Look-Up Table based FHE System for Privacy Preserving Anomaly Detection in Smart Grids

Ruixiao Li[†], Shameek Bhattacharjee[‡], Sajal K. Das[$], and Hayato Yamana[†]
[†]Department of Computer Science and Communications Engineering, Waseda University, Tokyo, Japan
[‡]Department of Computer Science, Western Michigan University, Kalamazoo, USA
[$]Department of Computer Science, Missouri University of Science and Technology, Rolla, USA
E-mail: {liruixiao, yamana}@yama.info.waseda.ac.jp, shameek.bhattacharjee@wmich.edu, sdas@mst.edu

***Abstract*—In advanced metering infrastructure (AMI), the customers' power consumption data is considered private but needs to be revealed to data-driven attack detection frameworks. In this paper, we present a system for privacy-preserving anomaly-based data falsification attack detection over fully homomorphic encrypted (FHE) data, which enables computations required for the attack detection over encrypted individual customer smart meter's data. Specifically, we propose a homomorphic look-up table (LUT) based FHE approach that supports privacy preserving anomaly detection between the utility, customer, and multiple partied providing security services. In the LUTs, the data pairs of input and output values for each function required by the anomaly detection framework are stored *to enable arbitrary arithmetic calculations over FHE*. Furthermore, we adopt a private information retrieval (PIR) approach with FHE to enable approximate search with LUTs, which reduces the execution time of the attack detection service while protecting private information. Besides, we show that by adjusting the significant digits of inputs and outputs in our LUT, we can control the detection accuracy and execution time of the attack detection, even while using FHE. Our experiments confirmed that our proposed method is able to detect the injection of false power consumption in the range of 11-17 secs of execution time, depending on detection accuracy.**

***Index Terms*—anomaly (attack) detection, smart grid, privacy-preserving, FHE, look-up table**

I. Introduction

Advanced Metering Infrastructure (AMI) in a smart (electrical) grid is an IoT system, consisting of communication networks, data management systems, and smart meters that record power consumption from every customer [1], [2]. The communication network connects the individual customer's smart meters to the data management system of the utility.

The smart meters and the AMI communication network are vulnerable to data integrity attacks where an adversary injects false power consumption data that pose serious threats to both the utilities and customers. Data falsification in AMI mostly focuses on electricity theft [3]–[6], i.e., the power consumption reported by the smart meter is lower than the true value, which we call a *deductive attack*. By contrast, a false increase in the reported power consumption is called an *additive attack* via a physical load-altering. An additive attack on the smart meters of a competitor company would increase the bills of its customers, thereby reducing the trust of customers on that company and also create a false power surge. The *camouflage attack* [7] launches the deductive and additive attacks simultaneously while keeping the mean power consumption unchanged, i.e., a set of the customer meters is under a deductive attack while another equal set of the meters is under an additive attack. Such attacks benefit one set of customers with lesser bills at the expense of another set without raising suspicion.

Since the utility controls customer billing, power quality and efficiency of the smart grid, detecting such attacks in utilities are critical. A common way of data integrity attack detection is to deploy an anomaly detection system that learns patterns in data that do not conform to expected benign behavior [8] to indicate the presence of attacks. There are two broad approaches for attack detection in smart grids via anomaly detection. One approach is to check abnormal information via hardware, such as the use of power line modems by Passerini et al. [9]. Another approach is to build an data driven anomaly detection system using software, such as machine learning [10]–[12], deep learning [13], and neural networks [14]. Since data driven approaches do not need extra hardware, they are feasible for community scale smart living IoT such as AMI.

A. Motivation and Challenges

From the above discussion, it is clear that data driven approaches are crucial for increased visibility and vigilance in the AMI for attack detection. However, paradoxically, an anomaly-based attack detection system requires the power consumption data of every customer to be revealed to the anomaly detection system, thereby compromising the privacy of the customers. Customers advocacy groups and governments are concerned about the privacy-sensitive nature of customer's smart data, which is misaligned with security goals of the grid utility. This motivates the need to design privacy-preserving data driven security frameworks for AMI.

To construct privacy-preserving anomaly detection systems, many studies have used differential privacy (DP) [15], which protects sensitive data by adding random noise. A disadvantage of DP is that it does not support exact computation, thereby limiting the accuracy of the results. Another approach is the use of secure multiparty computation (SMC) [16], [17] to protect sensitive data. SMC is based on joint operations involving multiple parties via secret sharing. With secret

sharing, sensitive data are divided into different parties for storage and computations. However, the communication costs of SMC are huge for streaming data applications like AMI. A third alternative is the use of homomorphic encryption (HE) [18], [19] that allows computations without the need to ever decrypt the individual customer's data. While HE has a smaller communication cost, it can only support additions and multiplications, resulting in difficulty to adopt anomaly detection techniques that involve advanced math operations. Furthermore, the execution overhead and the accuracy of an HE-based anomaly detection framework should not degrade due to the privacy requirement.

In [20], the authors introduced an anomaly detection system using fully homomorphic encryption (FHE) [21], which can also perform anomaly detection on deductive, additive, and camouflage attacks. Note that FHE is a type of HE enabling an unlimited number of computations on encrypted data. They proposed a specific algorithm to adopt FHE to enable anomaly detection. However, they did not offer flexibility for supporting arbitrary calculations and did not have provisions that allow a smart grid AMI utility to choose between desired levels of achievable anomaly detection accuracy and execution latency. To solve the above problems, we propose a system with a look-up table (LUT) based FHE approach integrated with private information retrieval (PIR) that can support arbitrary calculations, allow utilities a trade-off between execution latency and attack detection accuracy.

B. Contributions of This Work

The contributions of this work are fourfold.

