provides an oblivious data storage system from SGX to access external storage, which minimizes the response time. Obliviate [38] provides an SGX-based file system for oblivious data storage and access. EnclaveDB [39] is a secure database using SGX, which guarantees integrity, confidentiality, and freshness for data and queries.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2938875, IEEE Transactions on Information Forensics and Security

In aforementioned systems, authors have proposed effective designs including new system architectures, protocols and etc., to address possible security issues that may occur when deploying SGX. In SecGrid, we also develop protocols and study use cases to address both security and performance issues in smart grid systems.

III. Preliminaries

A. Enclaves in SGX

SGX refers to Intel Software Guard Extension, a set of CPU extensions, which can provide isolated execution environments, named **enclaves**, to protect the confidentiality and integrity of the data against all other software, even a compromised OS, on the platform. When a platform is equipped with a SGX-enabled CPU (such as the gateways and control center in our system), in addition to the enclave, the memory, BIOS, I/O and even power are treated as potentially untrusted. The general processing flow in enclave is shown in Fig. 1. The encrypted data firstly will be transmitted into enclave for decryption. Then the decrypted data will be the input of some function f. Finally, the output of f will be encrypted and then sent to the outside of the enclave.

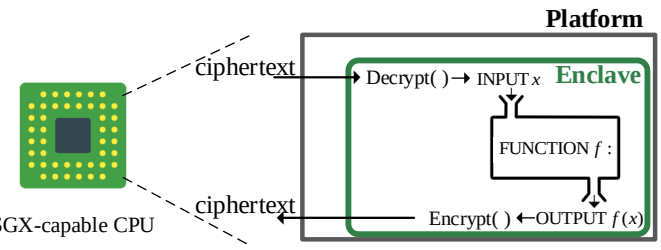


Fig. 1. General processing flow in enclave

SGX provides two core operations, *sealing* and *remote attestation*, which will be used in our system. *Sealing* is for storing data securely outside of the enclave. *Remote attestation* is for a remote party to verify the legitimacy of the enclave (i.e., the enclave is created by a legal SGX-capable CPU and the code is correctly loaded in the enclave). The details are as follows:

- *Sealing.* Each SGX-capable CPU has a hardware-protected sealed key called *Root Seal Key* that cannot be stolen or forged. An enclave can derive a *Seal Key* from the *Root Seal Key* using instruction `EGETKEY`. This key is specific to the enclave, and other enclaves cannot derive the same key. But the same enclave can always get this key even if it is destroyed and restarted. *Seal Key* is used to encrypt and authenticate data stored outside of the enclave.
- *Remote attestation.* SGX allows a remote party to check whether the code is correctly loaded in an enclave. When an enclave is created, the CPU will generate a hash of the state of the loaded code and static data, known as *measurement*, and a *report* that contains the *measurement* and optional self-defined data (e.g. a new generated public key). Then the software that created the enclave can ask for a *quote*, which consists of a *report* and its signature signed with a hardware-protected attestation key. Remote parties can verify the *quote* by contacting the Intel Attestation Server. Such procedure is known as ***remote attestation***. The detailed operations can be found in [19].

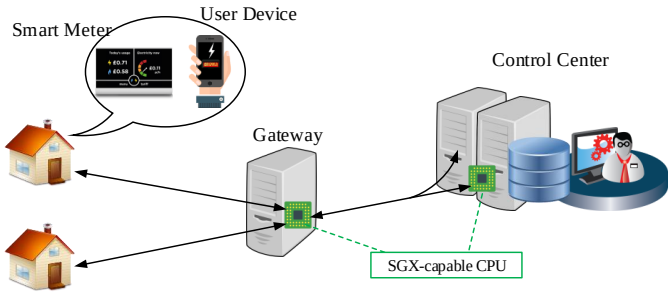


Fig. 2. System Architecture

B. Attacks against Enclave

The protection of SGX is restricted in CPU. Although data are encrypted, the memory access patterns may leak the privacy of data inside enclave [24]. Branches in program like `if-else` make data-dependent running patterns, and enclaves that are running such programs suffer from cache-timing attack [33], [34]. Access to storage outside enclave exposes address to a PCI-e bus listener or an operating system (OS), who can create page faults. Furthermore, if the address is data-dependent, there will be page-fault attacks [23], [24].

Also, rollback attack breaks the freshness of external data [25]. Some data that requires a long-term preservation should reside in persistent storage, like disk. These data can be rolled back to a previous version by a compromised OS. For example, replacing the reported power usage data at time slot t with $t-1$. In preventing rollback attack, integrity guarantee from Merkle tree [24] fails when the platform is restarted, while other methods are either too heavy or unable to be applied to smart grid architecture [25], [34]. Therefore, we need to develop new approach in our design to prevent such attack.

IV. System Model and Security Model

A. System Model

Our system adopts the typical architecture of smart grid, which is shown in Fig. 2. It contains a control center, gateways in residential area, smart meters, and user devices in home area. Control center and gateways are able to create enclaves, called the *control enclave* and the *gateway enclave*, respectively. Our design goal is to protect individual user's privacy, i.e., the reports generated by smart meters and intermediate results during processing. The final outputs of some functionalities may leave enclaves. In such case, we need to guarantee the outputs will not reveal individual user's private data. Thanks to SGX's design, all running or to be run codes inside enclaves can be verified by anyone. Thus any users could do the verification to guarantee their privacy is not leaked.

Control Center (CC). CC collects data and responds to requests from/to each gateway. It has a enclave, called *control*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2938875, IEEE Transactions on Information Forensics and Security

enclave (CE). CC also has all the initialization keys of smart meters ($\mathcal{K}^{\mathtt{i}}_{\mathsf{init}}$ for each smart meter i). These keys are used in the initialization phase of smart meters, and can be accessed by the *control enclave*. To prevent the CC or other potential attackers from knowing the smart meter's key, the *control enclave* takes Merkle tree based techniques to access these keys.

Gateways (GW). A GW runs a secure enclave, called *gateway enclave* (GE), which directly collects and processes the data reported by smart meters. Many functionalities, such as data aggregation, dynamic pricing, and load forecasting, can be performed inside the *gateway enclave*. The *gateway enclave* can establish shared keys with smart meters with the help of user device.

Smart Meters (SM). Every house is equipped with a SM to collect the power usage data and report the encrypted data to *gateway enclave* periodically, e.g., every 15 minutes. Considering the constrained resource of smart meter, the only cipher used here is AES-GCM used to guarantee both confidentiality and integrity of data. Each SM contains an initialization key $\mathcal{K}^{\mathtt{i}}_{\mathsf{init}}$ that is used to establish a new secret key $\mathcal{K}_{\mathtt{i}}$ between the SM and the *gateway enclave*.

User Devices (UD). A user needs a device to help his/her SM establish a secret key with *gateway enclave* in the initialization phase. The device can be a smartphone that is installed with an official application so that it can participate in the initialization of a new SM. UD has sufficient computing capability to run asymmetric cryptography algorithms and verify remote attestation.

B. Security Model

We assume that a malicious adversary who can control all the software, including the OS, in the CC and GW, tries to violate the *confidentiality* and *integrity* of customers' private data by performing the following attacks. The adversary can read, block, modify, and replay all messages sent by/to a secure enclave. The adversary is also able to observe memory access pattern and infer control flow in an enclave process, i.e., launching side-channel attacks. In particular, the adversary may perform rollback attack, that is, to replace the sealed data with a previous version. In real smart grid scenarios, CC is often considered as semi-trusted, i.e., they follow the pre-defined protocols while trying to learn as much users' privacy as possible. In SecGrid, such assumption can also be used. But thanks to SGX, SecGrid can work well under malicious model. We thus assume CC is malicious or may be compromised by attacker in this paper. SMs and UDs are assumed to be trusted in SecGrid. Although there exist practical attacks to break smart meters, there have been many defense methods [40], [41] proposed to defend against them, which are compatible with SecGrid and can be directly utilized.

