Comments on a Double-Blockchain Assisted Data Aggregation Scheme for Fog-Enabled Smart Grid

Pei-Yu Lin[1], Ya-Fen Chang[2], Pei-Shih Chang[2], and Wei-Liang Tai[3,*]

[1] Department of Electrical Engineering, National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan
[2] Department of Computer Science and Information Engineering, National Taichung University of Science and Technology, Taichung, Taiwan
[3,*] Bachelor Degree Program of Artificial Intelligence, National Taichung University of Science and Technology, Taichung, Taiwan
twl@nutc.edu.tw

Abstract. To comply with the specific requirements of smart grids, Chen et al. proposed a data aggregation scheme by utilizing double blockchains and the Paillier cryptosystem that is an additive homomorphic encryption system for public-key cryptography. Chen et al. claimed that their scheme could resist various attacks and ensure data confidentiality, data integrity, validity, identity anonymity and authenticity. However, after thoroughly analyzing their scheme, we find that it suffers from five flaws. Firstly, anonymity is not ensured as claimed. Secondly, private keys of smart meters and fog nodes can be easily retrieved. Thirdly, after a smart meter or a fog node's private key is revealed, a malicious entity can impersonate it and generate a valid signature of the forged data's ciphertext. Fourthly, in both of the UA-blockchain generation phase and FA-blockchain generation phase, the signature verification will never succeed. Fifthly, some statements in Chen et al.'s scheme are inaccurate or missing such that their scheme cannot work as claimed. The details of how these flaws damage Chen et al.'s scheme are shown in this paper.

Keywords: Blockchain, Smart Grid, Authentication, Homomorphism, Anonymity, Fog Computing, Data Aggregation.

1 Introduction

Smart grids are designed to manage the two-way flow of electricity and information between utilities and users. Digital technologies are used to improve the efficiency, reliability, and security of the grid and to provide users more information and control over their energy usage. For example, smart meters can send users' electricity consumption information to the control center, and the control center can analyze the obtained information to provide users with electricity consumption reports and suggestions to save energy. However, smart grids have special communication and computing requirements such that they need to be deployed widely. In addition, users' electricity consumption information sent by smart meters is often collected and stored by electricity suppliers without the appropriate permission. The data may be accessed by a third party such that users' living habits and related economic status will be revealed. This leads to privacy breaches while users are not aware that their personal data is being collected and shared in this way. That is, how to protect user privacy is an important issue in smart grids. On the other hand, because of the properties of smart grids, a variety of attacks may result in catastrophic damage. Thus, how to ensure the security of smart grids is another essential issue that has to be taken into consideration, and proper security mechanisms need to be employed to protect smart grids from these various attacks such as eavesdropping, tampering, and counterfeiting. Besides, performance is also a key issue to determine whether a smart grid can work and offer desired services well or not. To sum up, performance, privacy, and security are the main issues in smart grids. And data aggregation can be regarded as a representative mechanism in smart grids because of its remarkable advantages. Thus, a plenty of data aggregation schemes preserving privacy are proposed [1-7].

Zhang et al. [8] took advantages of the superior features of blockchain such as non-repudiation, untamperability, decentralization, and easy-to-trace and proposed a keyless signature scheme based on the blockchain architecture for smart grids. In Zhang et al.'s scheme, a new consensus mechanism is designed to turn the blockchain into an automatic access control manager such that no trusted third party is needed. Because blockchain possesses superior features, several researches are proposed from then on. In 2020, Alcaraz et al. [9] proposed a smart grid structure by using the three-layer-based interconnection architecture and blockchain technology to manage connections among devices, resources, and processes while ensuring reliability and security. Li et al. [10] proposed a blockchain-based anomalous electricity consumption detection method for smart grids. In Li et al.'s method, electricity consumption data is from readings of sensors and smart meters, a trained machine learning model is adopted to detect electricity consumption anomalies, and the blockchain is used to record all processes.

Recently, Chen et al. [11] proposed a data aggregation scheme for smart grids with a double-blockchain structure, which is called the double-blockchain-assisted secure and anonymous data aggregation scheme, DA-SADA. DA-SADA presents a network model with three layers, user layer, fog computing layer, and service supporting layer. And, there are two types of blockchains, UA-blockchain and FA-blockchain. In the user layer, the whole area is divided into several subareas, and smart meters *SM*'s are deployed to collect users' electricity consumption information. In a subarea, data collected by smart meters is encrypted and sent to a specific smart meter, namely an aggregation node in the user layer. An aggregation node in the user layer is responsible for aggregating data, generating the new block in UA-blockchain and sending the generated UA-blockchain to the subarea's corresponding fog node in the fog computing layer. In the fog computing layer, when a fog node receives information, it generate the corresponding digital signature and sends required data to a specific node, namely an aggregation node in the fog computing layer. The aggregation node in the fog computing layer is responsible for aggregating data, generating the new block in FA-blockchain and sending the generated FA-blockchain to the cloud server in the service supporting layer. When the cloud server receives the FA-blockchain, it can retrieve the needed information and make analysis to further determine strategies and improve the power utilization efficiency.