- We propose a privacy-preserving anomaly attack detection system using a LUT-based FHE integrated with PIR specifically for an AMI infrastructure. Our novel approach allows a utility to flexibly tune the anomaly detection accuracy and control the execution latency, which is a challenge in FHE. Specifically, our LUT-based FHE approach, i.e., preparing input-output value pairs as LUT and replacing functions to a search, allows the implementation of arbitrary functions even if the traditional FHE itself cannot handle complex functions (e.g., logarithm, inverse, division). To the best of our knowledge, this is the first implementation of flexible control of the detection accuracy and the execution latency over FHE. Furthermore, a PIR is adopted to enable *approximate search* without revealing any information to both the utility and the data management system. Note that approximate search enables the search even if LUT has no exact match input and returns the result whose input is nearest to the query.
- To control the detection accuracy, we provision for a *flexible precision parameter* that is learned based on the utility's desired anomaly detection accuracy, which allows a utility to understand the trade-offs between execution time and the anomaly detection accuracy. The precision parameter adaptively controls the number of rows in the LUT for FHE. Therefore, the execution time of LUT processing will be shorter when using fewer rows in the LUT but this comes at the cost of the anomaly detection accuracy. The suitable precision parameter of the LUT is found based on the training data set.
- To speed up the FHE processing, we add a *dropping of least significant bits* feature in our LUT-based FHE system. This allows that each of the entries in the LUT has a smaller plaintext space to compute the large numbers involved in FHE, which further reduces the execution time of our system.
- We implemented our framework with a Raspberry PI as a proxy for a smart meter. We deployed the look-up table based privacy-preserving anomaly detection system on a server and measured the real execution time for varying utility specified accuracy levels. Our results show that our method can detect the presence of attacks over encrypted data within 11-17 seconds on the server, depending on the desired accuracy level.

The rest of the paper is organized as follows. Preliminary techniques used in this study are presented in Section II. We introduce the proposed technique in Section III. Section IV presents the experimental evaluation and we conclude this work in Section V.

II. Preliminaries and Background

This section introduces the background, preliminaries and assumptions of the work. In our system, the smart meters are assumed honest, and the other parties are assumed honest-but-curious parties, i.e., they follow the protocol but try to find out as much as possible about the data. We assume the data integrity attack occurs before the smart meters encrypts the data, which is practical for many transactions and load altering exploits that lead to data falsification.

A. Anomaly Detection Metric

For the anomaly detection, we adopt the framework proposed in [7] which detects attacks with high detection sensitivity while minimizing the false alarms. In [7], the basic anomaly detection metric is the ratio between the harmonic mean (HM) and arithmetic mean (AM) of the power consumption from all smart meters in a micro-grid of size N. The authors show that another stateful metric known as residual under the curve (RUC) can be derived from the HM-AM ratio can detect various attacks, including deductive, addictive, and camouflage attacks. We denote N as the number of meters in each region, where the anomaly detection is performed in each region. The power consumption from a set of meters is denoted by $\boldsymbol{p}_t = [p_t^1, ..., p_t^N]$ at a time slot t. The power consumption from the i-th meter is shown as $p_t^i \in \mathbb{R}^+$. Let $P_t^i := ln(p_t^i + 2)$. We denote the harmonic mean and arithmetic mean at time slot t as HM_t and AM_t, where

$$HM_t = \frac{N}{\sum_{i=1}^{N} \frac{1}{P_t^i}}, \quad AM_t = \frac{\sum_{i=1}^{N} P_t^i}{N}. \tag{1}$$

HM_t and AM_t are calculated for each time slot t over a time window T. Each T contains 24 time-slots, i.e., hourly. Then

$T \in 1, ..., 365$ represents each day of a year. The HM-AM ratio Q^r of the day T is computed as follows:

$$Q^r(T) = \frac{\sum_{t \in T} HM_t(T)}{\sum_{t \in T} AM_t(T)} \quad (2)$$

This is a highly stable metric for anomaly detection in a smart grid, as demonstrated in [7]. Due to the page limitation, we only give a high-level description of it. (For further details, refer to the paper [7].) There are two phases in anomaly detection: the training and test phases. In the training phase, a utility determines the safe margin for the ratio metric. The threshold parameter κ is determined by the mean μ_d and standard deviation s_d of the daily ratio distribution, where $\kappa \in (0, 3s_d]$. The standard limit range of the safe margin is $[\mu_d - \kappa, \mu_d + \kappa]$. Besides, by adopting the residual under the curve (RUC) metric [7], which shows the transition of the residuals between the HM-AM ratio and the chosen safe margin over a sliding time frame, the tier two detector is constructed to detect the anomaly.

B. Brakerski-Fan-Vercautere (BFV) scheme

Our work is based on the Microsoft/SEAL homomorphic encryption library [23], which implements the Brakerski-Fan-Vercautere (BFV) scheme [24]. The BFV scheme performs modular arithmetic on the encrypted integers. A set of integers is encrypted into a single ciphertext for SIMD-style execution using the BFV scheme and packing technique [25] based on the Chinese remainder theorem (CRT). We denote the number of integers that can be packed into one ciphertext as l, and we call l as the number of slots in the FHE setting.

C. Look-up Table with FHE

The general philosophy of an LUT-based FHE system [21] is that we can implement the privacy-preserving anomaly detection system with arbitrarily different complex metric functions. To improve the practicality of FHE, we adopt a LUT, which evaluates approximations of arbitrary functions by changing the calculations to a search technique. In a LUT, input-output data pairs for a function are prepared, so that we can search the result of the function.

Note that FHE itself cannot handle any arithmetic operation except additions and multiplications. On the other hand, the base anomaly detection framework requires more complex arithmetic operations such as logarithms, division, inverse. Therefore, adopting an LUT based approach, we can place the LUT of any specific function in the cloud server and obtain the result from the LUT directly as a query response mechanism that allows calculations of arbitrary math functions without sacrificing privacy. We can omit arithmetic calculations to speed up.