We assume that the adversary cannot compromise the secure enclaves and the relevant enclave keys (e.g. $SealKey$ and attestation key). The adversary cannot break cryptographic primitives used in our system, i.e AES-GCM, Diffie-Hellman key exchange, etc. Compromising the user device, denial-of-service and physical attacks, such as power analysis, are out of the scope of this paper.

The adversary is assumed to not want to trigger alarm. Although the adversary can perform various attacks, he/she does not want to be detected by the smart grid system.

Besides, the grid administrator is responsible for the secure deployment of smart meters. Their keys are sealed by the *control enclaves* in advance.

V. System Design

A. Overview

Our goal is to guarantee customers' privacy while enabling rich functionalities in smart grid. Rich functionalities refer to various demand side management functionalities that process customers' private data to improve the grid's performance. These rich functionalities are hard to realized efficiently by cryptography-based schemes.

In SecGrid, a GW runs rich functionalities inside the *gateway enclave*. To prevent side-channel attacks, these functionalities should not contain data-dependent operations, that is, they need to be implemented obliviously. We discuss the possible leakage that may be introduced by three popular functionalities, namely, data aggregation, dynamic pricing, and load forecasting, and provide the secure implementations, which will be described in Section VI.

To guarantee the confidentiality and integrity of input and output data of rich functionalities, in SecGrid, we develop a new periodic report protocol to secure the transmission and storage of customers' private data. Each SM's reported data is encrypted using AES-GCM, and contains two new parameters, `nonce` and `ctr`. These parameters are carefully used to resist various attacks, such as replay attack and rollback attack, which, however, require more complex measures to prevent in other solutions. The secret key used in this protocol is established by our proposed initialization protocols for SMs and GWs. Considering the robustness of our system and preventing the adversaries from restarting the *gateway enclave*, we propose a status restoring protocol for GW, which can avoid data loss due to the restart of GW or *gateway enclave*.

B. CC/GW Initialization

This initialization phase involves the *control enclave* and *gateway enclave*. The *control enclave* will be initialized at the system setup, and when the *gateway enclave* starts to work, it will interact with the *control enclave* to authenticate each other as well as share symmetric/asymmetric keys and time information.

The initialization protocol is shown in Fig. 3. These six steps can be divided into two stages: **Attest and Key Exchange** and **Time Sync**. During the first stage, the *gateway enclave* and the *control enclave* attest each other through *remote attestation*, and then use Diffie-Hellman key exchange to share a secret key. In the second stage, the *control enclave* synchronizes its time to the *gateway enclave* as well as confirms the shared keys. The details are shown as follows.

Stage 1 [Attest and Key Exchange]

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2938875, IEEE Transactions on Information Forensics and Security

① The *gateway enclave* generates a public/private key pair ($\mathcal{PK}_{\mathsf{gw}}$, $\mathcal{SK}_{\mathsf{gw}}$) for a CCA2-secure public key cryptosystem.
② The *gateway enclave* sends its *remote attestation* message to the *control enclave*, which contains $\mathcal{PK}_{\mathsf{gw}}$ and a generated Diffie-Hellman parameter g^a.
③ Upon receiving the *remote attestation*, the *control enclave* verifies its legitimacy (the detailed verification phase of *remote attestation* is described in Section III-A). Then, it generates another Diffie-Hellman parameter g^b, and sends back its *remote attestation* message containing $\mathcal{PK}_{\mathsf{cc}}$, g^b and the current wall-clock time.

Stage 2 [Time Sync. and Confirm]

④ Once the message is received, the *gateway enclave* records the received time as reference time and starts the time counter from 0. After verifying the *remote attestation*, the *gateway enclave* obtains the time: the reference time plus the value of the time counter. Then, it encrypts the time with g^{ab} and signs the ciphertext with $\mathcal{SK}_{\mathsf{gw}}$. The *gateway enclave* sends the ciphertext and the signature to the *control enclave*.
⑤ The *control enclave* verifies the signature with $\mathcal{PK}_{\mathsf{gw}}$, decrypts the ciphertext with g^{ab}, and then compares the time with local time. If all the verifications succeed, the *control enclave* will seal g^{ab}, and return an *ack* message, which is encrypted using g^{ab} and signed using $\mathcal{SK}_{\mathsf{cc}}$.
⑥ Upon receiving the *ack*, the *gateway enclave* verifies the signature using $\mathcal{PK}_{\mathsf{cc}}$ and decrypts the ciphertext using g^{ab}. Then, it seals g^{ab} and $\mathcal{PK}_{\mathsf{cc}}$.

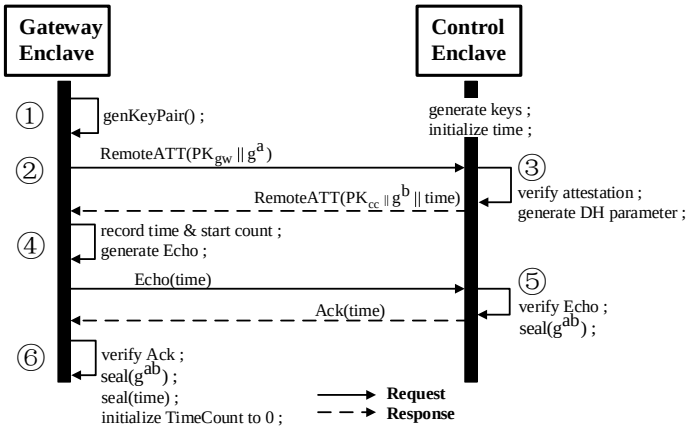


Fig. 3. CC/GW initialization protocol.

One thing to note that the time of the system does not need to be consistent with the Internet, and it just needs to be consistent within the system.

C. SM Initialization Protocol

The first time when a customer accesses the smart grid, he/she first installs an official applications on UD to initialize his/her SM. The SM initialization protocol is as shown in Fig. 4. This protocol can be triggered by the UD or the customer manually. We take the initialization of the *i*-th SM as an example in the following description. The initialization phase

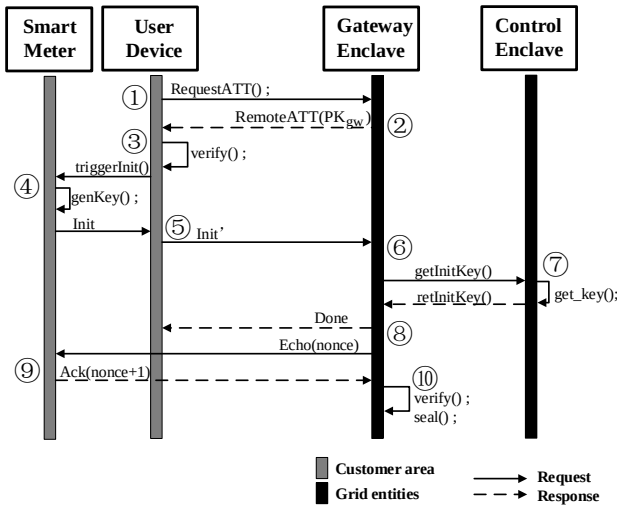


Fig. 4. The SM initialization protocol.

can be divided into three stages: **Attestation**, **Key Establishment**, and **Confirm**. The UD will verify the legitimacy of the *gateway enclave* according to its *remote attestation*. The second stage is used to establish a secret key between the SM and the *gateway enclave*, where the UD is used as a bridge. The SM has a sealed initialization key $\mathcal{K}_{\mathtt{init}}^{\mathtt{i}}$, which is also known by the *control enclave*. The secure deployment of smart meters' keys is the responsibility of the grid administrator. To enable the secure use of $\mathcal{K}_{\mathtt{init}}^{\mathtt{i}}$, we utilize Merkle tree based method, proposed in ZeroTrace [24], to guarantee integrity and freshness. Note that, after the following procedures, a fresh `nonce` will be securely obtained by the SM, and it will be used as one of the parameters in the first report of the SM.

