Efficient, Reliable and Fault-Diagnosable Authentication Scheme for Smart Grids

Post IEEE PES ISGT12 paper

Abstract—Authentication is a crucial requirement for smart grid communications. However, the current authentication approaches in the smart grid relying on per-packet signature and per-signature verification schemes introduce heavy costs for computation and communication. Furthermore, despite being much desired by the electric power industry, there are still no efficient fault diagnosis tools to support the smart grid’s authentication management yet. Thus, currently, the smart grid’s authentication requirements cannot be satisfied due to its very nature of having limited computation/communication resources in its component devices and of placing a high priority on fault protection and diagnosis.

In this paper, we present an efficient yet reliable scheme to authenticate delay-tolerant smart metering data in the smart grid communications. Cryptographic technologies like batch verification, signature amortization and signature aggregation are adapted to reduce both the number of public key operations and communication overheads. We further contribute two efficient fault diagnosis algorithms to pinpoint forged signatures if any. We have implemented a software package to accomplish our proposed authentication scheme and faults diagnosis algorithms (http://www.bigupload.com/en/file/33615/authsg.zip.html). Experimental results and performance analyses show that our scheme is able to provide significant gains in performance.

Index Terms—Authentication, batch verification, digital signature, fault tolerance, fault diagnosis, smart grid.

I. INTRODUCTION

The smart grid, or the intelligent electricity grid that utilize modern IT/communication/control technologies, become a global trend nowadays. It is a network of smart devices (like smart meters, intelligent electronic devices, etc.) and power infrastructure [10], [19]. The goals of the smart grid are to satisfy power demands in real-time, to optimally transmit and distribute electricity from suppliers to consumers, and to automatically monitor the power status, etc. [35]. Substantial benefits, including improved energy efficiency, promoted power reliability, renewable energy and decreased carbon emission, can be provided by the smart grid [27].

A. Motivations

In addition to the traditional components for power transmission and distribution, the smart grid also involves new components such as smart meters, two-way communication networks, and smart monitoring and control systems. This inevitably introduces new security risks related to data collection, data transportation, system monitoring, automation control, etc. For instance, smart grid communication systems, e.g., Zigbee [36], Wi-Fi alliance, etc., are vulnerable to forgeries and unauthorized modification of smart data (e.g., meter readings) as well as other attacks during the data packets’ transmitting in the air.

Smart data such as power consumption data, statuses, alarms, and events are generated from smart ends like smart meters and other various smart devise across the grid. These data are broadly utilized for the purposes of diagnosis, measurement, control, billing, troubleshooting, etc. Advanced Metering Infrastructure (AMI) is one of the most popular technologies to collect and analyze status, consumption, and diagnostic messages from smart meters.

As depicted above, AMI is composed of smart meters, data concentrators, and the central system (AMI Headend) of the utility company. Smart meters record the customers’ power usage data, statuses, etc. in real-time and periodically report them to the data concentrator via two-way communications [32], [33]. Data concentrators which are strategically positioned in substations are used to collect smart data from multiple smart meters. They relay the gathered data to the AMI Headend of utility companies for pricing and decision-making. Not all smart meters can communicate with the data concentrator directly [36]. Intermediate smart meters cooperate in relaying data packets from others till the packets reach the concentrator. Here, a security concern arises: even if the source node is secure, the intermediate node(s) or even the data concentrator can be compromised. They could modify or forge relayed smart data without being recognized by the smart grid. Therefore, security solutions especially the authentication schemes are mandatory to safeguard smart data.

Nevertheless, developing a security scheme for the smart grid is a challenging task because of the following reasons. 1) Huge volumes of data: it is estimated that the amount of data transported across the smart grid will be an increase of an order of magnitude within the next ten years [9], [37]. Therefore, it is required for the smart grid’s security scheme to be efficient and scalable enough to handle large data volumes. 2) Limited resources: Many components of the smart grid, e.g., smart meters, are usually equipped with low-end processors and limited amount of memory which cannot handle heavy computations too often. Wireless channels are
also low-bandwidth. This further necessitates the efficiency and light-weightness of the security scheme. Refer to Appendix B1. 3) Various requirements: smart data belongs to different kinds of categories [3], [31] (refer to Appendix B2), and thus are of different reliability and security requirements. Protective relaying data, for instance, is required to be transported without loss, and any latency more than 4 milliseconds is not affordable. In contrast, meter reading, for example, can be tolerant to latency ranging from minutes to hours [31]. Consequently, the smart grid’s security scheme should be able to provide different security solutions tailored to meet varying requirements. 4) Vital demands for reliability and fault protection: in a global utility consumer survey conducted by IBM in 2011, consumers highly expect the reliable energy supply and the prompt service restoration [38]. The nature of the smart grid ranks Reliability, Safety, and Availability (RSA) as the highest priority over the traditional security objectives of Confidentiality, Integrity and Authentication (CIA) [9]. Hence, the smart grid’s security scheme should not only satisfy CIA, but, more importantly, heavily emphasize on the built-in RSA capabilities. Moreover, in order to minimize the total outage/fault times of the smart grid’s operations, efficient fault-tolerance and fault diagnosis services are essential.

B. Our Contributions

Pioneer researches [2], [3], [4], [8], [9], [11], [18], [19], [21], [25], [28], [29], [30] have extensively explored ideas and methods to secure the smart data. Thus far, they mainly focus on the integrity and confidentiality issues. Unfortunately, issues concerning with authentication, its efficiency and fault diagnosis are not comprehensively addressed. Without a suitable authentication service, malicious smart meters could pretend to be others, spoof the forgery meter ID, falsify power usage data, and get power supply for free. Hence, customers and/or utility company face the risks of financial loss. To provide smart data authentication, digital signature is selected as the authentication primitive in this paper after a comprehensive comparison (refer to Section II - A for details). But, the signature technology demands heavy computation cost. Since most smart meters lack powerful computational capability, reducing numbers of digital signatures and verifications becomes necessary. In order to address this issue, we propose an efficient and reliable scheme to authenticate delay-tolerant smart data, e.g., smart metering, market pricing information, by means of lossless data aggregation (data concatenation) over lossy communication channels in the smart grid. The new contributions of proposed scheme are listed below:

1) Efficiency: To reduce the numbers of signature and verification operations, signature amortization and batch verification are respectively employed. To decrease the number of transmitted signatures, a Minimum Spanning Tree (MST)-based signature aggregation tree is proposed.