III. Proposed Approach

Figure 1 presents an overview of our proposed system. There are five parties: the utility, computation server (CS), LUT provider, data collector (DC) and N smart meters. All parties except smart meters are honest but-curious parties, i.e., they follow the protocol but try to find out as much as possible about the data.

We assume the data integrity attack occurs before the smart meters encrypt their data. Moreover, the five parties do not collude with each other. All parties hold the same public key; only the utility holds the secret key; the CS holds the re-linearization key and the galois key. The public key is used for encryption and calculation, the secret key for decryption, while the re-linearization key and the galois key are used for re-linearization and rotation.

Firstly, the LUT provider constructs the LUTs and sends them to the CS (Section III-A). Then, the smart meters send their power consumption data to the DC every time slot (in our setting, every one hour) (Section III-B). After collecting the power consumption data in a specific area, the data collector sends them to the CS every time slot. Subsequently, the CS computes the HM-AM ratio and sends it to the utility. During the calculation in the CS, the CS exchanges encrypted intermediate results with the utility for the LUT processing (Section III-C). Finally, the utility decides whether there is an attack or not.

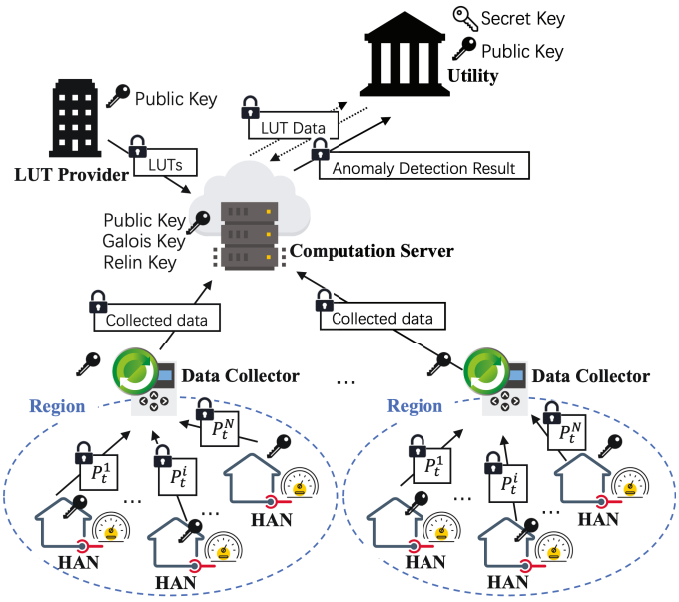

Fig. 1. Overview of the Proposed Approach

A. Initialization

Key generation (Utility side): The utility generates the secret key sk, the public key pk, the re-linearization key rk and the galois key gk. Holding sk for itself, sharing pk to other parties, and rk, gk to the CS.

Construction of LUTs (LUT provider side): The twelve LUTs including input and output tables shown in Table I are prepared and also explained in this subsection. Each entry in input tables are corresponding to the entry in output table. All tables are constructed by the LUT provider as matrices. With the *ciphertext packing feature*, each row of LUT is represented by one ciphertext.

TABLE I
PREPARED LOOK-UP TABLES

Calculation of	$\mathbf{x}$ (values in input table T_{in})	$\mathbf{f(x)}$ (values in output table T_{out})
HM_t	$\sum_{i=1}^{N}(\frac{1}{P_t^i})$	$f_1(x) = \frac{N}{x}$
AM_t	$\sum_{i=1}^{N}(P_t^i)$	$f_2(x) = \frac{x}{N}$
$\mathcal{H}_1$	$\sum_{t=1}^{24} HM_t$	$f_3(x) = \lfloor \frac{x}{100} \rfloor$
$\mathcal{H}_2$		$f_4(x) = x - f_3(x) \times 100$
$\mathcal{A}_1$	$\sum_{t=1}^{24} AM_t$	$f_5(x) = \lfloor \frac{1}{x}/100 \rfloor$
$\mathcal{A}_2$		$f_6(x) = \frac{1}{x} - f_5(x) \times 100$
$res2/100$	$res2$	$f_7(x) = \lfloor x/100 \rfloor$

In this work, we adopt the LUTs with the FHE for computing harmonic mean, arithmetic mean, and anomaly detection ratio, i.e., HM-AM ratio Q^r. Because Q^r is defined as a daily HM-AM ratio, we need to calculate each time-slot's HM_t and AM_t. Then, we need to calculate the sums of HM_t and AM_t, over a 24 hour period. Finally, $\sum_{t=1}^{24} HM_t$ is divided by $\sum_{t=1}^{24} AM_t$ to compute Q_r.

In our implementation, we replace division with a multiplication (shown in Eqn. 3) to shorten the execution time. The reason why we do not compute the division function with FHE using LUT processing, is because the division function requires a two-input function. Note, that the number of the entries in the output LUT for a multi-input function is the product of the entries of each input LUT. This would result in longer execution time due to the large output size in the LUT.

$$Q^r = \frac{\sum_{t=1}^{24} HM_t}{\sum_{t=1}^{24} AM_t} = \sum_{t=1}^{24} HM_t \times \frac{1}{\sum_{t=1}^{24} AM_t} \tag{3}$$