One significant problem is how the SM sends initialization message to the UD. We can utilize existing solutions such as ZigBee based communication protocol (used in OG&E company [21]) to achieve this purpose.

Stage 1 [UD←GE, Attestation]

① UD starts and requests a remote attestation from the *gateway enclave*.
② The *gateway enclave* returns its *remote attestation* message, which contains $\mathcal{PK}_{\mathsf{gw}}$, to the UD.
③ Upon receiving the *remote attestation*, the UD verifies it, and then triggers the initialization phase of the SM.

Stage 2 [SM↔GE, Key Establishment]

④ The SM starts the initialization phase. It first generates a new random key $\mathcal{K}_{\mathtt{i}}$, and then encrypts this key and its identifier $\mathtt{ID_i}$ with sealed initialization key $\mathcal{K}_{\mathtt{init}}^{\mathtt{i}}$. After that, the SM sends the `Init` message to the UD, where $\mathtt{Init} = \mathtt{ID_i}||E_{\mathcal{K}_{\mathtt{init}}^{\mathtt{i}}}(\mathtt{ID_i}||\mathcal{K}_{\mathtt{i}})$, $E(\cdot)$ is a symmetric cryptosystem.
(To be noted, the `Init` message does not need to be protected from eavesdropping since it has been encrypted. We will analyze this in detail in Section VII-D).
⑤ The UD encrypts the $\mathtt{ID_i}$ in `Init` message using $\mathcal{PK}_{\mathsf{gw}}$ to generate new $\mathtt{Init}'$ message, and sends it to the *gateway enclave*: $\mathtt{Init}' = \hat{E}_{\mathcal{PK}_{\mathsf{gw}}}(\mathtt{Init})$, $\hat{E}(\cdot)$ is a public-key cryptosystem.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2938875, IEEE Transactions on Information Forensics and Security

⑥ The *gateway enclave* extracts the $\mathtt{ID_i}$ with $\mathcal{SK}_{\mathsf{gw}}$, generates a *getInitKey()* message containing the $\mathtt{ID_i}$ encrypted with g^{ab}, and then sends it to the *control enclave*.

⑦ Upon receiving the message, the *control enclave* decrypts it and extracts the $\mathtt{ID_i}$. Then, the *control enclave* obtains the corresponding $\mathcal{K}^{\mathtt{i}}_{\mathsf{init}}$, void this key and then returns it after encryption.

⑧ The *gateway enclave* can decrypt the $\mathcal{K}^{\mathtt{i}}_{\mathsf{init}}$, and obtain the $\mathcal{K}_{\mathtt{i}}$ from $\mathtt{Init}'$ with it. The *gateway enclave* returns a *Done* message to the UD to notify the initialization has succeeded, and an *Echo* message to the SM, which contains an encrypted `nonce` using $\mathcal{K}_{\mathtt{i}}$.

Stage 3 [SM↔GE, Confirm]

⑨ The SM decrypts the `Echo` message, and sets the local time to *time*. Then, the SM returns an `Ack` message to the *gateway enclave*. This message contains the encrypted $\mathtt{nonce}+1$.

⑩ The *gateway enclave* verifies the `Ack` message, and then seals the $\mathcal{K}_{\mathtt{i}}$ with $\mathtt{ID_i}$ as associated data in AES-GCM.

D. Periodic Report Protocol

When the initialization phase is done, the SM shares a symmetric key with the *gateway enclave*, which will be used to secure the report data. In order to ensure the data integrity and prevent replay attack, we enable a monotonic counter `ctr`, which starts from 0, in SM to indicate different reports. Due to the continuity of the SM's reports, our use of `ctr` can resist rollback attack, which will be proved in our security analysis. Another parameter `nonce` is also used here to guarantee the freshness of reports. Considering that the freshness verification needs the *gateway enclave* to store every `nonce`, which is difficult since SGX has no permanent storage, we make clever use of cyclical nature of the reports. By letting the *gateway enclave* randomly choose the `nonce` for the SM to use in the next report, we avoid the storage of each `nonce`. The protocol proceeds as follows:

① When the *i*-th SM needs to report, it first increases the counter $\mathtt{ctr_i} = \mathtt{ctr_i} + 1$, and then generate the report $\mathtt{r_i} = \mathtt{ID_i}||E_{\mathcal{K}_{\mathtt{i}}}(\mathtt{ID_i}||\mathtt{m_i}||\mathtt{nonce}||\mathtt{ctr_i})$, where `nonce` is sent by the *gateway enclave* during last report period. The SM reports $\mathtt{r_i}$.

② Upon receiving the report, the *gateway enclave* decrypts the report with its sealed key $\mathcal{K}_{\mathtt{i}}$, and extracts data $\mathtt{m_i}$, `nonce`, and $\mathtt{ctr_i}$. Next, the *gateway enclave* verifies the correctness of `nonce` and obtains last counter $\mathtt{ctr}^{\mathtt{old}}_{\mathtt{i}}$ from storage, and then checks if $\mathtt{ctr_i} = \mathtt{ctr}^{\mathtt{old}}_{\mathtt{i}} + 1$. If all passed, the *gateway enclave* will seal $\mathtt{r_i}$ and process $\mathtt{m_i}$ with predefined functions. Otherwise, an error or attack may happen, the *gateway enclave* will report this alarm to CC immediately. Finally, based on the outputs of functions, the *gateway enclave* generates a report for CC and a response for the SM.

③ The *control enclave* can process these reports in the same way as what the *gateway enclave* does, and generate a response.

In both *control enclave* and *gateway enclaves*, there needs a secure database to store data. Here we can utilize secure database techniques based on SGX, like EnclaveDB [39], to achieve this purpose.

There might be failed reports caused by network error, adversarial attacks, etc. In such scenarios, the corresponding *gateway enclave* can simply ask the failed smart meter to re-report, and trigger an alarm if it fails again.

One key challenge of above protocol is how to design and program the functions executed inside the enclaves to prevent privacy leakage. As we illustrate before, SGX is not perfectly secure. For example, software-based side channel attacks can violate the data confidentiality even if the data is inside the *gateway enclave*. So we should take fully account of the secure design and implementation of these functions. We will discuss this issue in Section VI.

E. GW/CC Restart Protocol

When a *gateway enclave* restarts, it needs to restore its previous state, i.e., security keys, latest reports, and time. At restart Security keys are unsealed securely from local storage to establish secure channels. The *gateway enclave* loses all previously unsealed data and has to request these data from SMs or *control enclave*. The restart protocol is detailed as follows:

① The *gateway enclave* unseals $\mathcal{K}_{\mathtt{i}}$, $\mathcal{SK}_{\mathsf{gw}}$, $\mathcal{PK}_{\mathsf{gw}}$, $\mathcal{PK}_{\mathsf{cc}}$, g^{ab} and all latest sealed reports from storage. If the unsealing procedure fails, the alarm will be triggered. Otherwise, the *gateway enclave* sends newly generated `nonces` and requests to all SMs to ask for report as well as to the *control enclave* to ask for `time`. These new `nonces` will replace the old nonces and be used in new reports.

② **a.** Each SM returns its latest report to the *gateway enclave*.
b. The *control enclave* returns current time to the *gateway enclave*.