However, after thoroughly analyzing their scheme, DA-SADA, we find that it suffers from five flaws. Firstly, anonymity is not ensured as claimed. Secondly, private keys of smart meters and fog nodes can be easily retrieved. Thirdly, after a smart meter or a fog node's private key is revealed, a malicious entity can impersonate it and generate a valid signature of the forged data's ciphertext. Fourthly, in both of the UA-blockchain generation phase and FA-blockchain generation phase, the signature verification will never succeed. Fifthly, some statements in DA-SADA are inaccurate or missing such that it cannot work as claimed.

The rest of this paper is organized as follows: Section 2 reviews DA-SADA. The security analysis of DA-SADA is made in Section 3. At last, some conclusions are made in Section 4.

2 Review of DA-SADA

In this section, we review the double-blockchain assisted secure and anonymous data aggregation scheme, DA-SADA, proposed by Chen et al. DA-SADA consists of four phases, system initialization phase, UA-blockchain generation phase, FA-blockchain generation phase, and service supporting phase. Notations commonly used in DA-SADA are listed in Table 1, and the details are as follows.

Table 1. Notations used in DA-SADA.

Notation	Definition
TA	a trust authority that generates parameters for all devices
p, q	two large prime numbers
κ	a system security parameter denotes the length of prime numbers
N/λ	the system public/ private key
SM_{ij}	the i-th smart meter in the j-th subarea
X_{ij}	SM_{ij}'s public key
Y_{ij}	SM_{ij}'s private key
$Pseu_{ij}$	SM_{ij}'s pseudonym
fog_j	the fog node responsible for the j-th subarea in the user layer
X_j	fog_j's public key
Y_j	fog_j's private key
$H(.)$	a cryptographic hash function
$\|\|$	the concatenation operator

2.1 System Initialization Phase

In DA-SADA, TA is responsible for system initialization. First, TA generates all system parameters including the system's public and private keys, all smart meters and fog nodes' public and private keys, and pseudonyms for all smart meters and fog nodes. Then, *TA* distributes system parameters. At last, *TA* generates Bloom filters for all subareas and a Bloom filter in the fog computing layer. The details are as follows:

Step 1. *TA* selects the security parameter κ and two prime numbers p and q of length κ bits.

Step 2. *TA* computes the system public key $N = p \times q$ and the system private key $\lambda = lcm(p\text{-}1, q\text{-}1)$ for the homomorphic encryption algorithm.

Step 3. *TA* chooses a number $r \in \mathbb{Z}_N^*$ randomly, computes $s = r^N \bmod N^2$, and defines a function $L(u) = \frac{u-1}{N}$, where u denotes the input of the function $L(.)$.

Step 4. For each smart meter SM_{ij}, *TA* randomly chooses a prime number X_{ij} as SM_{ij}'s public key and computes SM_{ij}'s private key $Y_{ij} = X_{ij}^{-1} \bmod N^2$ and SM_{ij}'s pseudonym $Pseu_{ij} = X_{ij} \bmod N^2$.

Step 5. For each fog node fog_j, *TA* randomly chooses a prime number X_j as fog_j's public key and computes fog_j's private key $Y_j = X_j^{-1} \bmod N^2$ and fog_j's pseudonym $Pseu_j = X_j \bmod N^2$.

Step 6. *TA* chooses a cryptographic hash function $H(.)$: $\{0, 1\}^* \rightarrow Z_N^*$.

Step 7. After λ, N, s, $H(.)$, X_{ij}, Y_{ij}, X_j, and Y_j are generated, N and $H(.)$ are published online while (X_{ij}, Y_{ij}, s), (X_j, Y_j), and λ are assigned to SM_{ij}, fog_j, and the cloud server through a secure channel, respectively.

Step 8. For the j-th subarea, *TA* collects pseudonyms of all corresponding smart meters and generates a Bloom filter that is a θ-bit array and the element's value is set to one when its index is equal to $H(Pseu_{ij}) \bmod \theta$. Similarly, *TA* collects pseudonyms of fog nodes to generate a Bloom filter in the fog computing layer.

Step 9. At last, *TA* sends Bloom filters generated for subareas to the corresponding smart meters and the Bloom filter in the fog computing layer to fog nodes.

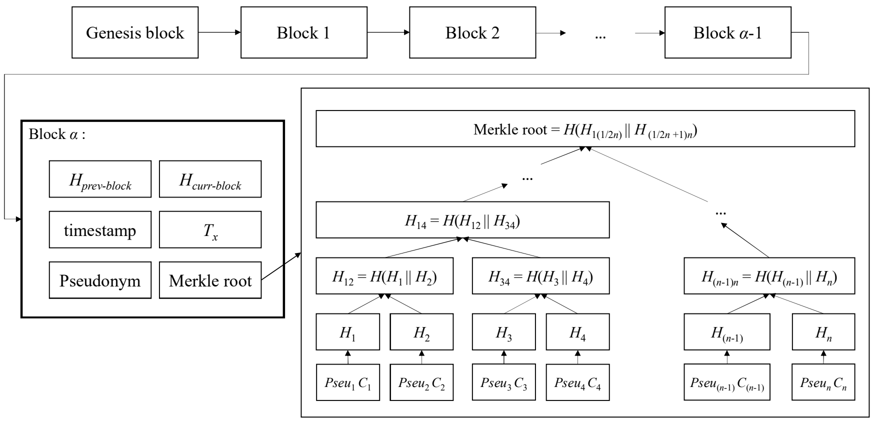

Fig. 1. The structure of UA-blockchain and how the Merkle root is generated in DA-SADA.