We use the integer-based FHE scheme in this work. In order to increase the accuracy, we scaled the input and output decimals with a *precision parameter* $2^{\mathfrak{p}}$ and round as integers. After we retrieve the $\sum_{t=1}^{24} HM_t$ and $\frac{1}{\sum_{t=1}^{24} AM_t}$ from LUT, the result of the multiplication between these two values becomes too large, which results in overflow of the plaintext space that cannot be encrypted. Thus, we *drop the least significant bits* of the result. As the significant digits of maximum scaled ratio Q^r in our experiment is 9 but the significant digits of the plaintext space is 5, we discard the last 4 digits of the Q^r result. By analyzing the significant digits on the calculation of (5), we can reduce the Q^r by $10,000$ times to discard the last 4 digits, thereby we set 100 in (Eqn. 4) (because $100 \times 100 = 10,000$). Then, we express each of the $\sum_{t=1}^{24} HM_t$ and $\frac{1}{\sum_{t=1}^{24} AM_t}$ as a polynomial shown below, where $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{A}_1$ and $\mathcal{A}_2$ are coefficients. Eqn. (7) shows the final result.

$$\begin{aligned} \sum_{t=1}^{24} HM_t &= \mathcal{H}_1 \times 100 + \mathcal{H}_2 \\ \frac{1}{\sum_{t=1}^{24} AM_t} &= \mathcal{A}_1 \times 100 + \mathcal{A}_2 \end{aligned} \tag{4}$$

Note that, the 100 is not a fixed value; it depends on the plaintext space in FHE setting and the significant digits we want to keep.In FHE, the larger plaintext space leads to more execution time. By dropping of least significant bits, we allow to compute large numbers with a smaller plaintext space which can reduce the execution time.

After computing $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{A}_1$, and $\mathcal{A}_2$ with LUTs shown in Table I, the daily ratio Q^r is computed as follows.

$$\begin{aligned} Q^r &= \sum_{t=1}^{24} HM_t \times \frac{1}{\sum_{t=1}^{24} AM_t} \\ &= (\mathcal{H}_1 \times 100 + \mathcal{H}_2) \times (\mathcal{A}_1 \times 100 + \mathcal{A}_2) \\ &= \mathcal{H}_1 \times \mathcal{A}_1 \times 10,000 \\ &\quad + (\mathcal{H}_1 \times \mathcal{A}_2 + \mathcal{H}_2 \times \mathcal{A}_1) \times 100 + \mathcal{H}_2 \times \mathcal{A}_2 \end{aligned} \tag{5}$$

Here, we define $res1$ and $res2$ as follows.

$$\begin{aligned} res1 &= \mathcal{H}_1 \times \mathcal{A}_1 \\ res2 &= \mathcal{H}_1 \times \mathcal{A}_2 + \mathcal{H}_2 \times \mathcal{A}_1 \end{aligned} \tag{6}$$

Then, we have the following formula to discard the last four digits of Q^r,

$$Q^r := res1 + \frac{res2}{100} \tag{7}$$

The whole aforementioned LUTs are constructed by the LUT provider.

B. Smart Meter (User) Side Computation

Power consumption data is retrieved by smart meters, encrypted, and sent to the data collector. Because the LUT method [21], [22] accepts a *packed ciphertext* whose all elements are identical to a search query, each smart meter creates two vectors $\boldsymbol{v}(P_t^i)$ and $\boldsymbol{v}(\frac{1}{P_t^i})$, whose all elements are same and computed from the power consumption P_t^i as:

$$\begin{aligned} \boldsymbol{v}(P_t^i)[k] &= Round(2^{\mathfrak{p}^{AMin}} \times P_t^i), \\ \boldsymbol{v}(\frac{1}{P_t^i})[k] &= Round(2^{\mathfrak{p}^{HMin}} \times \frac{1}{P_t^i}) \end{aligned} \tag{8}$$

, where $0 \leq k < l$ and l represents the length of these two vectors; $P_t^i := ln(p_t^i + 2)$ is described in Section II-A. To encode multiple integers into one ciphertext, we adopt plaintext and ciphertext packing technique [25]. Since we employ integer encoding rather than bit-wise encoding to speed up the execution time, the data must be converted into integers before encryption. Thus, the data are scaled with precision parameters $2^{\mathfrak{p}}$ and rounded into integers as described in III-A.

Though the inverse of P_t^i can be computed on the CS using LUTs, the smart meters send both the original power consumption data, P_t^i, and its inverse to the CS via the DC in order to reduce the execution time in the CS.

C. Server Side Computation

The processing steps are shown in Figure 2 which includes three times LUT processing. All the processes need to be executed with encryption except the decryption of intermediate results in the utility.

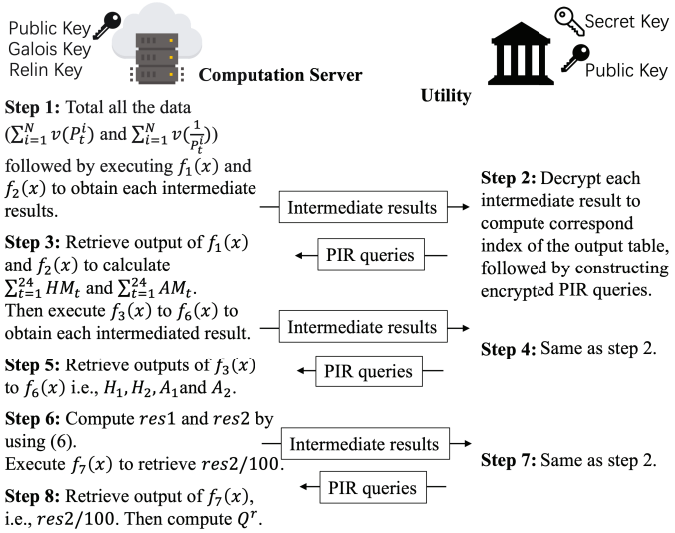

Fig. 2. LUT Processing ($f(x)$ is shown in Table I)

Step1: The CS totals all the data sent from smart meters via DC every time slot. Subsequently, we have two encrypted vectors whose elements are identical as shown below.