③ Upon receiving responses, the *gateway enclave* verifies the time then sets the local time. Then it checks the freshness of `nonces` and if $\mathtt{ctr_i}$ in each report satisfies $\mathtt{ctr_i} = \mathtt{ctr}^{\mathtt{old}}_{\mathtt{i}}$ or $\mathtt{ctr_i} = \mathtt{ctr}^{\mathtt{old}}_{\mathtt{i}} + 1$. If all passed, the *gateway enclave* restores successfully. Otherwise, the restore phase may encounter a problem, and the *gateway enclave* will trigger alarm to notify the grid administrator.

When a *control enclave* restarts, it also needs to restore its previous state. It firstly unseals private key $\mathcal{SK}_{\mathsf{cc}}$ and public keys $\mathcal{PK}_{\mathsf{gw}}$ of each *gateway enclaves*. There may exist reports that have not sealed yet. In such cases, the *control enclave* could ask *gateway enclaves* to report again, just like the restart phase for *gateway enclaves*. To restore time, the *control enclave* should ask at least one *gateway enclave* to report its time, and it thus requires at least one live *gateway enclave*.

VI. Functions

The protocols have secured the data submission from SMs to the *gateway enclave* and the *control enclave*. However, SGX is shown to be vulnerable to several types of attacks, in particular, cache timing and page table side-channel attacks, as well

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2938875, IEEE Transactions on Information Forensics and Security

as speculative attacks [33], [35], [36]. Although Intel SGX excludes side-channel from its security model, many schemes like Varys [42] and SCONE [43] have been proposed to protect programs running in SGX enclaves from side-channel attacks. Grid developers should use or refer to these schemes for secure function implementations. We discuss three distinct case studies where the security of users' data is put at risk due to data-dependent calculations. These functions are chosen because their secure implementations have been well studied for many years. Here we want to show how they could be securely implemented in a simple way in SecGrid. We also want to highlight that all these functions could be deployed simultaneously with SecGrid while other schemes could not be or are strictly limited due to performance reasons.

A. Case Study 1: Data Aggregation

Secure data aggregation is used to aggregate overall power usage of all customers over a timespan as follows:

$$\mathsf{PowerUsage}^{T}_{\mathsf{area}}(t) = \sum_{i \in S} \sum_{\Delta=t}^{t+T-1} \mathsf{PowerUsage}_i(\Delta), \quad (1)$$

where S is the customer set in this area. Compared with cryptography-based schemes [11], [13] that uses Paillier or BGN cryptosystem for homomorphic computation, our system only requires symmetric encrypted data for submission and computation. To make the aggregation oblivious, the *gateway enclave* follows by the order of the arriving reports to add up the power usage data, and thus no additional leakage exists.

B. Case Study 2: Dynamic Pricing

Typical dynamic pricing models include Time-of-Use (ToU), Critical Peak Pricing (CPP), and Real Time Pricing (RTP) [2], [5]. We here introduce the secure implementation of them.

ToU Electricity prices are different at peak time and at off-peak time. Peak time prices are higher than off-peak time for demand control. GW takes the price from a piecewise function:

$$\mathsf{PricePerUnit}(t) = \begin{cases} \mathsf{p} & \text{if } t \in \text{off-peak time} \\ \mathsf{p} + \Delta\mathsf{p} & \text{if } t \in \text{at-peak time} \end{cases}, \quad (2)$$

where t is the current time. The dynamic pricing does not have sensitive patterns, because the condition for the piecewise function (2) is time t, which is open.

But a secure and reliable time [36], [37] needs extra efforts, since timestamp in gateway BIOS can be tampered. As mentioned in Section V-B, *gateway enclaves* can obtain time from *control enclave*, i.e., gets time from a trusted source during initialization.

CPP. Peak time is not fixed. In holidays or the days with special events, it may not be suitable to use ToU pricing model, which is generally for regular days. Therefore, CPP is developed to also handle such case, which can be implemented in a way that is similar to ToU in our system, where days become the condition of specialized piecewise function [2], [4].

RTP. Real time pricing schemes allow the grid to charge customers with the nearest real time price, i.e., the price at each particular interval of time (e.g. one hour). The price can be announced one hour or a day ahead. We implement a day ahead RTP scheme proposed in paper [5] in our system, in which the grid utility releases the predicted prices of the next 24 hours.

Let m_h denote the reported power usage at hour h, and the pricing function which depends on three parameters a_h, b_h, $m_0 \geq 0$ be as follows:

$$\mathsf{RealTimePricing}(m_h) = \begin{cases} a_h, & \text{if } 0 \leq m_h < m_0, \\ b_h, & \text{if } m_h \geq m_0. \end{cases} \quad (3)$$

In this scheme, m_0 is a fixed value, while a_h and b_h change every hour and every day. In order to allow customers to have sufficient time to schedule their electricity consumption, the GW should predict the prices of the next 24 hours (i.e. 24 a_h and b_h) and broadcast the prices to the SMs. Let $\hat{a}[t][h]$ and $\hat{b}[t][h]$ denote the *predicted* parameters for the upcoming price tariff for each hour h on day t, the prediction model is formulated as follows:

$$\begin{aligned} \hat{a}[t][h] &= k_1 a[t-1][h] + k_2 a[t-2][h] + k_3 a[t-7][h], \\ \hat{b}[t][h] &= k_1 b[t-1][h] + k_2 b[t-2][h] + k_3 b[t-7][h]. \end{aligned} \quad (4)$$

Note that $\hat{a}[t][h]$ and $\hat{b}[t][h]$ are just *predicted* parameters. The true values ($a[t][h]$ and $b[t][h]$) will be known when the hour h comes, and be used to charge customers. As the condition for piecewise function is power usage x, the access pattern of a naïve implementation is *data-dependent*. To hide the access pattern, we use oblivious assembly functions `O_greater()` and `O_move()` [33], [36]. The function `Real_Time_Pricing` in Fig. 5 is executed in the *gateway enclave*, which avoids `if-else` branches and has no data-dependent operations. Thus it can thwart aforementioned side-channel attacks.

```
float Real_Time_Pricing(int m, float pa, float pb, int m0){
// pa and pb are the real value of a[t][h] and b[t][h]
      bool flag = O_greater(m, m0);
      //Keep two decimal places of pa and pb
      int a = int(pa * 100), b = int(pb * 100);
      int p = O_move(flag, a, b);
      float bill = (p / 100.0) * m;
      return bill;
}
```

```
O_greater(x, y) :
    mov     rcx, x
    mov     rdx, y
    cmp     rcx, rdx
    setg    al
    retn
```

```
O_move(cond, x, y) :
    mov     rcx, cond
    mov     rdx, x
    mov     rax, y
    test    rcx, rcx
    cmovz   rax, rdx
    retn
```

Fig. 5. Data oblivious real time pricing function.

C. Case study 3: Load Forecasting

Current commonly used load forecasting models include statistical based model and artificial intelligence based model

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2938875, IEEE Transactions on Information Forensics and Security

[2], [44]. Next, we describe the secure implementation of stochastic time series method [6] and neural network based algorithm [44].

Stochastic Time Series. This method uses a fit function to calculate a prediction of next moment (e.g. hourly) load from previous records. An example model is [6]:

$$\begin{aligned}\mathsf{Load}(t) =&\phi_1\mathsf{Load}(t-1)+\phi_2\mathsf{Load}(t-2)\\&+\ldots+\phi_k\mathsf{Load}(t-k)+noise(t).\end{aligned}\tag{5}$$

To implement this model, the cryptography-based methods have to use computationally expensive homomorphic multiplication and addition. While in our system, the GW can compute the predicted load inside the *gateway enclave* with decrypted data directly.

Neural Network Neural network based algorithms can figure out the relationship between referring variables and power consumption by supervised learning [44]. Referring variables may include history consumption, day (e.g. holiday), and weather (e.g. temperature). Similarly, these data items are obtained from the *control enclave*. Compared with pure cryptography schemes that need to leverage homomorphic encryption, we can run oblivious machine learning algorithms [33] on plaintext data in the *gateway enclave*.