2.2 UA-Blockchain Generation Phase

After the system is initialized, each smart meter encrypts a user's power consumption data and generates the corresponding signature for integrity. Then, one of smart meters in the same subarea is chosen as the aggregation node in the user layer because it possesses the most remaining computational resources. The aggregation node collects reports of smart meters, verifies signatures, obtains the aggregated ciphertext, and generates the transaction, where the aggregation node and these smart meters are in the same subarea. The aggregation node records the transaction in a block and broadcasts this block in the subarea for authentication. When the number of positive verification results reaches the threshold, the new block is added to the UA-blockchain in the user layer. Fig. 1 depicts the structure of the UA-blockchain and how the corresponding Merkle root is generated. The details of this phase are as follows:

Step 1. In the j-th subarea in a certain time slot t_s, SM_{ij} first sets a parameter $g = N + 1$ and encrypts the collected data d_{ij} by computing $C_{ij} = (1 + d_{ij}N) \times s \bmod N^2$ instead of $g^{d_{ij}} \times r^N \bmod N^2$ to improve computation efficiency, where $C_{ij} = g^{d_{ij}} \times r^N \bmod N^2 = (N+1)^{d_{ij}} \times r^N \bmod N^2 = (1 + d_{ij}N) \times s \bmod N^2$, i is in $[1, n]$ and n denotes the number of smart meters in the j-th subarea.

Step 2. SM_{ij} uses the encrypted data C_{ij}, its pseudonym $Pseu_{ij}$, its private key Y_{ij}, and the timeslot t_s to generate the signature $\sigma_{ij} = H(u_{ij}||Pseu_{ij})^{Y_{ij}} \bmod N^2$, where $u_{ij} = H(C_{ij} \,||\, t_s)$.

Step 3. SM_{ij} sends the report $\{Pseu_{ij}, t_s, C_{ij}, \sigma_{ij}\}$ to the aggregation node.

Step 4. After receiving the report $\{Pseu_{ij}, t_s, C_{ij}, \sigma_{ij}\}$, the aggregation node checks the effectiveness of $Pseu_{ij}$ with the Bloom filter and the validity of the report with the timestamp.

Step 5. The aggregation node uses batch verification to verify these received signatures by checking if $\prod_{i=1}^{n} \sigma_{ij}^{X_{ij}} \bmod N^2 = \prod_{i=1}^{n} H\,(H(C_{ij}||t_s)||Pseu_{ij}) \bmod N^2$.

Step 6. If it holds, the aggregation node obtains the aggregated ciphertext C_j for the j-th subarea by computing $C_j = \prod_{i=1}^{n} C_{ij} \bmod N^2$ and generates the transaction $T_x = (C_j, Pseu_{ij}, t_s)$.

Step 7. Then the aggregation node records the transaction $T_x = (C_j, Pseu_{ij}, t_s)$ in a new block that also includes the Merkle root, the hash value of the previous block $H_{prev\text{-}block}$, and the hash value of the current block $H_{curr\text{-}block}$, where the Merkle root is obtained by setting a leaf node's value with the ciphertext and the pseudonym and hashing them and the corresponding hash results as shown in Figure 4, and $H_{curr\text{-}block}$ = SHA256($index + H_{prev\text{-}block} + Pseu_{ij} + timestamp + C_j + \sum_{ij} transactions_{ij}$).

Step 8. After generating the new block, the aggregation node broadcasts it in the j-th subarea. Then, each smart meter SM_{ij}, an ordinary node, verifies records in the new block and its related data only by checking whether it is identical to the original data or not. If the verification is successful, the smart meter SM_{ij}, an ordinary node, broadcasts the verification result in the j-th subarea.

Step 9. If the number of the correctness confirmation messages sent by other distinct smart meters in the j-th subarea is equal to or more than $2n/3+1$, the new block is considered to be valid and added to the UA-blockchain, where n denotes the number of smart meters in the j-th subarea.