2) Reliability: Erasure code is combined with signature amortization to support fault tolerance against signature packet losses for lossy channels.

3) Fault diagnosis: Efficient fault diagnosis algorithms are designed for batch verification and signature aggregation to detect forgery signatures and to minimize fault times.

4) Authentication with fault tolerance: Digital signatures are deployed in such a way that when the data concentrator / AMI Headend is out of service, alternative or redundant devices can directly verify the following digital signatures without any additional setup or configuration. This solves the single-point failure problem.

5) Implementation: A software package is developed to accomplish our proposed authentication scheme and faults diagnosis algorithms. We implement and integrate cryptographic primitives or schemes like batch verification, signature aggregation, tree-based fault diagnosis algorithms, etc.

Due to the space limit, cryptographic primitives, adapted MST and designed algorithms cannot be included in main documents. Refer to [16] or Appendix for detailed description.

II. AUTHENTICATION PRIMITIVES

A. Digital Signature vs. Pairwise Key

Both pairwise symmetric key incorporating with Hash-based Message Authentication Code (HMAC) schemes and public-key (digital signature) schemes are intensively utilized in practice for authentications. The pairwise key scheme demonstrates that its rate of data throughput is high and its key lengths (e.g. 64 /128 bits for symmetric key vs. 1024 bits for private key of public key system) is relatively short. However, it raises complicated key management issues: 1) A number of key pairs should be managed in a large network which results in the mandatory deployment of an unconditionally trusted TTP (Trusted Third Party). 2) The frequency to refresh session keys is high – the worst case is that each communication session demands a new session key [20]. It implies that key management of pairwise key system requires expensive cost.

Consequently, there are a few concrete limits while utilizing pairwise key and HMAC in the smart grid: 1) Smart data collected from smart ends are demanded by multiple recipients, e.g. data concentrators, workstations in substations or AMI Headend in utility companies. Thus, hop-by-hop pairwise keys are required and encryption and decryption at every hop is needed. Or, a few couples of pairwise keys should be established: between a smart meter and a data concentrator; between a smart meter and an AMI Headend / workstations in utility companies’ networks; etc. 2) Although most of smart nodes, e.g. smart meters are stationary, there are still a number of roaming nodes e.g. Plug-in Electrical Vehicles (PEV). When a PEV roams to a new area, a new pairwise key is required between the PEV and the corresponding local data concentrator or AMI Headend or workstations in control networks. 3) Pairwise key scheme presents single-point failure. Once the data concentrator is out of service, complicated backup mechanism is demanded. Or, starting over the mechanism to establish secure channels between every smart meter and the backup/alternative data concentrator is required. Thus, pairwise key consumes heavy cost to provide fault tolerance service.
In contrast, authentication schemes relying on public key operations e.g. digital signatures are vice versa: 1) It is fault-tolerant because a backup/alternative data concentrator can verify the digital signature seamlessly without further pre-configuration. 2) It can accommodate the roaming PEV and process its authentication request efficiently. 3) Key management is easier as only a functionally trusted TTP is required. “Off-line” mode as opposed to in real time is required. 4) Its private key / public key pairs can be valid for long periods (e.g. many sessions or even years) [20]. However, its disadvantage is that its throughput rates are significantly slower than best known symmetric key encryption. And, its key bit lengths are long.

In this paper, we are targeting on tremendously reducing the number of digitally signing and verification operations in order to promote our authentication scheme’s performance.

B. Digital Signature Schemes

Digital signature, which we use in this paper to authenticate smart data, is a cryptographic primitive. As one example, we introduce a short signature scheme of Boneh, Lynn, and Shacham (BLS) [7] whose key length of 160 bits provides similar level of security to 1024-bit RSA in Appendix C. Then, we briefly describe batch verification [5], [7], [12] and signature aggregation [6]. Here, Public Key Infrastructure (PKI) [20], [24] and TTP are used to issue / revoke certificates and public & private key pairs for smart meters. Refer to [16] and Appendix-C for detailed descriptions.

III. PROPOSED SOLUTIONS

In this section, we propose an authentication scheme to legalize the smart data with tremendously less signing and verification operations. Meanwhile, our scheme provides fault tolerance and fault diagnosis services which satisfy electric power industries’ request for reliability and safety [35]. For sake of communication efficiency, we utilize MST algorithm to aggregate lossless smart data (refer to Appendix-D or [16]).

A. Batch Verification and Fault Diagnose Algorithm

The deployment of digital signature makes our solution fault tolerant in architecture. However, per-packet signing and per-signature verification is computationally expensive: The concentrator needs to verify every signature. To promote concentrator’s efficiency, batch verification is deployed which provides the same level of security but reduces the number of verification operations from l to 1 when the concentrator verifies l signatures signed by the same sender. Fig. 1 (a) and (b) illuminates change after deployments of batch verification. Since the verification operation relying on pairing operation costs significantly higher than multiplication [1], batch verification expressed in Algorithm 1 (refer to Appendix A) saves large computational resources for the concentrator.

Fault Diagnosis Algorithm for Batch Verification

To pinpoint the cause of batch verification failures and locate bogus signatures, we contribute two novel fault diagnosis algorithms: one is Binary Code (BC) fault diagnosis algorithm and the other is α—ary Tree-based (αTree) fault diagnosis algorithm.

1) Binary Code (BC) fault diagnosis algorithm

This algorithm is used to handle a scenario in which there is one and only one bogus signature among l received signatures, \( \sigma_0 \ldots \sigma_{l-1} \). In detail, every signature can be identified by an index number ranging from 0 to \( l - 1 \) base on its arrival sequence. Without losing generality, let us assume \( l = 2^r \). Our algorithm uses an array with x elements, namely, \( B[0 \ldots x-1] \). Each element in \( B[] \) is calculated by some multiplication combinations of l signatures. The rule is that, for signature \( \sigma_j \), if the \( j^{th} \) bit of binary value \( j \) is equal with 1, the \( j^{th} \) element in the array, \( B[i] \), will multiply signature \( \sigma_j \). Otherwise, do not. The result of each element in the array is verified by batch verification scheme. Algorithm 2 (refer to Appendix A) describes how it works. Fig. 2 (a) demonstrates the procedure in case that \( l = 8 \) and \( \sigma_2 \) is the fake signature.
2) α—ary Tree-based (αTree) fault diagnosis algorithm