D. General Functions

Besides the three functions we described above, there are many other functions often performed by the grid to improve the system performance. Many of them require customers' private data as inputs. All these functions can be denoted as $f(\mathbf{x},\mathbf{y})$, where $\mathbf{x}$ is the privacy-related input, and $\mathbf{y}$ is the remaining input, such as electricity price, time, and weather conditions. In general, $f(\mathbf{x},\mathbf{y})$ can be seen as a combination of simple functions. The three implemented functionalities contain basic mathematic operations, like addition, and general functions, like piecewise function, unary function, multi function, so they provide an implementation reference for $f(\mathbf{x},\mathbf{y})$.

For example, UDP [4] is a usage-based dynamic pricing scheme. For individual electricity usage $e_{i,t}$ of user i and community-wise electricity usage $e_{s,t}$ at time slot t, pricing function $F(e_{i,t})$ is defined as :

$$F(e_{i,t})=\begin{cases}p_1, & \text{if } e_{s,t}<e_m,\\ p_2, & \text{if } e_{s,t}>e_m,\ e_{i,t}\le e_a,\\ a+be_{i,t}+ce_{i,t}^2, & \text{if } e_{s,t}>e_m,\ e_{i,t}>e_a,\end{cases}\tag{6}$$

where $e_{s,t}$ is calculated based on electricity usage of whole community, e_m and e_a are pre-defined parameters. When $e_{i,t}\le e_a$, user i has static price p_1 or p_2. When $e_{i,t}>e_a$, the dynamic price is applied. In this case, $F(e_{i,t})$ is referred as $f(\mathbf{x},\mathbf{y})$, $e_{s,t}$ and $e_{i,t}$ are referred as $\mathbf{x}$, $\{p_1,p_2,a,b,c,e_m,e_a\}$ are referred as $\mathbf{y}$. To perform $f(\mathbf{x},\mathbf{y})$ in a privacy-preserving way, the enclaves in our system collect $\mathbf{x}$ from SMs, and request $\mathbf{y}$ from a trusted data source via HTTPS [37]. For example, the grid administrator can post the latest pricing strategy on a website, where the public can easily verify and the enclaves can obtain the data they need. From this prospective, our SecGrid system is able to support rich functionalities with privacy protection, and this feature is not available in other cryptography-based smart grid systems.

It is worth noting that SGX has a limited working memory in the current design and exceeding memory may have serious impacts on performance. Loading large libraries like machine learning toolsets when running is possible to exceed such memory. There are works like SCONE [43] can be used to reduce required memory of large libraries while maintaining security guarantees. In fact, as shown in [45], compared with homomorphic encryption-based schemes, SecGrid would still be much faster even if the running codes exceed working memory.

VII. Security Analysis

Our system, **SecGrid**, aims to protect customers' privacy and execute functions securely against malicious adversaries. Specifically, our system should guarantee (1) confidentiality, (2) integrity, and (3) availability. And we will describe how the various attacks are prevented by our proposed secure protocols and oblivious operations. In addition, we analyze the security for initialization phases, which are the foundation of our system security.

A. Confidentiality

Theorem 1. *Customer data from the periodical report protocol will never leak outside the enclave. Only the outputs of data aggregation, dynamic pricing, and load forecasting are revealed to the power grid company.*

Proof. Data confidentiality comes from (a) secure communication protocols for periodic report to hide data on the fly; (b) oblivious operations in *gateway enclave* to hide access pattern.

(a) Outside enclave Periodic reports from SM are encrypted with AES-GCM under key $\mathcal{K}_{\mathtt{i}}$ shared between SM and GW enclave. This encryption is IND-CCA2 and any *p.p.t (probabilistic polynomial-time)* adversary cannot break the confidentiality without $\mathcal{K}_{\mathtt{i}}$. For key establishment, SM and the *gateway enclave* initialize key K_i using the protocol of Section V-C. SM generates the key and notifies the gateway in $\mathtt{Init}'$ encrypted with $\mathcal{PK}_{\mathtt{gw}}$. Only the *gateway enclave* can decrypt $\mathtt{Init}'$ and reveal $\mathcal{K}_{\mathtt{i}}$ with CC's keyring. The sealing of data uses *SealKey* to protect confidentiality.

(b) Inside enclave The functions are implemented on the top of oblivious operations of Section VI. The memory and execution patterns are no longer data-dependent.

- *Data Aggregation*. Reports are summed up according to the order of arrival. The aggregation is independent from the electricity consumption data in SM reports.
- *Dynamic Pricing*. ToU and CPP use the piecewise function but the condition is timestamp, which is not private. When RTP is used for leveled price, the condition becomes the usage x which is private. We use oblivious assembly functions [33], [36] `O_greater()` and `O_move()` to make RTP function oblivious.
- *Load Forecasting*. Both regression and neural network systems leverage oblivious learning algorithms [33].

It shows the confidentiality throughout user data lifecycle. □

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2938875, IEEE Transactions on Information Forensics and Security

B. Integrity

Theorem 2. *Integrity and freshness of periodic reports from SM are achieved in the secure communication protocol.*

Proof. AES-GCM $= (\mathcal{K}, \mathcal{E}, \mathcal{D})$ provides *existential unforgeability*. Any *p.p.t.* adversary A should fail to forge a ciphertext:

$$\mathsf{Adv}^{\text{exist}}_{A,(\mathcal{K},\mathcal{E},\mathcal{D})} \stackrel{\text{def}}{=} \mathsf{Pr}\left[sk \leftarrow \mathcal{K}; y \leftarrow A^{\mathcal{E}_{sk}(\cdot)} : \mathcal{D}_{sk}(y) \neq \perp\right] \leq \epsilon,$$

where A should never receive the ciphertext y in return from the encryption oracle $\mathcal{E}_{sk}(\cdot)$. The advantage ϵ is negligible.

The existential unforgeability guarantees the message originates from SM. To prevent rollback attacks, we use monotonic GW counter $\texttt{nonce}$ and SM counters $\texttt{ctr}_\texttt{i}$ to prevent any steal packet of $\texttt{ctr}_\texttt{i}^{\texttt{old}} \leq \texttt{ctr}_\texttt{i}$ to forge a packet for $\texttt{ctr}_\texttt{i}$. If a steal report is replayed, *gateway enclaves* can trigger an alarm. □

Theorem 3. *Integrity (and freshness) of the external database in gateway enclaves and control enclave is guaranteed.*

Proof. Gateway enclaves seal K_i and $\texttt{ctr}_\texttt{i}$ with identifier $\texttt{ID}_\texttt{i}$ as the associated data in AES-GCM. The ciphertext itself is binded with $\texttt{ID}_\texttt{i}$. We only need to consider rollback attacks. An attack that changes $\texttt{ctr}_\texttt{i}$ to a steal value $\texttt{ctr}_\texttt{i}^{\texttt{steal}}$ triggers an alarm to CC because a new packet from i-th SM with $\texttt{ctr}_\texttt{i} > \texttt{ctr}_\texttt{i}^{\texttt{steal}} + 1$ indicates the gateway has been rollbacked.

The *control enclave* which stores all SM initialization keys accesses the database with Merkle tree. The root hash is inside the enclave. During SM initialization, key $\mathcal{K}^{\texttt{i}}_{\texttt{init}}$ is labeled as "$\texttt{void}$" to avoid double registration. This requires freshness which is guaranteed by Merkle tree and *control enclave*. □

C. Availability

We provide robustness to increase system availability by the GW restart protocol of Section V-E. During short-term abrupt failures, the GW can restart the enclave and recollect the data feed from *control enclave*. Lost periodic reports will be retried upon GW request. If an adversary prevents GW from restarting successfully, e.g. keep restarting a gateway any number of times, CC will not receive regular report and alarm will then be triggered.