2.3 FA-Blockchain Generation Phase

The process to generate the FA-blockchain is similar to that to generate the UA-blockchain. In the FA-blockchain generation phase, each fog node first receives encrypted data from the corresponding UA-blockchain and generates the corresponding signature of the encrypted data for integrity. Then, one of fog nodes is chosen as the aggregation node in the fog computing layer because it possesses the most remaining computational resources. The aggregation node in the fog computing layer collects reports of fogs, verifies signatures, obtains the aggregated ciphertext for all subareas, and generates the transaction. The aggregation node in the fog computing layer records the transaction in a block and broadcasts this block to other fog nodes for authentication. When the number of positive verification results reaches the threshold, the new block is added to the FA-blockchain in the fog computing layer. The details of the FA-blockchain generation phase are as follows:

Step 1. When fog_j that is responsible for the j-th subarea gets the aggregated power consumption ciphertext C_j, fog_j uses the encrypted data C_j, its pseudonym $Pseu_j$, its private key Y_j, and the timeslot t_s to generate the signature $\sigma_j = H(u_j||Pseu_j)^{Y_j} \bmod N^2$, where j is in $[1, m]$, m denotes the number of fog nodes in the fog computing layer and $u_j = H(C_j \,||\, t_s)$.

Step 2. fog_j sends the report $\{Pseu_j, t_s, C_j, \sigma_j\}$ to the aggregation node in the fog computing layer.

Step 3. After receiving the report $\{Pseu_j, t_s, C_j, \sigma_j\}$, the aggregation node in the fog computing layer checks the effectiveness of $Pseu_j$ with the Bloom filter and the validity of the report with the timestamp.

Step 4. The aggregation node in the fog computing layer uses batch verification to verify these received signatures by checking if $\prod_{j=1}^{m} \sigma_j^{X_j} \bmod N^2 = \prod_{j=1}^{m} H\,(H(C_j||t_s)||Pseu_j) \bmod N^2$.

Step 5. If it holds, the aggregation node in the fog computing layer obtains the aggregated ciphertext C_{AS} for all subareas by computing $C_{AS} = \prod_{j=1}^{m} C_j \bmod N^2$ and generates the transaction $T_x' = (C_{AS}, Pseu_j, t_s)$.

Step 6. Then the aggregation node in the fog computing layer records the transaction $T_x' = (C_{AS}, Pseu_j, t_s)$ in a new block that also includes the Merkle root, the hash value of the previous block $H_{prev\text{-}block}'$, and the hash value of the current block $H_{curr\text{-}block}'$, where $H_{curr\text{-}block}' = \text{SHA256}(index + H_{prev\text{-}block}' + Pseu_j + timestamp + C_{AS} + \sum_j transactions_j)$.

Step 7. After generating the new block, the aggregation node in the fog computing layer broadcasts it to other fog nodes. Then, each fog node fog_j, an ordinary node in the fog computing layer, verifies records in the new block and verifies its related data only by checking whether it is identical to the original data or not. If the verification is successful, the fog node fog_j, an ordinary node in the fog computing layer, broadcasts the verification result to other fog nodes in the fog computing layer.

Step 8. If the number of the correctness confirmation messages sent by other distinct fog nodes in the fog computing layer is equal to or more than $2m/3+1$, the new block is considered to be valid and added to the FA-blockchain, where m denotes the number of fog nodes in the fog computing layer.

2.4 Service Supporting Phase

When the cloud server receives the FA-blockchain of the fog computing layer, it gets the aggregated power consumption ciphertext C_{AS} for all subareas and decrypts it to get the aggregated plaintext $M = L(C_{AS}{}^{\lambda} \bmod N^2) / L(g^{\lambda} \bmod N^2)$ by using the decryption procedure mentioned in the Paillier cryptosystem. Then the cloud server recovers subareas' data UA_j's with their proposed Horner rule-based analytical algorithm, where $UA_j = \sum_{i=1}^{n} d_{ij}$ and $M = \sum_{j=1}^{m} UA_j$. When the cloud server obtains the power consumption of each subarea, the future power usage of each subarea can be predicted, and decision support for power dispatch and price adjustment can be provided.

3 Security Analysis of DA-SADA

Chen et al. claimed that DA-SADA could resist various attacks and ensure data confidentiality, data integrity, validity, identity anonymity and authenticity. However, after thoroughly analyzing DA-SADA, we find that it suffers from five flaws. Firstly, anonymity is not ensured as claimed because a smart meter's pseudonym is fixed. Secondly, private keys of smart meters and fog nodes can be easily retrieved. Thirdly, after a smart meter or a fog node's private key is revealed, a malicious entity can impersonate it and generate a valid signature of the forged data's ciphertext. Fourthly, in both of the UA-blockchain generation phase and FA-blockchain generation phase, the signature verification will never succeed such that legal signatures are always regarded as invalid. Fifthly, some statements in DA-SADA are inaccurate or missing such that DA-SADA cannot work as claimed. The details are as follows:

3.1 Failure to Ensure Anonymity

In the system initialization phase, TA randomly chooses a prime number X_{ij} as the smart meter SM_{ij}'s public key and computes SM_{ij}'s private key $Y_{ij} = X_{ij}^{-1} \bmod N^2$ and SM_{ij}'s pseudonym $Pseu_{ij} = X_{ij} \bmod N^2$. And, TA randomly chooses a prime number X_j as the fog node fog_j's public key and computes fog_j's private key $Y_j = X_j^{-1} \bmod N^2$ and fog_j's pseudonym $Pseu_j = X_j \bmod N^2$. Each smart meter SM_{ij}'s pseudonym $Pseu_{ij}$ and each fog node fog_j's pseudonym $Pseu_j$ are always fixed because they are not updated in other phases. Moreover, these pseudonyms are not concealed when transmitted or included in blocks. Thus, a specific smart meter or fog node will be traced or monitored. According to the above, anonymity is not ensured in Chen et al.'s scheme.