This algorithm can be used for any scenario regardless the number of fake signatures received. In details, a concentrator constructs an α—ary verification tree T in which every node can be denoted as $<h,i>$ where $h$ is the height (level) of the node and $i$ is the index of the node at level $h$. Each node is associated with a signature. There are two kinds of nodes in $T$: leaf nodes and intermediate nodes. The leaf node’s signature is assigned with signatures the concentrator receives from the meter. The intermediate node’s signature is the multiplication of all its children’s signatures. Please refer to (1) on how to calculate signatures associated with nodes in the tree.

$$
\sigma_{<h,i>} = \begin{cases} 
\sigma_i & \text{if } n_{<h,i>} = \text{Leaf} \\
\prod_{j=0}^{l<h,i> \land (i+j)} \sigma_{<h+1,a(i+j)} & \text{if } n_{<h,i>} \neq \text{Leaf}
\end{cases}
$$

The fault diagnosis verification algorithm starts at the root node. First, verify its signatures. If there is a verification failure, all this node’s children nodes should be verified. Otherwise, do not verify any of its offspring. Repeat this procedure. This procedure is addressed in Algorithm 3 (refer to Appendix A) in which $\alpha = 3$ as a study case. Comparing with [13], our proposal is more comprehensive. Refer to Fig. 2 (b) as an example.

B. Signature Amortization for Package Blocks

Batch verification scheme in subsection A can verify $n$ signatures in one verification operation rather than per-signature individually. This saves significant processing resources for concentrators. However, the smart meter still has to sign per-packet individually. Since most smart meters are low-capacity, too many signings will drain smart meter’s processing capacity. Solutions to sign $n$ packets with only one signature are highly demanded by smart meters. Our approach deploys an efficient scheme, Signature Amortization (SAm) to amortize the digital signature over a block of packages. SAm is described in Algorithm 4 (refer to Appendix A). Fig. 1(c) illustrates how signature amortization works.

Furthermore, the communication in last mile of the smart grid maybe lossy as well as has channel instability and restricted resources. Packets could be lost during data transmission. For example, power control messages such as demand-respond data own higher priority over smart metering data. When the congestion happens, it is possible to drop the smart metering data and its signature. Therefore, if a signature lost, its corresponding block of packages cannot be verified. Erasure code such as Information Dispersal Algorithm (IDA) [22] could be used to encode a signature and amortize the result over a block of packets. Even only $m$ out of $n$ packets ($m < n$) are successfully delivered to the receiver end, the signature still can be decoded. We adapt IDA algorithm with our improvement which is demonstrated in Algorithm 5 (refer to Appendix A). Refer to [17] and [22] for IDA.

C. MST-based Signature Aggregation (MST-SA)

In batch verification and signature amortization, all signatures will be sent over communication channels. Considering about the length of a signature (e.g., 1024 bits for RSA digital signature and 157 bits for pairing [7]), it will consume the limited bandwidth of wireless communication. Signature aggregation can save communication cost via aggregating a number of signatures into a single one, only which will be transmitted on the air.

MST-SA: Our proposal, MST-SA integrates MST structure with the signature aggregation scheme. We term it MST signature tree (shortly, signature tree) which holds the same nodes and structure as MST. Furthermore, each node, namely $n_i$, representing a smart meter, is associated with two signatures, $(\sigma_{n_i}^k, \sigma_{n_i, \text{tree}}^k)$. The former, $\sigma_{n_i}^k$ (namely node signature) is the signature to sign message $m_k$ from the smart meter $n_i$ (concentrator’s node signature is assumed to be 1). The later, $\sigma_{n_i, \text{tree}}^k$, (namely tree signature) is the signature for sub-tree rooted at node $n_i$. $\sigma_{n_i, \text{tree}}^k$ is calculated with signature aggregation scheme — multiplying $n_i$’s node signature $\sigma_{n_i}^k$ by all tree signatures of node $n_i$’s children nodes. Refer to formula (2) as follows:

$$
\sigma_{n_i, \text{tree}}^k \left\{ \begin{array}{ll} 
\sigma_{n_i}^k, & \text{if } n_i = \text{leaf node} \\
\prod_{j=0}^{k \leq l} \sigma_{n_j, \text{tree}}^k, & \text{if } n_i \neq \text{leaf node}
\end{array} \right.
$$

where $n_i$ is $n_i$’s child; $k$: message number.

Fig. 3. MST-based signature aggregation for smart meters.
In the MST-based signature aggregation, after receiving all its children nodes’ tree signatures, a node, following (2), calculates its tree signature which is sent to its parent node. Then, its parent follows the same process. Repeat this procedure with the bottom-up manner. At last, tree signature of the root node can be calculated by the concentrator. Algorithm 6 (refer to Appendix A) and Fig. 3 demonstrate it.

An active attacker can drain the concentrator’s computational resource by constantly sending bogus signatures. To handle the scenario that the verification for root node’s tree signature fails, MST-based fault diagnosis verification algorithm is designed to pinpoint the forged signings for signatures aggregation operation. The concentrator asks all nodes in MST for their tree signatures. After receiving them all, the concentrator constructs the MST-based signature tree, follows the post-order tree travel algorithm to explore every node in it, and verify every node’s tree signature. If there is a failed signature from any smart meters, this algorithm will not calculate any ancestor of this failed node until meeting a leaf node during post-order tree travel. All this procedure is described in Algorithm 7 (refer to Appendix A).

D. Integrated Authentication Solution

In this subsection, a scenario is given as an example to demonstrate how it executes: smart meter, $n_x$ authenticates $\alpha$ blocks of packets, $\{B_1, B_2, \ldots, B_\alpha\}$, and every block contains $\beta$ packets, $\{M^{B_1}_1, M^{B_1}_2, \ldots, M^{B_\beta}_1\}$. First, Alg. 4-Signature Amortization is used to sign every block in amortization. Eventually, $\alpha$ signatures $\{\sigma_1, \ldots, \sigma_\alpha\}$ are generated for corresponding blocks. Each of them is encoded by IDA and sent to concentrator one by one. The concentrator decodes them by IDA but does not verify them individually. Instead, it uses Alg. 1-Batch Verification to verify them in a batch. Only one verification operation is required for all $\alpha$ blocks. During the data transportation in MST, Alg.6 - MST-based Signature Aggregation is used by intermediate node (e.g. $n_y$) to aggregate signatures into single one which is sent to the concentrator. Failures occurred in Alg.1 and Alg.6 are diagnosed by BC (Alg. 2) / $\alpha$Tree (Alg. 3) and MST-based verification tree (Alg. 7), respectively.