D. Security for Initialization

The security of our system relies on the initialization phases, including CC/GW initialization and SM initialization. The initialization of CC and GW is actually the setup of the *gateway enclave* and the *control enclave*. The setup phase of enclaves is secure according to our security model. Therefore, the CC/GW initialization cannot be broken by the adversary.

The UD takes an important role in the SM initialization phase, which involves four entities, the SM, the UD, the *gateway enclave* and the *control enclave*. As the communications between enclaves are secure and the SM cannot be compromised by the adversary, the security for the UD is essential for the initialization. Due to the limited resource of the SM, there cannot be a secure channel establish between the SM and the UD. But, the UD's role is to forward the message between the SM and the *gateway enclave*, and prove the legitimacy of the *gateway enclave* for the user. Therefore, even if the UD is compromised, the adversary cannot break the system, because he/she has no decryption key and cannot forge any message.

VIII. Performance Evaluation

In this section, we evaluate the performance of our proposed system. We first describe the experimental environment and performance benchmarks. Then, we evaluate the networking and processing overhead of the proposed protocols. Finally, experimental results show that the functions implemented in our system represent a significant performance improvement over the existing cryptography-based solutions.

A. Experimental Setup & Performance Benchmarks

TABLE I
Time complexity of operations in SGX.

Operation	Time
Create enclave	5.6 ms
Sealing (0.1 KB)	0.015 ms
Unsealing (0.1 KB)	0.01 ms
Remote Attestation	39 ms
ECDSA signing (0.1 KB)	0.69 ms
ECDSA verification (0.1 KB)	1.21 ms
AES-GCM 128-bit encryption (0.1 KB)	0.0011 ms
AES-GCM 128-bit decryption (0.1 KB)	0.0017 ms

Experimental setup. Our experimental platform runs Windows10 enterprise on Intel Kaby Lake i5-7500@3.40GHz processor, with 8 GB RAM and 128 GB SSD. We developed and compiled our code in Visual Studio 2015 with Intel(R) SGX SDK 1.8 and Intel(R) SGX PSW 1.8. The asymmetric cryptography used in our system for signing is 256-bit ECDSA. We use Diffie-Hellman key exchange to establish shared keys, and 128-bit AES-GCM for symmetric message encryption and authentication. In the following analysis, each experiment has been repeated 100 times and average results are reported.

Performance benchmarks. TABLE I provides the time complexity of the basic operations used in SGX. From this table, we can see that symmetric encryption and decryption are very fast (less than 1 microsecond). The most expensive operation is remote attestation. The main reason is that the verifier needs to interact with Intel Attestation Server via network.

B. Performance of the Protocols

The protocol performance is determined mainly by network complexity and time complexity. Since the size of network packages in these protocols is relatively small (typically, < 100 bytes), we only consider the number of communications (message complexity) for the network complexity. The time complexity of each protocol is contributed by the time complexity of operations shown in TABLE I. The complexities of time and network of the proposed protocols are shown in TABLE II.

System initialization protocol. Both *gateway enclave* and *control enclave* need remote attestation, which is the most

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2938875, IEEE Transactions on Information Forensics and Security

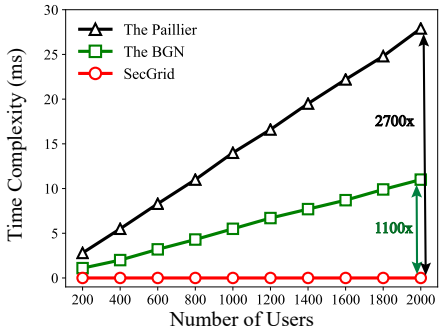


Fig. 7. Time complexity of data aggregation.

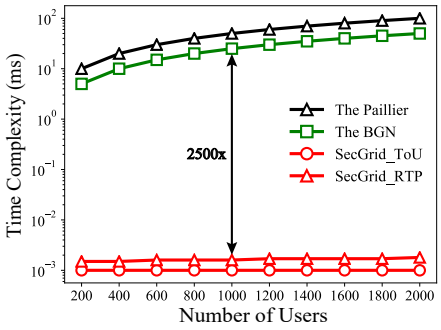


Fig. 8. Time complexity of dynamic pricing.

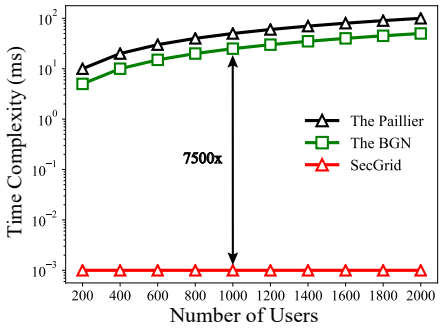


Fig. 9. Time complexity of load forecasting.

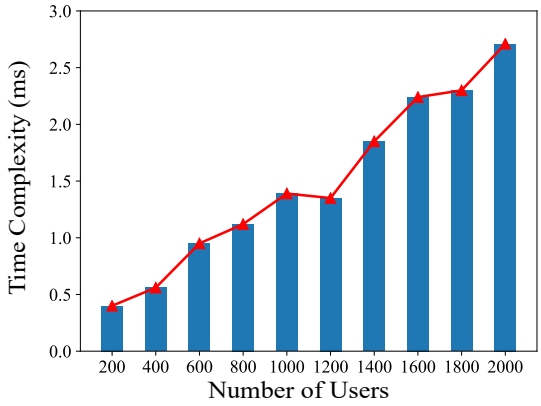


Fig. 6. Time complexity of data transmission.

TABLE II
PERFORMANCE OF PROTOCOLS

	SM	UD	Gateway Enclave Processing	Control Enclave Processing	Network
System Initialization	–	–	41.8 ms	40.8 ms	4
SM Initialization	≈ 0	33.3 ms	9.5 ms	1.69 ms	8
Periodic Report	≈ 0	–	2.7 ms	2.1 ms	4
GW Restart	≈ 0	–	$3.1 \times n$ ms	< 1 ms	$n+2$

expensive operation in this protocol (about 39 ms each). The protocol requires key exchange between the two enclaves. We can ignore the computational cost of key generation. The most expensive parts are signing and verifying, which take 0.69 ms and 1.21 ms, respectively. Other operations are relatively faster, e.g. sealing, which need less than 0.1 ms in total.

SM initialization protocol. This protocol involves all four entities in our system. The SM needs to generate a symmetric key and encrypt the `Init` message with it. This procedure takes less than $10\mu s$. The UD verifies the remote attestation of the *gateway enclave* and encrypts the `Init`$'$ message with $\mathcal{PK}_{\mathsf{gw}}$, which takes about 32 ms and 0.69 ms respectively. The *gateway enclave* verifies signatures received from the UD and the *control enclave*, decrypts `Init`$'$, seals key, and generates a signature of $Echo$. All of these operations require only 9.5 ms. The *control enclave* needs one signing and a key access procedure, which totally take about 2.1 ms.

Periodic report protocol. SM encrypts every report before sending it to the *gateway enclave*, which, as we analyzed before, has almost no cost. The *gateway enclave* needs to decrypt and seal these reports, and encrypts a new report for the *control enclave* as well as a response for the SM. These operations take about 2.1 ms. We do not include the cost of executing functions here, which will be fully evaluated in Section VIII-C. The *control enclave* follows the same steps.

GW restart protocol. Once a GW restarts, the *gateway enclave* needs to restart, too. It will send a request to the *control enclave* to obtain fresh time, and ask each SM to send a new report to avoid losing report without sealing. This procedure requires a *gateway enclave* to communicate $n+2$ times with SM and *control enclave*. It takes 3.1 ms to process a report, and the *gateway enclave* requires n times processing.