$$\begin{aligned} \boldsymbol{v}(P_t^i) &= [\sum_{i=1}^{N} v(P_t^i), ..., \sum_{i=1}^{N} v(P_t^i)] \\ \boldsymbol{v}(\frac{1}{P_t^i}) &= [\sum_{i=1}^{N} v(\frac{1}{P_t^i}), ..., \sum_{i=1}^{N} v(\frac{1}{P_t^i})] \end{aligned} \tag{9}$$

Each of the above two ciphertexts is input to LUT processing to calculate AM_t and HM_t respectively. The LUT processing [21], [22] is adopted to compute the function HM_t and AM_t of each time slot t. The CS searches the inputs $\sum_{i=1}^{N} \boldsymbol{v}(P_t^i)$ and $\sum_{i=1}^{N} \boldsymbol{v}(\frac{1}{P_t^i})$ in input LUTs T_{in} of function AM_t shown as $f_2(x)$ and HM_t shown as $f_1(x)$ in Table I, respectively. Then, the CS sends the intermediate result to the utility. Note that the intermediate result corresponds to the output table index which is randomized [22].

Step2: After step1, the utility receives the intermediate result from the CS followed by one-time decryption. Then, the utility obtains the index of the closest entry to the input and calculates the index of the outputs in T_{out}. Subsequently, the utility *makes encrypted PIR queries* and sends them to the CS in order to let the CS extract the output.

Step3: After receiving the PIR queries from the utility, the CS extracts the corresponding HM_t and AM_t results from output LUTs, i.e., T_{out}. The CS sums the 24 hours' HM and AM to obtain $\sum_{t=1}^{24} HM_t$ and $\sum_{t=1}^{24} AM_t$ of a day. Because the 24 hours' $HM(AM)$ is input to the next LUT processing, we need to put the 24 hours' $HM(AM)$ in all slots of the corresponding input ciphertext. To fill in, we adopt *totalSum* operation [26] shown as Algorithm 1.

Then, the CS search the inputs $\sum_{t=1}^{24} HM_t$ and $\sum_{t=1}^{24} AM_t$ in input LUTs (shown as $f_3(x)$ to $f_6(x)$ in Table I) to obtain the intermediate result which is sent to the utility from the CS.

Algorithm 1: totalSum

Data: A ciphertext ct
Result: A ciphertext ct'
for $i = 0$ *to* $log_2 l$ **do**
  $ct' = ct$;
  $ct' = rotate(ct', 2^i)$;
  $ct+ = ct'$;
end

Step 4: Same as step 2, the utility makes PIR queries, encrypts them to send to the CS.

Step 5: The CS extracts $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{A}_1$, and $\mathcal{A}_2$, each of which is shown in (4), from the output LUTs by using the received PIR queries.

Step 6: The CS computes $res1$ and $res2$ according to (6), and searches $res2$ from the input LUT shown as $f_7(x)$ in Table I in order to obtain $\frac{res2}{100}$. Then, the CS sends the intermediate result to the utility.

Step 7: Same as step 2 and step 4.

Step 8: The CS extracts $\frac{res2}{100}$ from the output LUT followed by computing the daily ratio according to (7). Finally, the CS sends the final result, i.e., Q^r, to the utility.

To enhance the above process, we extend our previously proposed method [22] which adopts random sifting of encrypted intermediate results to hide the exact index from the utility. This can improve the security. In this study, the LUT provider updates each LUT to add fake values that are different each time the LUT is updated. Note that the values in LUTs are encrypted. This operation requires less time than our previous random shift operation, and it can also be guaranteed that the index is different every time so that the utility cannot know or infer any positional information of the exact value in LUT. To ensure the accuracy does not change every time, we need to guarantee that the original entries must be included in the table, and add different confusing entries. For example, if the original entries in a LUT are $[2, 5, 8]$, an updated LUT is like $[1, 2, 3, 4, 5, 8]$ or $[2, 3, 5, 6, 8, 9]$, so that the LUT holds all original entries but at the different position every time.

Anomaly detection by HM-AM ratio (Utility side): The utility decrypts the final result Q^r. Since the final result is an integer value with the decimal point raised to a higher digit, the decimal point position is restored to have the HM-AM ratio of the day. Finally, the utility adopts the residual under the curve (RUC) metric proposed in [7] to detect the anomaly.

Security Analysis: We provide an intuitive security analysis of our protocol. Our goal is to protect the privacy of power consumption data of households from all parties. Besides, the utility, the LUT provider, and the smart meters do not collude with the other parties because the utility has the secret key; the LUT provider has LUT data; the smart meters have power consumption data. Because the data owners, i.e., the smart meters and the LUT provider, do not collude with other parties and receive nothing, they cannot know anything.

As for the DC and the CS, they only handle encrypted data, and hence they never access the privacy-sensitive individual data. Only the information the utility knows is the HM-AM ratio result which is an aggregate. Although the utility decrypts the intermediate results sent from the CS, it only knows the matched indices of LUTs that are updated by the LUT provider periodically, e.g., every hour. Thus, it guarantees only the statistics showing how many times each index of the LUTs is referenced during a specific period leaks to the utility.

IV. Experimental Evaluation

We implemented the proposed system using Microsoft/SEAL Library 3.2.0 [1]. One server was prepared for the operations of three parties: the utility, the LUT provider, and the CS. Besides, one Raspberry PI 4 Model B was prepared for the operations of the smart meter. The machine information is listed in Table II. Although the machine is equipped with 18 cores, the experiment was conducted with one thread. Note that, in this implementation, we omit the DC because the DC only gathers and re-sends the data to the CS. The *initialization* is implemented in the server, the *smart meter (user) side computation* is implemented in the Raspberry PI, and the *server-side computation* is implemented in the server.

The FHE parameters used are listed in Table III. The total LUT size including all input (output) LUTs is listed in the row of LUT provider of Table V.

In this evaluation, we conducted three experiments by changing the size of LUTs, i.e., different number of data points, which affects the accuracy.