IV. SECURITY AND PERFORMANCE ANALYSES

A. Security Analysis

The security of our approach is based on digital signature schemes [20]. Signature aggregation, batch verification and signature amortization are proved to be secure in [6], [12], [22] respectively. Conventional authentication requirement could be satisfied by our authentication scheme. We loosely analyze our proposed scheme in Appendix E.

Replay Attack:

To protect our scheme against replay attacks, time-stamp / random number can be employed so that malicious attacks cannot reuse previous signatures. The smart meter can generate the random value with sufficient length (e.g. 64 bits) so that it is too expensive to successfully guess the random value for the adversary. Therefore, the probability to replay the previous valid signature is negligible.

Denial of Services (DoS) Attacks:

To mitigate Denial of Services (DoS) attacks, some solutions proposed in previous approaches could be borrowed: use distillation codes; then, forged packets and legal packets are separated into different categories; at last, erasing code is invoked over each category. Refer to [24] for details.

B. Performance Analysis for Authentication Schemes

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>PERFORMANCE EVALUATION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Schemes</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Per-Sign-Ver</td>
<td>$(\alpha \beta n) \text{Ver.}</td>
</tr>
<tr>
<td>Batch Verif.</td>
<td>$(\alpha n) \text{Ver} + (\alpha \beta n) \text{Mul.}</td>
</tr>
<tr>
<td>Sign Aggre.</td>
<td>$(\alpha \beta n) \text{Ver} + (\alpha \beta) \text{Sign.}</td>
</tr>
<tr>
<td>Sign. Amort.</td>
<td>$(\beta n) \text{Ver.} + \text{Mul.}</td>
</tr>
<tr>
<td>Our proposal</td>
<td>$(n) \text{Ver.} + (\alpha \beta n) \text{Mul.}</td>
</tr>
</tbody>
</table>

*$c$: constant; $\text{Mul.}$: multiplication; $(\text{Sign.})$: bit length of signature;

Table I evaluates the communication and computation cost for scenario we mentioned in section III (D). The metrics we use is the number of signature, verification and multiplication operations. Notice that signature and verification operations cost a similar level but they both execute hundreds of times heavier than multiplication [1]. According to Table I, we find that our authentication scheme in this paper is more lightweight than per-sign and per-verification solution: the number of verifications at the concentrator end is significantly reduced by $1 / \alpha \beta$ and the number of signature operations at every meter is steadily dwindled down by $1 / \beta$. We conclude that our proposal makes substantial performance gains.

Authentication Message Delay

The authentication delays for per-packet signing and per-signature verification approach and our scheme are listed below. Notice that the signal propagation delay is not included since it is negligible.

\[
T_{\text{Per-Sign-Ver}} = T_{\text{Meter-Signing}} + T_{\text{Meter-Sent}} + \sum_{i=1}^{l} (T_{\text{Meter_i-Receive}} + T_{\text{Meter_i-Sent}}) + T_{\text{Concentrator-Receive}} + \sum_{i=1}^{l} T_{\text{Concentrator-Verify_i}}
\]

\[
T_{\text{Our-Scheme}} = n(T_{\text{CollectBlock}} + T_{\text{IDA-en}}) + T_{\text{Meter-Signing}} + T_{\text{Meter-Sent}} + \sum_{i=1}^{l} (T_{\text{Meter_i-Receive}} + T_{\text{Meter_i-Sent}} + T_{\text{Mul.}}) + T_{\text{Concentrator-Receive}} + nT_{\text{IDA-de}} + T_{\text{Concentrator-Verify_i}}
\]
The dominating operation costing the most times in formulas above is verification and signing operations which last not more than 20 ms together according to our experiments. Furthermore, times spending on waiting for the completion of the whole block data collection is flexible. This delay part depends on the block size and the rate of channels; however, it is trivial as compared to that of verification and signing operations. Therefore, the delay time is affordable for smart data e.g. power usage data (tolerant to minutes and even hours) except for near real-time data (tolerant to 4ms) [31].

Payload

Under the PKI solution, BLS algorithm signs the message with its private key and includes the Certification Authority’s certificates before forwarding them to the receipts:

\[ n_1 \rightarrow Rs: \{ m_k \parallel \sigma_{n_i}^k \parallel Cert_{n_i} \parallel Timestamp \};\]

Where \( n_1 \) is a smart device; \( m_k \) is plaintext; \( Rs \) are receipts; \( Cert_{n_i} \) is \( n_i \)’s certificate. In our implementation, the secret key length is 210 bytes and the public key length 64 bytes (x-coordinate). The signature \( \sigma_{n_i}^k \) is 64 bytes. \( Cert_{n_i} \) is consisting of a public key and \( n_i \)’s certificates (around 64+125 bytes, together). Therefore, the entire payload is roughly 254 bytes for one BLS signature. Regarding to the introduction of signature amortization and aggregated signature algorithm in our scheme, the payload is reduced to \( \approx 254/(2*q) = 127/\beta \) bytes.

However, IDA-encode increases the payload to \( \approx (127*n)/(m*\beta) \) bytes where \( m \) is the number of inputted pieces, \( n \) is the number of outputted pieces and \( n/m \) is redundant ratio for IDA-encode algorithm.

C. Performance Analysis for Fault Diagnosis Algorithms

Table II evaluates the number of verifications required to pinpoint the fault in case of only one bogus signature. We find that the Binary Code Alg. shows the best performance. They always provide better performance than others.

In case of more than one bogus signature, we analyze the worst scenario and the average scenario of tree-based fault diagnosis algorithms. Not losing generality, we assume that the concentrator receives \( N = \alpha h \) signatures, in which \( x \) signatures are bogus (\( h, x, \) and \( N \) are integers).

**Worst scenario analysis**

Considering about a balanced tree with degree \( \alpha \) and height \( h \), there are \( N = \alpha h \) leaf nodes representing \( N \) received signatures. All leaf nodes associated with bogus signatures require verification. So do their ancestors and their siblings.