3.2 Disclosure of the Private Key

In the system initialization phase, a smart meter SM_{ij} is assigned with the private key Y_{ij} and the public key X_{ij}, where $Y_{ij} = X_{ij}^{-1}$ mod N^2 and SM_{ij}'s pseudonym $Pseu_{ij} = X_{ij}$ mod N^2. SM_{ij}'s pseudonym $Pseu_{ij}$ will be transmitted in the UA-blockchain generation phase or included in blocks of the UA-blockchain, so $Pseu_{ij}$ can be easily obtained. In addition, the parameter N is published online. Thereupon, SM_{ij}'s private key Y_{ij} can be retrieved by computing $Y_{ij} = Pseu_{ij}^{-1}$ mod $N^2 = X_{ij}^{-1}$ mod N^2. Similarly, a fog node fog_j is assigned with the private key Y_j and the public key X_j, where $Y_j = X_j^{-1}$ mod N^2 and fog_j's pseudonym $Pseu_j = X_j$ mod N^2. Because fog_j's pseudonym $Pseu_j$ will be transmitted in the FA-blockchain generation phase or included in blocks of the FA-blockchain, $Pseu_j$ can be easily obtained. Thereupon, fog_j's private key Y_j can be retrieved by computing $Y_j = Pseu_j^{-1}$ mod $N^2 = X_j^{-1}$ mod N^2. As a result, private keys of smart meters and fog nodes can be easily retrieved in Chen et al.'s scheme.

3.3 Generation of a Valid Signature of the Forged Data's Ciphertext

After a smart meter's private key is revealed, a malicious entity can impersonate it and generate a valid signature of the forged data's ciphertext. The details are as follows. In the system initialization phase, a smart meter SM_{ij} is assigned with the private key Y_{ij}, the public key X_{ij}, and s securely, where $s = r^N$ mod N^2 and $r \in \mathbb{Z}_N^*$. In the UA-blockchain generation phase, SM_{ij} computes the ciphertext C_{ij} of the data d_{ij} by computing $C_{ij} = (1 + d_{ij}N) \times s$ mod N^2. Actually, the Paillier homomorphic cryptosystem allows a user to compute his/her personal ciphertext with an arbitrary random number, and the receiver who gets the aggregated ciphertext can retrieve the aggregated data without knowing what the involved random numbers are. That is, the malicious entity can chooses a random number $r_{ij} \in \mathbb{Z}_N^*$, generates the forged data d_{ij}', and computes the corresponding ciphertext C_{ij}' by computing $C_{ij}' = g^{d_{ij}'} \times r_{ij}{}^N$ mod N^2. After retrieving a smart meter SM_{ij}'s private key Y_{ij} by computing $Y_{ij} = Pseu_{ij}^{-1}$ mod N^2, the malicious entity computes $u_{ij}' = H(C_{ij}' || t_s)$ and the signature $\sigma_{ij}' = H(u_{ij}' || Pseu_{ij})^{Y_{ij}}$ mod N^2 and sends the report $\{Pseu_{ij}, t_s, C_{ij}', \sigma_{ij}'\}$ to the aggregation node. Thus, after retrieving a smart meter SM_{ij}'s private key Y_{ij}, a malicious entity can impersonate SM_{ij} and further generate a valid signature of the forged data's ciphertext. Similarly, because a fog node fog_j's private key Y_j can be easily retrieved, a malicious entity can impersonate fog_j and generate a valid signature of the forged data's ciphertext as well.

3.4 Failed Signature Verification

In both of the UA-blockchain generation phase and FA-blockchain generation phase, an aggregation node verifies signatures with batch verification. In the UA-blockchain generation phase, the aggregation node in the user layer verifies these received signatures σ_{ij}'s by checking if $\prod_{i=1}^{n} \sigma_{ij}^{X_{ij}}$ mod $N^2 = \prod_{i=1}^{n} H\,(H(C_{ij}||t_s)||Pseu_{ij})$ mod N^2. That is, the equality of $\sigma_{ij}^{X_{ij}}$ mod N^2 and $H(H(C_{ij} \,||\, t_s) \,||\, Pseu_{ij})$ mod N^2 is checked with a batch approach, where i is in $[1, n]$ and n denotes the number of smart meters in the