C. Performance Analysis of Functions

Before the functions being executed, the encrypted data need to transmit into the *gateway enclave* from outside. We test the time complexity of data transmission, the result is shown in Fig. 6. We can see that the time complexity is very low, around 1.5 ms per 1000 users.

We implement the three functions mentioned in our paper, namely, data aggregation, dynamic pricing, and load forecasting. Existing cryptography-based schemes usually use Paillier or BGN cryptosystem to realize homomorphic operations in ciphertext. To compare our implementations with them, we also implement these three functions with Paillier and BGN cryptosystem, which are denoted as the Paillier and the BGN in the following description. Details are described as follows.

Data aggregation. As shown in Fig. 7, our SecGrid has the lowest cost among all schemes. When the number of users is 2000, the time complexity of ours is 0.011 ms, which is $1100\times$ and $2700\times$ faster than the Paillier and the BGN, respectively.

Dynamic pricing. We implement two dynamic pricing algorithms, ToU and RTP. As for Paillier and BGN, we use fixed-price scheme, i.e., they only need to perform homomorphic multiplication, which in fact has less time complexity than existing dynamic pricing schemes. As shown in Fig. 8, the

time complexity of our scheme is stable around $1\mu s$, which is $10^5\times$ faster than that of the Paillier and the BGN.

Load forecasting. Our load forecasting model is free from using Paillier and BGN that need to perform homomorphic addition and multiplication. From Fig. 9, we can see that our implementation is $2500\times$ and $7500\times$ faster than the Paillier and the BGN, respectively, and the time complexity of our protocol does not increase as the number of users increases.

In summary, the system and SM initialization can be finished within 100 ms and 50 ms, respectively, and the processing of periodic report including three functions can also be completed within around $3 \times n$ ms, where n is the number of reports or users. The most expensive operation in this system is remote attestation, followed by ECDSA signing and verifying. Other operations usually take less than 10 ms. Overall, compared to cryptography-based solutions, our SecGrid system has much better performance.

IX. Conclusion

In this paper, we presented a practical smart grid system, named SecGrid, to enable rich functionalities without leakage of customers' private data. With our system, smart meters only need to support AES-GCM instead of heavy complex cryptography. Also, the gateway can use rich functionalities to process customers' private data at a high speed with privacy preserving. We proved that our design is sufficiently secure against malicious adversaries. We also implemented SecGrid, and thoroughly evaluated its performance. The experimental results show that our system is very efficient, as the implemented functionalities far outperform existing solutions in terms of time complexity.

Acknowledgements

The authors sincerely thank the editor, Dr. Aris Gkoulalas Divanis, and all the anonymous reviewers for their valuable suggestions that have led to the present improved version of the original manuscript. This work is supported in part by the National Key R&D Program of China under Grant No. 2016YFB0800301, the National Natural Science Foundation of China under Grant No. 61972371 and No. 61671420, and Youth Innovation Promotion Association Chinese Academy of Sciences (CAS) under Grant No. 2016394.

References

[1] M. Erol-Kantarci and H. T. Mouftah, "Energy-efficient information and communication infrastructures in the smart grid: A survey on interactions and open issues," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 179–197, 2015.

[2] A. R. Khan, A. Mahmood, A. Safdar, Z. A. Khan, and N. A. Khan, "Load forecasting, dynamic pricing and DSM in smart grid: A review," *Renewable and Sustainable Energy Reviews*, vol. 54, pp. 1311–1322, 2016.

[3] D. He, S. Chan, and M. Guizani, "Win-win security approaches for smart grid communications networks," *IEEE Network*, vol. 31, no. 6, pp. 122–128, 2017.

[4] X. Liang, X. Li, R. Lu, X. Lin, and X. Shen, "UDP: Usage-based dynamic pricing with privacy preservation for smart grid," *IEEE Transactions on Smart Grid*, vol. 4, no. 1, pp. 141–150, 2013.

[5] A.-H. Mohsenian-Rad and A. Leon-Garcia, "Optimal residential load control with price prediction in real-time electricity pricing environments," *IEEE transactions on Smart Grid*, vol. 1, no. 2, pp. 120–133, 2010.

[6] I. Moghram and S. Rahman, "Analysis and evaluation of five short-term load forecasting techniques," *IEEE Transactions on Power Systems*, vol. 4, no. 4, pp. 1484–1491, 1989.

[7] M. Q. Raza and A. Khosravi, "A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings," *Renewable and Sustainable Energy Reviews*, vol. 50, pp. 1352–1372, 2015.

[8] S. Mishra, X. Li, A. Kuhnle, M. T. Thai, and J. Seo, "Rate alteration attacks in smart grid," in *Proceedings of the 34th IEEE International Conference on Computer Communications (INFOCOM)*, 2015, pp. 2353–2361.

[9] Y. Liu, S. Hu, and T.-Y. Ho, "Leveraging strategic detection techniques for smart home pricing cyberattacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 220–235, 2016.

[10] J. Zhao, T. Jung, Y. Wang, and X. Li, "Achieving differential privacy of data disclosure in the smart grid," in *Proceedings of the 33th IEEE International Conference on Computer Communications (INFOCOM)*, 2014, pp. 504–512.

[11] S. Li, K. Xue, Q. Yang, and P. Hong, "PPMA: Privacy-preserving multi-subset aggregation in smart grid," *IEEE Transactions on Industrial Informatics, Avaliable online*, 2017.

[12] M. A. Rahman, M. H. Manshaei, E. Al-Shaer, and M. Shehab, "Secure and private data aggregation for energy consumption scheduling in smart grids," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 2, pp. 221–234, 2017.

[13] T. W. Chim, S.-M. Yiu, V. O. Li, L. C. Hui, and J. Zhong, "PRGA: Privacy-preserving recording & gateway-assisted authentication of power usage information for smart grid," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 1, pp. 85–97, 2015.

[14] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the 17th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 1999, pp. 223–238.

[15] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Proceedings of the 3rd Theory of Cryptography Conference (TCC)*, 2005, pp. 325–342.

[16] M. d. O. R. Keiko V.O.Fonseca, "Demonstrator for the end-to-end secure and privacy-friendly applications for smart metering data," https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5b8e57b48&appId=PPGMS, accessed August 21, 2019.

[17] V. Y. Pillitteri and T. L. Brewer, "Guidelines for smart grid cyber security," *NIST Interagency/Internal Report (NISTIR)-7628 Rev1*, 2014.

[18] D. He, S. Chan, and M. Guizani, "Cyber security analysis and protection of wireless sensor networks for smart grid monitoring," *IEEE Wireless Communications*, vol. 24, no. 6, pp. 98–103, 2017.

[19] V. Costan and S. Devadas, "Intel SGX explained," *IACR Cryptology ePrint Archive*, 2016.

[20] S. Tan, D. De, W.-Z. Song, J. Yang, and S. K. Das, "Survey of security advances in smart grid: A data driven approach," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 397–422, 2017.

[21] K. Sha, N. Alatrash, and Z. Wang, "A secure and efficient framework to read isolated smart grid devices," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2519–2531, 2017.

[22] S. Lee, M.-W. Shih, P. Gera, T. Kim, H. Kim, and M. Peinado, "Inferring fine-grained control flow inside SGX enclaves with branch shadowing," in *Proceedings of the 26th USENIX Security Symposium (USENIX Security)*, 2017, pp. 557–574.

[23] M.-W. Shih, S. Lee, T. Kim, and M. Peinado, "T-SGX: Eradicating controlled-channel attacks against enclave programs," in *Proceedings of the 24th Annual Network and Distributed System Security Symposium (NDSS)*, 2017.

[24] S. Sasy, S. Gorbunov, and C. Fletcher, "ZeroTrace: Oblivious memory primitives from intel SGX," *Cryptology ePrint Archive*, 2017.