When the number of overlapped intermediate nodes is the minimum, the worse scenario happens. It occurs when \( x \) bogus signatures are evenly distributed on \( N \) leaf nodes. The number of verifications for the worse case is derived:

\[ Ver_{wrst}(N, x) = x \times \alpha \times \log_\alpha \frac{N}{x} + \frac{xx - 1}{\alpha - 1}. \]  

**Average scenario analysis**

The average scenario happens when \( x \) bogus signatures are randomly distributed on \( N \) leaf nodes. The average number of verifications is inferred:

\[ Ver_{avg}(N, x) = \alpha \sum_{i=0}^{h-1} \alpha^i \left( 1 - \frac{C_{x}^{N-x} \alpha^i}{C_{N}^{x}} \right). \]  

Table III shows the evaluation for binary and trinary trees.
Fig. 5 Overall Computational Cost and Communication Overhead for Our Proposal

Fig. 6 Experimental Results for Batch Verification and Signature Aggregation

Simulation Result for Fault Diagnosis Algorithms

Based on MATLAB platform [39], simulations for fault diagnosis algorithms have been accomplished. The metric is the number of verification operations. Fig. 4 (a) shows the evaluated number of verifications processed when only one bogus signature exists for 100 (a randomly selected number) signatures. The result demonstrates that our binary code fault diagnosis algorithm processes the least number of verification operations. When multiple bogus signatures exist in 100 signatures, the evaluated experimental results are demonstrated in Fig. 4 (b) and (c), for the average case and for the worst case, respectively. The result shows that the binary tree fault diagnosis algorithm or the trinary tree one ranks the best for different scenarios. However, most of the times, trinary tree algorithm is better. Furthermore, according to Fig. 4 (b) and (c), binary and trinary tree fault diagnosis algorithms respectively show better performance than per-signature verification in case of less than 20 and 22 bogus signatures, among all 100 received signatures.

V. IMPLEMENTATION AND TEST RESULTS

A. Implementation Details

We develop a software package and use it as a test bed for experimental evaluation. We implement and integrates components of Batch Verification, Signature Aggregation, Binary-tree Fault Diagnose, Trinary-tree Fault Diagnose, …, $\alpha$Tree fault diagnosis algorithms, etc. They are implemented by C language based on Pairing-Based Cryptography (PBC) library [41] built on the GNU Multiple Precision arithmetic (GMP) library [40]; GMP library provides arbitrary precision arithmetic APIs which are invoked by PBC to support pairing-based cryptosystem. Our implementation has been executed on Virtual Machine hosted by Oracle’s VirtualBox. Here is the detailed configuration of VM – OS: Ubuntu 11.10; Memory: 496MB; Processor: Intel Core i5-M560; CPU 2.67GHz; Disk 7.9 GB. In our implementation, we use the pairing-friendly elliptic curves and $E(F_p); y^2 = x^3 + x$ with a 512-bit prime. For details, MNT elliptic curve of embedding degree 6 is with order 160-bit and base field order 512-bit.
Moreover, in our scheme, we use a pair \((x, y)\) to represent a point on an elliptic curve group \(G \subset E\left(F_q\right)\). However, instead of the pair, only the \(x\)-coordinate of the signature point \(s \in G\) on elliptic curve group \(G\) is sent as they demonstrate the same security level.

### B. Test Results

The goals of our experiments are to estimate the performance of our authentication scheme and fault diagnosis algorithms specifically focusing on the computational cost and communication overhead. These goals enable us to showcase the feasibility of our schemes. Furthermore, it can help us determine the performance gains or additional cost introduced by our scheme at different scenarios. Regarding our authentication scheme, we execute the cryptographic primitives we implement e.g. BLS signature, batch verification, signature aggregation, signature amortization, etc. individually. Thereafter, we execute our entire authentication scheme. To evaluate our fault diagnosis algorithms, we demonstrate the times and number of verification operations required to pinpoint the bogus signatures among 100 signatures. The times to generate, initialize, delete and cleanup the fault diagnosis trees are counted in.

#### Computational Cost:

Our experimental result shows that the times to achieve a BLS signing and a BLS verification are ranging from 1.59ms, to 10.44ms and from 2.22ms to 18.53ms, respectively.

Fig. 6 (a) shows the comparison between batch verification and per-signature verification in terms of verification execution times when the batch size ranges from 1 to 500. Fig. 6 (b) compares per-signature verification with signature aggregation in terms of verification execution times when signatures are signed by different smart meters, the number of which range from 50 to 500.

Fig. 5 (a) shows the overall computation cost comparison between our integrated solution and per-sign per-verification scheme. It demonstrates that the entire computational times are significantly decreased. It matches with our performance assessment: the numbers of verification and signing operations are dropped by \(1/\alpha\) and \(1/\beta\), respectively.

Fig. 7 (a)-(d) depicts the number of verification operations and times required to pinpoint from 1 to 50 forgery signatures among 100 signatures when using binary tree, trinary tree, 4-
ary tree, 5-ary tree, 10-ary tree, 20-ary tree, 30-ary tree and 50-ary tree. It is based on a scenario in which bogus signatures are randomly distributed among 100 signatures. Fig. Appendix 3 (a)-(d) (Appendix F) demonstrates scenario with evenly distributed. The result shows that our fault diagnosis algorithms perform better when the number bogus signatures is roughly less than 15. However, different fault diagnosis trees demonstrate unlike thresholds which range from 8 to 35.

Communication Overhead:

To evaluate the communication overhead, our proposal is further simulated via Network Simulation-2 (ns-2) [34], a widely used simulation tool. This simulation utilized the test scenario: area (50 × 50 meters), 50 nodes (1GHz processor for each node), 10 repetitions, mobility ratio (10% mobile nodes), mobility mode (random waypoint model with no pause time, maximum speed 20 m/s - high mobility scenarios), Zigbee wireless communications and ns 2.30.

Fig. 5 (b) addresses the overall communication costs, when 200 packets are sent by per smart meter (50 blocks and 4 packets per block). Fig. 5 (b) shows that number of transmitted digital signatures is reduced by around 50% via deploying signature aggregation. Since signature amortization scheme reduces the number of signatures from $\beta$ to 1 per block, our proposal reduces the overall authentication messages by $\approx 1 / (2\beta)$.