TABLE II
Machine Information

Server	
OS	CentOS Linux 7 (Core)
CPU	four Intel Xeon E7-8880 v3 @2.3 GHz (Turbo Boost: 3.1 GHz)
Main memory	3 TB
Raspberry PI 4	
RAM	4GB
CPU	a 64-bit quad-core Arm Cortex-A72 @1.5 GHz
SD card	64GB

TABLE III
FHE Parameters of SEAL

Scheme	BFV
Poly modulus degree	8,192
Coeff modulus size	218 bits
Plain modulus	786,443
Noise standard deviation	3.2
Number of slots	8,192

TABLE IV
Precision Parameters

Experiment	$\mathfrak{p}^{AMin}$	$\mathfrak{p}^{HMin}$	$\mathfrak{p}^{AMout}$ **and** $\mathfrak{p}^{HMout}$	$\mathfrak{p}^{Inv_{out}}$
#1	5	10	9	21
#2			7	19
#3			5	17

[1]https://github.com/microsoft/SEAL

The dataset used in this study includes the power consumption data of the smart grids of 200 households in Texas, USA; these were three-year data (2014 to 2016) from the Pecan Street Project [2]. We used data from 2014 and 2015 in the training phase and data from 2016 in the testing phase. The lower and upper power consumption limits were set to 50 and 6000 W during pre-processing, same as to the previous work [7], [20].

A. Precision Parameters Setting

Table IV shows the precision parameters of LUTs in three different experiments in which the size of output tables are different; experiment #1 uses the largest output tables, and experiment #3 uses the smallest ones.

We set the same precision parameters $\mathfrak{p}^{AM_{in}}$ and $\mathfrak{p}^{HM_{in}}$ and applied to all the experiments.

In the same way, we set the precision parameters $\mathfrak{p}^{AM_{out}}$ and $\mathfrak{p}^{HM_{out}}$ to move the decimal points of AM_t and HM_t as shown in (10). The objective of precision parameters is to change AM_t and HM_t to integers. By adopting the precision parameters, the values stored in the output table T_{out} of $f_1(x)$ and $f_2(x)$, shown in Table I, are multiplied by $2^{\mathfrak{p}^{AM_{out}}}$ and $2^{\mathfrak{p}^{HM_{out}}}$, respectively.

$$\begin{aligned} &Round(2^{\mathfrak{p}^{AMout}} \times \frac{\sum_{i=1}^{N} P_t^i}{N}), \\ &Round(2^{\mathfrak{p}^{HMout}} \times \frac{N}{\sum_{i=1}^{N} \frac{1}{P_t^i}}) \end{aligned} \tag{10}$$

Next, we also set the precision parameters $\mathfrak{p}^{Inv_{out}}$ to move the decimal points of $\mathcal{A}_1$ and $\mathcal{A}_2$ as shown in (11). The objective of precision parameters is to change $\mathcal{A}_1$ and $\mathcal{A}_2$ to integers. By adopting the precision parameter, the values stored in the output table T_{out} of $f_5(x)$ and $f_6(x)$, shown in Table I, are multiplied by $2^{\mathfrak{p}^{Inv_{out}}}$.

$$\begin{aligned} &Round(2^{\mathfrak{p}^{Inv_{out}}} \times \mathcal{A}_1), \\ &Round(2^{\mathfrak{p}^{Inv_{out}}} \times \mathcal{A}_2) \end{aligned} \tag{11}$$

Finally, to remove the effect of precision parameters when calculating Q^r, we calculate Q^r as shown in (12), where $Dec(Q^r)$ represents decryption of Q^r.

$$Q^r := \frac{Dec(Q^r) \times 10,000}{2^{(\mathfrak{p}^{HM_{out}} + \mathfrak{p}^{Inv_{out}})}} \tag{12}$$

B. Computation and Communication cost

Table V lists the computation time and communication cost for each step under one thread. The computation time is the average of the five tests. We tested the runtime to compute the data in 24 time-slots (one day).

Since we know the data points appeared in the training phrase, i.e., in 2014 to 2015 data, we prepare those data points in input LUTs and corresponding outputs are prepared in output LUTs. Then, we omit some of them based on the precision parameter. Firstly, the entries in the input LUT are prepared as all the appeared values in the past power

[2]https://www.pecanstreet.org/

consumption data. Then, the number of significant digits or less are removed after applying the precision parameter, which decreases the number of entries in the input LUT. For example, assume that we have three entries: 1010, 1015, and 1100. By applying precision parameter 100, i.e., dividing by 100, the entries become 10 and 11 because 10.10 and 10.15 become the same 10 by truncating the decimal point, decreasing the number of entries.

Because the time of data transmission depends on the network configuration, we only provide the size of the transmitted file. The transmitted files include 1) the keys from the utility to all parties, 2) LUTs from the LUT provider to the CS, 3) encrypted power consumption data from the smart meters to the CS via the DC, 4) intermediate results from the CS to the utility, and 5) PIR queries and the final result from the utility to the CS.

The result shows the encryption time by Raspberry PI is approximately 2.6 seconds. After collecting a day-wise (24 time-slots) encrypted power consumption data from all houses, the CS can compute the anomaly detection metric Q^r. In the experiments, we tested daily Q^r. The AM_t and HM_t is computed per hour (per time slot). In summary, as shown in the row of SUM in Table V, the anomaly detection needs 11 s to 17 s depending on the size of the LUT, i.e., depending on the accuracy.

C. Accuracy of Anomaly Detection

Similar to the work of Ishimaki et al. [20], we used the receiver operating characteristics (ROC) curve to compare the accuracy of anomaly detection with and without secure computation using LUT-based FHE. To evaluate the performance of the anomaly detector, ROC curves were obtained using different standard limits to check the accuracy of the real anomaly detection and false attack alarms.