[25] S. Matetic, M. Ahmed, K. Kostiainen, A. Dhar, and et al., "ROTE: Rollback protection for trusted execution," in *Proceedings of the 26th USENIX Security Symposium (USENIX Security)*, 2017, pp. 1289–1306.

[26] R. Strackx and F. Piessens, "Ariadne: A minimal approach to state continuity," in *Proceedings of 25th USENIX Security Symposium (USENIX Security)*, 2016, pp. 875–892.

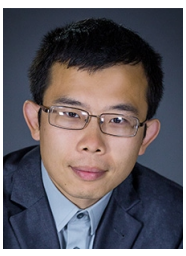
[27] L. F. Sarmenta, M. Van Dijk, C. W. O'Donnell, J. Rhodes, and S. Devadas, "Virtual monotonic counters and count-limited objects using

a TPM without a trusted OS," in *Proceedings of the first ACM workshop on Scalable trusted computing (STC)*. ACM, 2006, pp. 27–42.
[28] W. Tushar, C. Yuen, D. B. Smith, and H. V. Poor, "Price discrimination for energy trading in smart grid: A game theoretic approach," *IEEE Transactions on Smart Grid*, vol. 8, no. 4, pp. 1790–1801, 2017.
[29] F. Ye, Y. Qian, and R. Q. Hu, "A real-time information based demand-side management system in smart grid," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 329–339, 2016.
[30] S. Misra, S. Bera, and T. Ojha, "D2P: Distributed dynamic pricing policyin smart grid for phevs management," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 3, pp. 702–712, 2015.
[31] A. Ahmad, N. Javaid, M. Guizani, N. Alrajeh, and Z. A. Khan, "An accurate and fast converging short-term load forecasting model for industrial applications in a smart grid," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2587–2596, 2017.
[32] S.-C. Chan, K. M. Tsui, H. Wu, Y. Hou, Y.-C. Wu, and F. F. Wu, "Load/price forecasting and managing demand response for smart grids: Methodologies and challenges," *IEEE signal processing magazine*, vol. 29, no. 5, pp. 68–85, 2012.
[33] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors," in *Proceedings of the 25th USENIX Security Symposium (USENIX Security)*, 2016, pp. 619–636.
[34] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, and M. Russinovich, "VC3: Trustworthy data analytics in the cloud using SGX," in *Proceedings of the 36th IEEE Symposium on Security and Privacy (SP)*, 2015, pp. 38–54.
[35] W. Zheng, A. Dave, J. G. Beekman, R. A. Popa, and et al., "Opaque: An oblivious and encrypted distributed analytics platform," in *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017, pp. 283–298.
[36] B. Fisch, D. Vinayagamurthy, D. Boneh, and S. Gorbunov, "Iron: functional encryption using Intel SGX," in *Proceedings of the 24th ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 765–782.
[37] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town Crier: An authenticated data feed for smart contracts," in *Proceedings of the 23th ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016, pp. 270–282.
[38] A. Ahmad, K. Kim, M. I. Sarfaraz, and B. Lee, "Obliviate: A data oblivious file system for intel SGX," in *25th Annual Network and Distributed System Security Symposium (NDSS)*, 2018.
[39] C. Priebe, K. Vaswani, and M. Costa, "EnclaveDB: A secure database using SGX," in *EnclaveDB: A Secure Database using SGX*. IEEE, 2018, p. 0.
[40] M. Esmalifalak, L. Liu, N. Nguyen, R. Zheng, and Z. Han, "Detecting stealthy false data injection using machine learning in smart grid," *IEEE Systems Journal*, vol. 11, no. 3, pp. 1644–1652, 2017.
[41] Y. He, G. J. Mendis, and J. Wei, "Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2505–2516, 2017.
[42] O. Oleksenko, B. Trach, R. Krahn, M. Silberstein, and C. Fetzer, "Varys: Protecting SGX enclaves from practical side-channel attacks," in *Proceedings of 2018 USENIX Annual Technical Conference (ATC)*, 2018, pp. 227–240.
[43] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M. L. Stillwell *et al.*, "SCONE: Secure linux containers with intel SGX," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 689–703.
[44] F. L. Quilumba, W. J. Lee, H. Huang, D. Y. Wang, and R. L. Szabados, "Using smart meter data to improve the accuracy of intraday load forecasting considering customer behavior similarities," *IEEE Transactions on Smart Grid*, vol. 6, no. 2, pp. 911–918, 2015.
[45] L. V. Silva, P. Barbosa, R. Marinho, and A. Brito, "Security and privacy aware data aggregation on cloud computing," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 6, 2018.

Shaohua Li received the B.E. degree from the Department of Information Security, University of Science and Technology of China (USTC) in 2016, and received his M.S. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2019. Now he is a Ph.D. student in Department of Computer Science in ETH Zurich, Switzerland. His research interests include network security protocol design and analysis. This work was completed when he was a graduated student in Department of EEIS, USTC.

Kaiping Xue (M'09-SM'15) received his bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC) in 2003 and received his Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From May 2012 to May 2013, he was a postdoctoral researcher with Department of Electrical and Computer Engineering, University of Florida. Currently, he is an Associate Professor in the Department of Information Security and Department of EEIS, USTC. He is serving on the editorial boards of several journals, including IEEE Transactions on Wireless Communications (TWC), IEEE Transactions on Network and Service Management (TNSM), Ad Hoc Networks, IEEE Access and China Communications. He also served as a guest editor of IEEE Journal on Selected Areas in Communications (JSAC) and a lead guest editor of IEEE Communications Magazine. His research interests include next-generation Internet, distributed networks and network security. He is the corresponding author of this paper.

David S.L. Wei (SM'07) received his Ph.D. degree in Computer and Information Science from the University of Pennsylvania in 1991. From May 1993 to August 1997 he was on the Faculty of Computer Science and Engineering at the University of Aizu, Japan (as an Associate Professor and then a Professor). He has authored and co-authored more than 100 technical papers in various archival journals and conference proceedings. He is currently a Professor of Computer and Information Science Department at Fordham University. His research interests include cloud computing, big data, IoT, and cognitive radio networks. He was guest editors or lead guest editors of several special issues in IEEE Journal on Selected Areas in Communications, IEEE Transactions on Cloud Computing and IEEE Transactions on Big Data. He also served as an Associate Editor of IEEE Transactions on Cloud Computing, 2014-2018, and an Associate Editor of Journal of Circuits, Systems and Computers, 2013-2018.



Hao Yue received his B.Eng. degree in Telecommunication Engineering from Xidian University, Xi'an, China, in 2009, and Ph.D degree in Electrical and Computer Engineering from University of Florida, Gainesville, FL, USA, in 2015. He is now an Assistant Processor with the Department of Computer Science, San Francisco State University, San Francisco, CA, USA. His research interests include cyber-physical systems, cybersecurity, wireless networking, and mobile computing.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2938875, IEEE Transactions on Information Forensics and Security

Nenghai Yu received the B.S. degree from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 1987, the M.E. degree from Tsinghua University, Beijing, China, in 1992, and the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), Hefei, China, in 2004. Since 1992, he has been a Faculty in the Department of Electronic Engineering and Information Science, USTC, where he is currently a Professor. He is the Executive Director of the Department of EEIS, USTC, and the Director of the Information Processing Center, USTC. He has authored or co-authored more than 130 papers in journals and international conferences. His research interests include multimedia security, multimedia information retrieval, video processing, and information hiding.

Peilin Hong received her B.S. and M.S. degrees from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1983 and 1986. Currently, she is a Professor and Advisor for Ph.D. candidates in the Department of EEIS, USTC. Her research interests include next-generation Internet, policy control, IP QoS, and information security. She has published 2 books and over 150 academic papers in several journals and conference proceedings.