VI. RELATED WORKS

Since the smart grid is a new area for both industry and academics, security research for the smart grid is just starting [3]. A few pioneer works [2], [8], [15], [18], [21], [25], [28], [29], [30] have been proposed. They mainly focus on confidentiality and integrity. Even though some solutions concentrate on authentication service for the smart grid, they still utilize the per-packet signature and per-signature verification scheme, which is computationally expensive.

Meanwhile, a number of lightweight authentication schemes have been proposed for self-organized networks [23]. Nevertheless, none of them can be used in the smart grid directly due to the smart grid’s unique requirements for RSA. In this section, we will review recent security solutions designed to protect the smart grids.

A. Bartoli et al. [2] propose a secure and lossless aggregation protocol providing CIA service. It includes two security solutions, end-to-end and hop-by-hop: The former is based on secret credentials shared between the gateway and every meter. In the later, pairwise keys are used between each smart meter and its one-hop neighbor. The pairwise key plays the authentication role. Nevertheless, costs to establish presumed pairwise keys are hidden. Furthermore, key maintenance cost should also be considered.

F. Li, B. Luo and P. Liu [18] present an efficient information aggregation approach, in which, an aggregation tree constructed via breadth-first traversal of the graph and rooted at the collector unit, is deployed to cover all smart meters in the neighborhood. The control unit collects all smart meters’ information in this area. Furthermore, to protect users’ privacy, all information is encrypted by homomorphic encryption algorithm. However, the approach faces the potential risk that malicious smart meters can forge packets but the smart grid cannot detect / diagnose bogus data.

D. Wu and C. Zhou [29] propose a key management scheme for the smart grid in which well-known Needham-Schroeder protocol is deployed to generate session keys. This proposal eliminates man-in-the-middle and replay attacks.

H. Khurana et al. [15] provide guidelines for authentication protocol used in the smart grid. Seven principles, including names, encodings, trust assumptions, secret releases, security parameters, etc. are proposed.

D. Wei et al. [28] propose a distributed and scalable security framework in concept with the layered architecture. It can protect the smart grid via integrating security agents, security switches and security managements.

M. M. Fouda et al. [9] propose a lightweight authentication scheme in which smart meters can first establish mutual authentication and generate session keys via Diffie-Hellman key exchange. The following session key will be updated with the hash-based authentication code technologies.


A. R. Metke and R. L. Ekl [21] propose a security solution for the smart grid by utilizing the PKI and trust computing. The three components, certificate management, trust anchor security and attribute certificate in PKI are carefully illustrated and tailored to meet the smart grid’s security requirement.

J. Chao et al. [8] adapts RFID communication standard security protocol and utilizes it in the smart grid. One-time password is deployed for user authentication.

H. K.-H. So et al. [25] propose an Identification-Based Signcryption (IBS) approach based on elliptic curve public key cryptography to provide CIA services.

VII. CONCLUSION

Authentication for smart data is a critical issue for the smart grid’s security. Unfortunately, previous researches deploy the standardized authentication protocols or the per-signing per-verification scheme to validate messages. They are lack of performance optimization, vulnerable to packet loss, complicated in key management and not resilient to DoS attacks. Furthermore, efficient tools to pinpoint the forged signature are not provided. In this paper, we integrated several efficient signature schemes to significantly reduce costs to achieve the authentication goal. Our proposal is an efficient scheme because: 1) The number of verifications at the concentrator end is significantly reduced by $1/(\alpha \beta)$ and the number of signature operations at every meter is steadily dwindled down by $1/\beta$ where $\alpha$ is the number of blocks for packets and $\beta$ is the number of packets in every block. 2) The overall communication cost in terms of the number of authentication messages is reduced by $\approx 1/(2\beta)$. Most importantly, fault diagnosis algorithms are presented to detect the bogus signatures and minimize the fault execution time.
VIII. REFERENCES


APPENDIX

A. Algorithms

Algorithm 1: Batch Verification
/* concentrator knows smart meter’s public key and, the number of accumulated signatures from n_i, list_i, in advance. */
/* The concentrator processes the followings: */
For ( k ← 0; k < k + 1 )
    Listen on the channel and receives triple \{ n_k, M_k, \sigma_k \}
    temp_h ← temp_h \times H(y_n, M_k);  
    temp_o \leftarrow temp_o \times \sigma_k^i; 
End For
/* Verifies signatures sent from smart meter */
IF \( \text{calls “Fault Diagnosis Algorithm for Batch Verification”; return FALSE; } \)
End IF
Return TRUE;

Algorithm 2: Binary Code (BC) fault diagnosis algorithm
Result \leftarrow 0;
For ( k ← 0; k < k + 1 )
    For ( i ← 0; i < x; i ← i + 1 )
        For ( k←0; k < k + 1 )
            IF (k’s i\text{th} bit equal 1)
                temp_h[k] ← temp_h[k] \times H(y_n, M_k);
                temp_o \leftarrow temp_o \times \sigma_k^i;
            End IF
        End For
    End For
End For
/* Verification */
IF \( \hat{\sigma} (\text{temp}_h, y_n) \neq \hat{\sigma} (\text{temp}_o, g) \)
    calls “Fault Diagnosis Algorithm for Batch Verification”; return FALSE;
End IF

Algorithm 3: aTree fault diagnosis algorithm
IF (node = NULL)
    Return NULL;
For ( i=0; i < all node’s child node; i++)
    nodePTR ← child[i]
    IF (verification (nodePTR) \text{ Equal: False})
        aTree(nodePTR);
    End IF
End For

Algorithm 4: Signature Amortization (SAm) Algorithm
/* each smart meter \( n_i \) signs a block of messages \( M_1, ..., M_{n_i} \) */
h_{n_i} ← NULL /* store hash result of the block */
For ( k←1; k ≤ n_i; k ← k + 1 )
h_{n_i} \leftarrow h_{n_i} || Hash(M_k)
End For

h_{n_i} \leftarrow Hash(h_{n_i});  \ h_i \leftarrow H(y_n, h_{n_i});  \ \sigma_i \leftarrow h_{n_i}(h_i);
/* Get n slices of usage data */  
\sigma_1, \sigma_2, ..., \sigma_{n_i} = HDA - encode(\sigma_i, n_i, m_i);
/* Get n slices of hash result of usage data */
h_1, h_2, ..., h_{n_i} = HDA - encode(h_{n_i}, n_i, m_i);
For ( k←1; k ≤ n_i; k ← k + 1 )
    /* reassemble data and hash; send results to parent */
    \( n_i \rightarrow n_i \)’s parent node: \{ M_k || \sigma_k[k] || h_2[k] \}
End For
/* Concent. receives slices and blocks hash result slices */
\( \rightarrow \)’s parent node: \{ \}