Figure 3 to Figure 8 show the ROC curve of deductive attack, camouflage attack, and additive attack, respectively. Such attacks are with two extreme $\Delta_{avg} = 200$W and 800W.

The result shows the stability of Experiment #1 is the best, and the detector performance in experiments of the ciphertext calculation is similar to the results of the plaintext calculation. In Experiment #2, the detector performance drops under the camouflage attack when Δ=200 W. And most of the detector performance drops in Experiment #3.

V. Conclusion

We proposed a LUT-based FHE system for privacy-preserving anomaly-attack detection. The experiments confirmed that our proposed method is able to detect the injection of false power consumption in the range of 11-17 secs of execution time, depending on the detection accuracy. To the best of our knowledge, this is the first implementation of flexible control of the anomaly detection accuracy and the execution time over FHE. We can detect deductive, additive, and camouflage attacks with reasonable accuracy. Compared to the related work [20], our proposed method is flexible where we can simply change the accuracy and the execution time by adjusting the LUTs besides applicable to other privacy-preserving systems.

There remains additional challenges that we plan to tackle in future work. For example, using the LUT-based FHE to compute the functions multiple times results in a gradual increase of the calculation error. Similarly, we do not yet have a systematic way to prepare data points in input LUTs for a variety of functions. Therefore, in future we will further optimize the method to systematically control the accuracy and execution time for any functions or sequence of functions. Besides, we will simulate the smart meters using over 100 Raspberry PIs to prepare an actual situation to investigate problems when applied to real-world situations.

Acknowledgment

This work was supported by Japan–US Network Opportunity 2 by Commissioned Research of the National Institute of Information and Communications Technology (NICT), Japan and NSF grants SATC-2030611, SATC-2030624, DGE-1433659, CNS-1818942. Special thanks to Dr. Yu Ishimaki for his help in fruitful discussions.

References

[1] R. R. Mohassel et al., "A survey on Advanced Metering Infrastructure," International Journal of Electrical Power & Energy Systems, vol. 63, pp. 473-484, 2014.

[2] U.S. Department of Energy, "Advanced metering infrastructure and customer system," AMI and Customer Systems: Results from the SGIG Program (2016). https://www.energy.gov/sites/prod/files/2016/12/f34/AMI%20Summary%20Report_09-26-16.pdf.

[3] S. Mujeeb et al., "Electricity Theft Detection with Automatic Labeling and Enhanced RUSBoost Classification using Differential Evolution and Jaya Algorithm," IEEE Access, 9, 128521-128539, 2021.

[4] Y. Peng et al., "Electricity Theft Detection in AMI Based on Clustering and Local Outlier Factor," IEEE Access, 9,107250-107259, 2021.

[5] A. Ullah, et al., "Synthetic Theft Attacks Implementation for Data Balancing and a Gated Recurrent Unit Based Electricity Theft Detection in Smart Grids," Conference on Complex, Intelligent, and Software Intensive Systems, Springer, Cham, pp. 395-405, 2021.

[6] I. U. Khan et al., "Big Data Analytics for Electricity Theft Detection in Smart Grids," 2021 IEEE Madrid PowerTech, pp. 1-6, 2021.

[7] S. Bhattacharjee and S. K. Das, "Detection and Forensics against Stealthy Data Falsification in Smart Metering Infrastructure," IEEE Transactions on Dependable and Secure Computing, 18(2), pp. 356-371, 2021.

[8] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM computing surveys (CSUR), 41(3), pp. 1-58, 2009.

[9] F. Passerini and A. M. Tonello, "Smart Grid Monitoring Using Power Line Modems: Anomaly Detection and Localization," IEEE Transactions on Smart Grid, 10(6), pp. 6178-6186, 2019.

[10] X. Liu and P. S. Nielsen, "Regression-based Online Anomaly Detection for Smart Grid Data," arXiv preprint arXiv:1606.05781, 2016.

[11] H. Karimipour et al., "Intelligent Anomaly Detection for Large-scale Smart Grids," 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), pp. 1-4, 2019.

[12] S. Bhattacharje et al., "Statistical Security Incident Forensics against Data Falsification in Smart Grid Advanced Metering Infrastructure," Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, pp. 35-45, 2017.

[13] I. Siniosoglou et al., "A Unified Deep Learning Anomaly Detection and Classification Approach for Smart Grid Environments," IEEE Transactions on Network and Service Management, 18(2), pp. 1137-1151, 2021.

[14] F. Zheng et al., "Anomaly detection in smart grid based on encoder-decoder framework with recurrent neural network," Journal of China Universities of Posts and Telecommunications, 24(6), pp. 67-73, 2017.

TABLE V
THE COMPUTATION TIME AND COMMUNICATION COST FOR ANOMALY ATTACK DETECTION USING 24 TIME-SLOTS

Party	Operation	Experiment #1			Experiment #2			Experiment #3		
		Runtime [s]	VmSize [MB]	File size [MB]	Runtime [s]	VmSize [MB]	File size [MB]	Runtime [s]	VmSize [MB]	File size [MB]
CS	Key generation	1.949	241.4	201.9	1.949	241.4	201.9	1.949	241.4	201.9
LUT Provider	(a)	1.554	51.0	94.0	0.997	47.6	35.2	0.874	47.0	19.8
Raspberry PI	(b)	2.575	231.6	1.1	2.575	231.6	1.1	2.575	231.6	1.1
CS	(c) step 1	2.243	60.0	7.2	2.243	60.0	7.2	2.243	60.0	7.2
Utility	(c) step 2	0.751	59.3	2.2	0.751	59.3	2.2	0.751	59.3	2.2
CS	(c) step 3	1.563	268.4	15.2	1.553	268.4	4.2	1.535	268.4	1.1
SUM	(c) step 1-3	4.557	–	–	4.546	–	–	4.528	–	–
Utility	(c) step 4	0.787	70.0	2.2	0.720	55.4	2.2	0.698	49.5	2.2
CS	(c) step 5	5.239	4611.1	–	3.179	4385.5	–	2.648	4385.6	–
CS	(c) step 6	0.933	276.9	17.0	0.905	264.9	4.1	0.889	261.9	1.1
Utility	(c) step 7	0.793	69.9	1.1	0.730	52.2	1.1	0.705	49.3	1.1
CS	(c) step 8	4.830	2400.7	0.5	2.339	2400.7	0.5	1.807	2400.7	0.5
SUM	(c) step 1-8	**17.138**	–	–	**12.420**	–	–	**11.274**	–	–