j-th subarea. Because $\sigma_{ij} = H(u_{ij}||Pseu_{ij})^{Y_{ij}}$ mod N^2 and $u_{ij} = H(C_{ij} \,||\, t_s)$, $\sigma_{ij}^{X_{ij}}$mod N^2 $= H(H(C_{ij}||t_s)||Pseu_{ij})^{Y_{ij}X_{ij}}$ mod N^2. Unfortunately, $\sigma_{ij}^{X_{ij}}$ mod $N^2 \neq H(H(C_{ij} \,||\, t_s) \,||\, Pseu_{ij})$ mod N^2. In the system initialization phase, the smart meter SM_{ij} is assigned with the private key Y_{ij} and the public key X_{ij}, where $Y_{ij} = X_{ij}^{-1}$ mod N^2 and SM_{ij}'s pseudonym $Pseu_{ij} = X_{ij}$ mod N^2. To show that $\sigma_{ij}^{X_{ij}}$mod $N^2 \neq H(H(C_{ij} \,||\, t_s) \,||\, Pseu_{ij})$ mod N^2, an example is given. First, we set $N = 3 \times 5$ and randomly select a prime number $X_{ij} = 227$, the private key $Y_{ij} = 227^{-1}$ mod 225 = 113. Suppose $H(u_{ij} \,||\, Pseu_{ij}) = H(H(C_{ij} \,||\, t_s) \,||\, Pseu_{ij}) = 2$. Then the signature $\sigma_{ij} = H(u_{ij}||Pseu_{ij})^{Y_{ij}}$mod $N^2 = 2^{113}$ mod 225 = 109, and $\sigma_{ij}^{X_{ij}}$mod $N^2 = 109^{227}$ mod 225 = 121 $\neq H(H(C_{ij} \,||\, t_s) \,||\, Pseu_{ij})$. As a result, valid signatures will be never verified successfully in the UA-blockchain generation phase. Similarly, in the FA-blockchain generation phase, the aggregation node in the fog computing layer verifies these received signatures σ_j's by checking if $\prod_{j=1}^{m} \sigma_j^{X_j}$ mod $N^2 = \prod_{j=1}^{m} H\,(H(C_j||t_s)||Pseu_j)$ mod N^2. That is, the equality of $\sigma_j^{X_j}$ mod N^2 and $H(H(C_j \,||\, t_s) \,||\, Pseu_j)$ mod N^2 is checked with a batch approach, where j is in [1, m] and m denotes the number of fog nodes in the fog computing layer. Because $\sigma_j = H\,(u_j||Pseu_j)^{Y_j}$ mod N^2 and $u_j = H(C_j \,||\, t_s)$, $\sigma_j^{X_j}$ mod $N^2 = H(H(C_j||t_s)||Pseu_j)^{Y_jX_j}$ mod N^2. Unfortunately, $\sigma_j^{X_j}$mod $N^2 \neq H(H(C_j \,||\, t_s) \,||\, Pseu_j)$ mod N^2 because the fog node foj_j is assigned with the private key Y_j and the public key X_j, where $Y_j = X_j^{-1}$ mod N^2. As a result, valid signatures will be never verified successfully in the FA-blockchain generation phase.

3.5 Inaccurate and Missing Statements

Some statements in the proposed scheme are inaccurate or missing such that Chen et al.'s scheme cannot work as claimed. The details are as follows.

How to Obtain the Related Public Keys. In the system initialization phase, after TA generates λ, N, s, $H(.)$, X_{ij}, Y_{ij}, X_j, and Y_j, N and $H(.)$ are published online while (X_{ij}, Y_{ij}, s), (X_j, Y_j), and λ are assigned to SM_{ij}, fog_j, and the cloud server through a secure channel, respectively. In the UA-blockchain generation phase, the aggregation node in the user layer needs each smart meter SM_{ij}'s public key X_{ij} to verify the received signatures σ_{ij}'s by checking if $\prod_{i=1}^{n} \sigma_{ij}^{X_{ij}}$ mod $N^2 = \prod_{i=1}^{n} H\,(H(C_{ij}||t_s)||Pseu_{ij})$ mod N^2. In the FA-blockchain generation phase, the aggregation node in the fog computing layer needs each fog node fog_j's public key X_j to verify the received signatures σ_j's by checking if $\prod_{j=1}^{m} \sigma_j^{X_j}$ mod $N^2 = \prod_{j=1}^{m} H\,(H(C_j||t_s)||Pseu_j)$ mod N^2. However, how these aggregation nodes obtain the related public keys X_{ij}'s and X_j's is missing.