Algorithm 5: IDA Algorithm

Algorithm 5-A: IDA-Encode
/* sender processes the followings */
/* INPUT : a block of data \( C_j \) */
/* OUTPUT : encoded vectors \( T_1, T_2, ... T_n \) */
/* (1) Split \( C_j \) into 
\( N/m \) pieces where \( N = n/8 \)
\( C_j = (c_{1,1}, c_{1,2}, ..., c_{1,m}) \), ..., \( (c_{N-m+1}, ..., c_{N}) \)
where \( c_i \) is byte */
R_j = (c_{i-1}+1, ..., c_{i,m}), where \( i < N/m \)
/* (2) Following the specification of IDA to process \( C_j \) */
/* choose \( n \) vectors \( A_i \); let every subset of \( m \) different vectors are linearly independent. Next, process \( C_j \) */
A_i = (\( a_{i1}, ..., a_{im} \)), \( 1 \leq i \leq n \)
\( T_i = A_i \cdot (R_1, R_2, ..., R_{N/m}) \)
\( = (a_{i1}c_{1,1} + ... + a_{im}c_{N,1}) \cdot ...
\( = (a_{i1}c_{1,m} + ... + a_{im}c_{N,m}) \)
where \( 1 \leq i \leq n \)
/* (3) Send to the receiver. */
Forwardsender → receiver: \( T_1, T_2, ..., T_n \)

Algorithm 5-B: IDA-Decode
/* Receiver processes the following */
/* INPUT : encoded vectors \( T_1, T_2, ..., T_m \) */
/* OUTPUT : a block of data \( C_j \) */
/* (1) Assume that the receiver receives \( T_1, T_2, ..., T_m \) */
\( T_1 = A_1 \cdot R_1, A_2R_2, ..., A_m \cdot R_{N/m} \)
\( T_2 = A_1 \cdot R_1, A_2R_2, ..., A_{m-1} \cdot R_{N/m} \)
\( ... \)
\( T_m = A_1 \cdot R_1, A_2R_2, ..., A_{m-1} \cdot R_{N/m} \)
/* Prepare for the calculation of \( R_j \) */
Based on \( T_1, ..., T_m \) and formula (8), we can get: */
consumers. Wired networks such as Power Line Communication (PLC) are preferred to save costs. Zigbee / 802.15.4 technology polls data every 15 minutes and standardizes with the Fi alliance, etc.

Smart meters utilize Phasor Measurement Units (PMU) and Global Positioning System (GPS) time stamps to measure power status and electricity consumption based on waveforms as well as the magnitude and the phase angle of voltage [4]. Status, usage data and events are collected and information flow is transferred following the path: smart meters, data concentrators (in substations), and the central system of the utility company. Meanwhile, other messages, e.g., demand-response commands are sent back and forth.

Two-way communication systems, wired or wireless, act as the backbone to relay data packets in the smart grid. To connect substations and the central system, wired networks such as Power Line Communication (PLC) [31] are deployed to transfer control commands, statuses, usage data, etc. For communications in the last mile (e.g. from a substation to smart meters), wireless networks such as Zigbee, Wi-Fi alliance, etc. are preferred to save costs. Zigbee / 802.15.4 technology polls data every 15 minutes and standardizes with the data rate of 250kbps in maximum, which both result in a 4 milliseconds’ interval. This imposes a constraint in communication bandwidth [36]. In contrast, although Wi-Fi alliance demonstrates a higher communication capacity, it does not have the mesh function yet.

Up until now, communication systems in the last mile often face harsh network environments, unreliable wireless channels, and relatively demanding operational requirements. The following factors particularly make it more difficult to secure the smart data in the last mile. 1) Although utilizing wireless network communication in the smart grid saves costs,
its native infrastructure is insecure; 2) Most smart meters are configured with low-capacity devices which are restricted in their computational capability. It is hard to perform many and frequent digital signature and verification operations, which are computation-intensive, on the smart meters. Furthermore, unreliable channels and limited communication bandwidths also lead to packet losses / errors; 3) The open architecture in the last mile shares the wireless medium and channels. It is easier for illegitimate users and malicious adversaries to access, interfere, or block the wireless channels. As a result, damages, such as unauthorized message forwarding or high rate of packet losses / errors, are more likely to happen.

We keep the above limitations in mind when designing our proposed security scheme. In particular, we avoid resource-intensive solutions but opt for light-weight yet reliable ones.

**B2. Different Categories of Smart Data**

<table>
<thead>
<tr>
<th>Max. Latency</th>
<th>Communication Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤ 4 ms</td>
<td>Protective relaying</td>
</tr>
<tr>
<td>Sub-seconds</td>
<td>Status Monitoring</td>
</tr>
<tr>
<td>Seconds</td>
<td>Substation SCADA</td>
</tr>
<tr>
<td>Minutes</td>
<td>Market Pricing Info</td>
</tr>
<tr>
<td>Hours</td>
<td>Meter Reading</td>
</tr>
<tr>
<td>More than Days</td>
<td>Long-term usage</td>
</tr>
</tbody>
</table>

As mentioned before, smart grid communications transmit different kinds of smart data which have different requirements. Table Appendix-I (adapted from [3], [31]) shows data categories and corresponding maximum latencies across the smart grid. Acceptable latencies for protective relaying, status monitoring and substation SCADA are less than 4ms, sub-seconds and seconds, respectively. Meanwhile, they are required to be delivered without any losses. Hence, part of low-priority data (e.g. meter reading) may have to be dropped while the bandwidth is inadequate. So, our proposed scheme is designed to handle cases in which some digital signatures associated with low-priority smart data are lost in case of limited channel bandwidth or network congestions.