(a) Construction of LUTs, (b) Encryption of inputs, (c) LUT processing

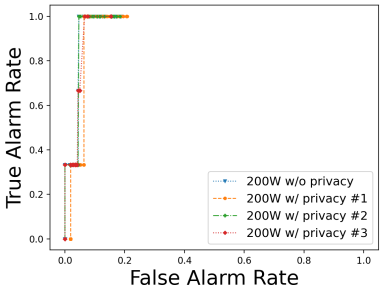

Fig. 3. Deductive(Δ_{avg}=200 w)

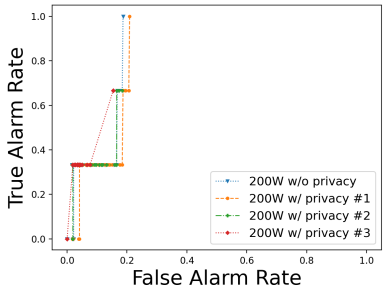

Fig. 4. Camouflage(Δ_{avg}=200 w)

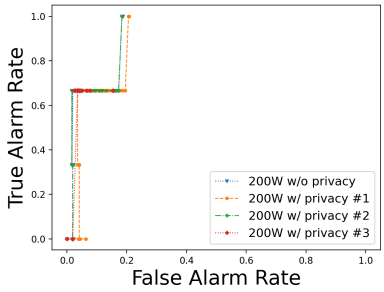

Fig. 5. Additive(Δ_{avg}=200 w)

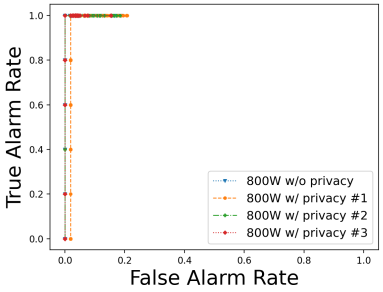

Fig. 6. Deductive(Δ_{avg}=800 w)

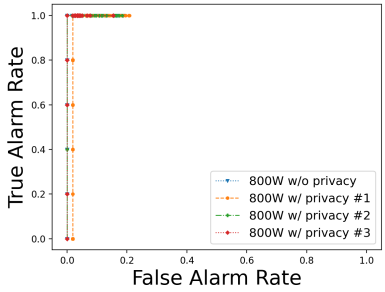

Fig. 7. Camouflage(Δ_{avg}=800 w)

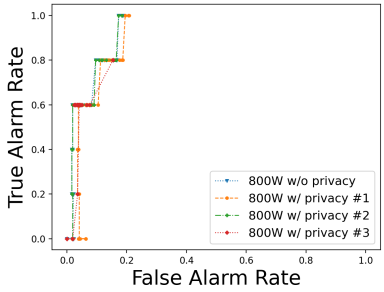

Fig. 8. Additive(Δ_{avg}=800 w)

[15] J. Zhao et al., "Achieving differential privacy of data disclosure in the smart grid," IEEE INFOCOM 2014-IEEE Conference on Computer Communications, IEEE, pp. 504-512, 2014.

[16] G. Danezis et al., "Smart meter aggregation via secret-sharing," Proceedings of the first ACM workshop on Smart energy grid security, pp. 75-80, 2013.

[17] M. A. Mustafa et al., "A secure and privacy-preserving protocol for smart metering operational data collection," IEEE Transactions on Smart Grid, 10(6), pp. 6481-6490, 2019.

[18] F. Li, B. Luo, and P. Liu, "Secure information aggregation for smart grids using homomorphic encryption," 2010 first IEEE international conference on smart grid communications, IEEE, pp. 327-332, 2010.

[19] X. Liang et al., "UDP: Usage-based dynamic pricing with privacy preservation for smart grid," IEEE Transactions on smart grid, 4(1), pp. 141-150, 2013.

[20] Y. Ishimaki et al., "Towards Privacy-preserving Anomaly-based Attack Detection against Data Falsification in Smart Grid," 2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pp. 1-6, 2020.

[21] R. Li, Y. Ishimaki and H. Yamana, "Privacy preserving calculation in cloud using fully homomorphic encryption with table lookup," 2020 5th IEEE International Conference on Big Data Analytics (ICBDA), IEEE, pp. 315-322, 2020.

[22] R.Li and H. Yamana, "Fast and Accurate Function Evaluation with LUT over Integer-based Fully Homomorphic Encryption," International Conference on Advanced Information Networking and Applications, Springer, Cham, pp. 620-633, 2021.

[23] Microsoft Research, Redmond, WA, Microsoft SEAL (release 3.2) (2019). https://github.com/Microsoft/SEAL.

[24] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," IACR Cryptol. ePrint Arch. 2012 144, 2012.

[25] N. P. Smart and F. Vercauteren, "Fully homomorphic SIMD operations," Journal of Designs, Codes and Cryptography, 71(1), pp. 57-81, 2014.

[26] S. Halevi and V. Shoup. "Algorithms in HElib," Annual Cryptology Conference, Springer, Berlin, Heidelberg, pp. 554–571, 2014.