Inaccurate Block Structure. Aggregation nodes in the user layer and fog computing layer store the transactions in blocks of the UA-blockchain and FA-blockchain, respectively. In the UA-blockchain generation phase, the aggregation node in the user layer generates the transaction T_x = (C_j, $Pseu_{ij}$, t_s) and records the transaction (C_j, $Pseu_{ij}$, t_s) in a new block that also includes the Merkle root, the hash value of the previous block $H_{prev\text{-}block}$, and the hash value of the current block $H_{curr\text{-}block}$, where the Merkle root is obtained by setting a leaf node's value with the ciphertext and the pseudonym and hashing them and the corresponding hash results as shown in Figure 4, and $H_{curr\text{-}block}$ = SHA256($index$ + $H_{prev\text{-}block}$ + $Pseu_{ij}$ + $timestamp$ + C_j + $\sum_{ij} transactions_{ij}$). And, the aggregation node broadcasts the new block in the j-th subarea. Then, each smart meter SM_{ij}, an ordinary node, verifies records in the new block and verifies its related data only by checking whether it is identical to the original data or not. If the verification is successful, SM_{ij} broadcasts the verification result in the j-th subarea.When the number of the correctness confirmation messages sent by other distinct smart meters in the j-th subarea is equal to or more than $2n/3+1$, the new block is considered to be valid and added to the UA-blockchain. However, parameters or symbols $index$, $Pseu_{ij}$, $timestamp$, $\sum_{ij} transactions_{ij}$, and "+" are not defined accurately. And, only one transaction T_x = (C_j, $Pseu_{ij}$, t_s) is recorded in the new block while all involved C_{ij}'s are absent. This approach makes it impossible for each smart meter SM_{ij}, an ordinary node, to verify its related data only by checking whether it is identical to the original data or not. Moreover, SM_{ij}, an ordinary node, cannot verify whether C_{ij} is indeed aggregated to generate C_j. Thus, as shown in Fig. 2, a block should be modified to consist of a block header and a block body. The block head should include the sequence number of the block $index$, all involved transactions (C_{ij}, $Pseu_{ij}$, t_s)'s, the pseudonym of the aggregation node in the user layer, the aggregated ciphertext C_j, the Merkle root, the hash value of the previous block $H_{prev\text{-}block}$, and the hash value of the current block $H_{curr\text{-}block}$, where $H_{curr\text{-}block}$= SHA256($index$ || $H_{prev\text{-}block}$ || the pseudonym of the aggregation node in the user layer || t_s || C_j || transactions || the Merkle root) and the Merkle root is obtained by setting a leaf node's value with the ciphertext C_{ij} and the corresponding pseudonym $Pseu_{ij}$ and hashing them to get the corresponding hash results hierarchically. The corresponding Merkle tree is stored in the block body.

On the other hand, the similar problems will be encountered in the FA-blockchain generation phase. To overcome these problems, some modifications should be made. For simplicity, the differences between blocks of the UA-blockchian and those of the FA-blockchain are listed. First, transactions (C_j, $Pseu_j$, t_s)'s, the pseudonym of the aggregation node in the fog computing layer and the aggregated ciphertext C_{AS} instead of (C_{ij}, $Pseu_{ij}$, t_s)'s, the pseudonym of the aggregation node in the user layer and C_j are stored in the block header of the FA-blockchain. Second, $H_{curr\text{-}block}$= SHA256($index$ || $H_{prev\text{-}block}$ || the pseudonym of the aggregation node in the fog computing layer || t_s || C_{AS} || transactions || the Merkle root), and the Merkle root is obtained by setting a leaf node's value with the ciphertext C_j, the corresponding pseudonym $Pseu_j$, and the time slot t_s, and hashing them to get the corresponding hash results hierarchically.

Failure to Retrieve Subareas' Data. When the cloud server receives the FA-blockchain of the fog computing layer, it gets the aggregated power consumption ciphertext C_{AS} for all subareas and decrypts it by using the Paillier homomorphic decryption algorithm to get the aggregated plaintext $M = L(C_{AS}^{\lambda} \bmod N^2) / L(g^{\lambda} \bmod N^2)$. Chen et al.'s proposed a Horner rule-based analytical algorithm and attempted to recovers subareas' data UA_j's, where $UA_j = \sum_{i=1}^{n} d_{ij}$ and $M = \sum_{j=1}^{m} UA_j$. However, the theoretical derivations to show why their proposed Horner rule-based analytical algorithm can recovers subareas' data are not correct. Meanwhile, the designed algorithm cannot work, either. In their designed algorithm, UA_j is computed with a modulus R that a product of all random numbers r_j's. This makes UA_j is in [0, R] while the range of UA_j should be in [0, N^2-1]. On the other hand, in the FA-blockchain generation phase, the aggregation node in the fog computing layer generates the transaction T_x = (C_{AS}, $Pseu_j$, t_s) and records the transaction (C_{AS}, $Pseu_j$, t_s) in a new block. Because only one transaction T_x = (C_{AS}, $Pseu_j$, t_s) is recorded in the new block while all involved C_j's are absent in the FA-blockchain. Thus, it is impossible for the cloud server to retrieves subareas' data UA_j's because C_j's are unknown.

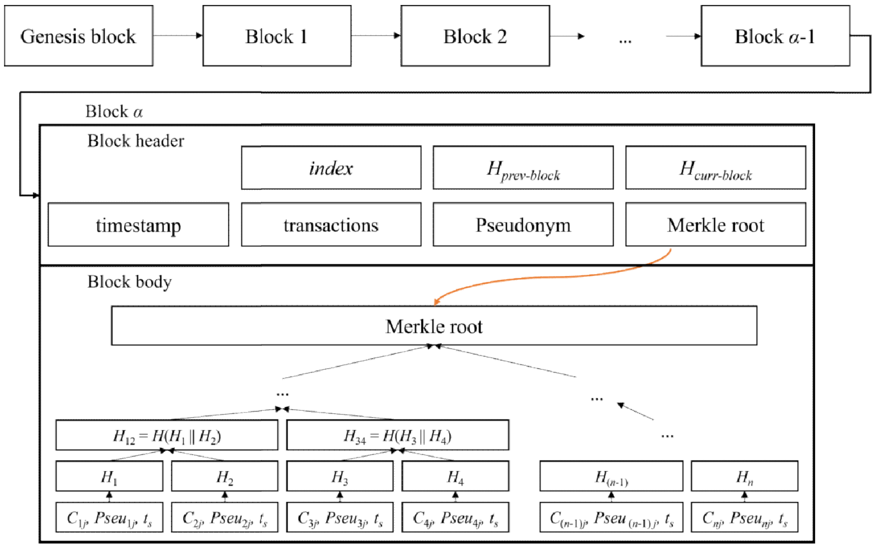

Fig. 2. The structure of the modified blockchain.

4 Conclusions

e specific requirements of smart grids. They claimed that their scheme could defend against various attacks and guarantee data confidentiality, data integrity, validity, anonymity of identity, and authenticity. However, after thoroughly analyzing their scheme, we find that it suffers from five flaws. Firstly, because the pseudonyms of the smart meters and fog nodes are fixed, anonymity is not guaranteed as claimed. Secondly, private keys of smart meters and fog nodes can be easily obtained. Thirdly, after a smart meter or fog node's private key is revealed, a malicious entity can impersonate it and generate a valid signature of the forged data's ciphertext. Fourthly, in both of the UA-blockchain generation phase and FA-blockchain generation phase, the signature verification will never succeed such that legal signatures are always regarded as invalid. Fifthly, in Chen et al.'s scheme, how to obtain the related public keys is absent, the block structure is inaccurate, and the cloud server cannot retrieve subareas' data. Due to the above analysis, proper modification is needed; otherwise, Chen et al.'s scheme can neither work nor preserve the claimed superior properties.

Acknowledgement

This work was supported in part by National Science and Technology Council under the Grants MOST 110-2221-E-992-097-MY3, MOST 110-2221-E-025-012-, MOST 111-2221-E-025-007-, and MOST 110-2221-E-025-014-MY2.

References

1. Lu, R., Liang, X., Li, X., Lin, X., Shen, X.: An Efficient and Privacy-Preserving Aggregation Scheme for Secure Smart Grid Communications. IEEE Transactions on Parallel and Distributed Systems 23(9), 1621–1631 (2012).
2. Jia, W., Zhu, H., Cao, Z., Dong, X., Xiao, C.: Human-Factor-Aware Privacy-Preserving Aggregation in Smart Grid. IEEE Systems Journal 8(2), 598–607 (2014).
3. Liang, K., Susilo, W.: Searchable Attribute-Based Mechanism With Efficient Data Sharing for Secure Cloud Storage. IEEE Transactions on Information Forensics and Security 10, 1981–1992 (2015).
4. Wan, Z., Zhu, W. T., Wang, G.: PRAC: Efficient Privacy Protection for Vehicle-to-Grid Communications in the Smart Grid. Computers & Security 62, 246–256 (2016).
5. Lu, R., Heung, K., Lashkari, A. H., Ghorbani, A. A.: A Lightweight Privacy-Preserving Data Aggregation Scheme for Fog Computing-Enhanced IoT. IEEE Access 5, 3302–3312 (2017).
6. Mahmood, K., Li, X., Chaudhry, S. A., Naqvi, H., Kumari, S., Sangaiah, A. K., Rodrigues, J.J.P.C.: Pairing Based Anonymous and Secure Key Agreement Protocol for Smart Grid Edge Computing Infrastructure. Future Generation Computer Systems 88, 491–500 (2018).
7. Eltayieb, N., Elhabob, R., Hassan, A., Li, F.: An Efficient Attribute-Based Online/Offline Searchable Encryption and Its Application in Cloud-Based Reliable Smart Grid. Journal of Systems Architecture 98, 165–172 (2019).
8. Zhang, H., Wang, J., Ding, Y.: Blockchain-Based Decentralized and Secure Keyless Signature Scheme for Smart Grid. Energy 180, 955–967 (2019).
9. Alcaraz, C., Rubio, J. E., Lopez, J.: Blockchain-Assisted Access for Federated Smart Grid Domains: Coupling and Features. Journal of Parallel and Distributed Computing 144, 124–135 (2020).
10. Li, M., Zhang, K., Liu, J., Gong, H., Zhang, Z.: Blockchain-Based Anomaly Detection of Electricity Consumption in Smart Grids. Pattern Recognition Letters 138, 476–482 (2020).
11. Chen, S., Yang, L., Zhao, C., Varadarajan, V., Wang, K. Double-Blockchain Assisted Secure and Anonymous Data Aggregation for Fog-Enabled Smart Grid. Engineering 8, 159–169 (2020).