**C. Cryptographic Primitives**

1) **Bilinear map**

Bilinear map [1] works as the basis of our approach. \( \mathbb{G} \) and \( \mathbb{G}_T \) are a cyclic additive group and a cyclic multiplication group generated by \( P \) with the same order \( q \), respectively. A mapping \( \hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T \) satisfies the following properties:

- **Bilinear**: for all \( u, v \in \mathbb{G}; a, b \in \mathbb{Z} \), we have \( \hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab} \), where \( = \) is an equation;

- **Computable**: there exists an efficient computable algorithm to compute \( \hat{e}(u, v) \), \( \forall u, v \in \mathbb{G} \);

- **Non-degenerate**: for the generator \( g \) of \( \mathbb{G} \), if \( p \) is the order of \( \mathbb{G} \), we have \( \hat{e}(g, g) \neq 1 \in \mathbb{G}_T \);

2) **BLS short signature scheme**

**Key generation**

- Randomly selects \( x \sim \mathbb{Z}_p \) and calculates \( y \sim g^x \in \mathbb{G} \); \( x \) and \( y \) are the private and public keys respectively; \( x \in \mathbb{Z}_p \) and \( y \in \mathbb{G} \); \( \leftarrow \) is assignment;

**Signature generation**

- Given a message \( M \in \{0,1\}^* \), computes \( h \leftarrow H(y, M) \) where \( H \) is a collision-resistant hash function e.g. MapToPoint hash [5], [7] such that \( H:[0,1]^* \rightarrow \mathbb{G} \);

- Computes \( \sigma \leftarrow h^x \) where \( \sigma \in \mathbb{G} \) is signature;

**Signature verification**

- Receiver Bob verifies signature:

- Obtains Alice’s public key \( y \), signature \( \sigma \) and message \( M \), performs \( h \leftarrow H(y, M) \);

- Performs verification: \( \hat{e}(h, y) = \hat{e}(\sigma, g) \);

3) **Batch Verification**

**Signature generation**

- Sender Alice calculates signatures:

- Given \( n \) messages \( \{M_1, M_2, \ldots, M_n\} \) and public key \( y \), computes \( h_i \leftarrow H(y, M_i) \) for all \( i \in [1, n] \);

- Computes \( \sigma_i \leftarrow h_i^x \) where \( \sigma_i \in \mathbb{G} \) is signature;

**Batch verification**

- Receiver Bob verifies signature:

- Obtains Alice’s public key \( y \), signatures \( \{\sigma_1, \sigma_2, \ldots, \sigma_n\} \) and messages \( \{M_1, M_2, \ldots, M_n\} \);

- Performs verification:

\[
\hat{e}\left(\prod_{i=1}^{n} h_i, y\right) = \hat{e}\left(\prod_{i=1}^{n} \sigma_i, g\right) \quad (1)
\]

4) **Signature aggregation**

**Signature aggregation**

- Distinct \( n \) users \( \{u_1, u_2, \ldots, u_n\} \) sign \( n \) distinct messages \( \{M_1, M_2, \ldots, M_n\} \) with its own public key, \( \{y_1, y_2, \ldots, y_n\} \) by BLS scheme, respectively; Calculate: \( h_j \leftarrow H(y_j, M_j) \in \mathbb{G} \); \( \leq j \leq n \); Obtains signatures: \( \{\sigma_{1j}, \ldots, \sigma_{nj}\} \) where \( \sigma_j \leftarrow h_j^x \in \mathbb{G} \) is calculated by corresponding meter;

- Aggregates all \( n \) signatures into a single signature \( \sigma_{12,\ldots,\cdot,n} \leftarrow \prod_{j=1}^{n} \sigma_j \in \mathbb{G} \);

**Signature verification**

- Verifier can:

- Obtains \( n \) users’ Public keys, \( \{y_1, y_2, \ldots, y_n\} \);

- One signature aggregations \( \{\sigma_{12,\ldots,\cdot,n}\} \);

- Messages \( \{M_1, M_2, \ldots, M_n\} \);

- Performs verification:

\[
\hat{e}(\sigma_{12,\ldots,\cdot,n}, g) = \prod_{j=1}^{n} \hat{e}(h_j, y_j) \quad (2)
\]

**D. MST**

Fig. Appendix-2(a) is an example of the smart grid’s communication including both the data concentrator and a number of smart meters. It can be denoted as \( M = ([m_1; \ldots; m_{|M|}]; L) \) where \( m_i \) is a smart meter node and \( L \) is the set of communication channels established by two smart meters. \( M \) is modeled as a connected, undirected graph \( G = ([n_1; \ldots; n_{|M|}], E, W(e)) \) where vertex \( n_i \) corresponds to node \( m_i \) in \( M \). \( E \) denotes the set of edges in \( L \) and \( W \) is the set of weights for all edges. There is an edge in \( E \) between a pair of vertices \( n_i \) and \( n_j \) if nodes \( m_i \) and \( m_j \) in \( M \) enable successful communication directly. Each edge is associated
with a weight number, calculated by the combination of communication bandwidth as well as both smart meters’ CPU power, memory capacity, etc. Our solution utilizes MST algorithm (refer to [26] for details) to construct a spanning tree, based on which, data is collected. Fig. Appendix-2(b) demonstrates the result.

Fig. Appendix-2. Example of MST with smart meters and concentrators

We notice that there are some non-respond scenarios for some smart meters: 1) Smart meters are out of service because of hardware/software failures. 2) Smart meters do not respond to any request due to overloaded tasks or malicious activities such as physical tampering, Denial-of-Service (DoS) attacks, etc. 3) To save the energy, smart meters switch to sleep-mode if applicable in case of no power usage to report.

The approach in [18] using Breadth-First Searching (BFS) spanning tree does not mention how to maintain the spanning tree in case of scenarios aforementioned. In our solution, when not receiving keep-alive/beacon messages from their parents, smart meters send out Parent Request (PR) to the concentrator. The concentrator re-executes MST algorithm within itself, and after completion, broadcasts Parent-Child Association (PCA) updates to nodes with either parent or children node changes. When the concentrator fails, backup concentrator takes changes seamlessly via constructing MST and broadcasting PCAs to all smart meters. It collects data via MST and verifies corresponding signatures without extra configurations. The single-point failure problem is fixed. Less communication cost is required to maintain MST. Refer to Fig. Appendix 2(e) - (f) for details.

E. Security Analyses

Message Authentication:

As one of the basic security requirements, message authentication for smart data in the smart grid is presented by our scheme.

1) In the proposed scheme, the signature is generated by \( \sigma \leftarrow H^x(y,M) \). BLS signature scheme is assumed to follow Computational Diffie-Hellman (CDH) problem in \( \mathbb{G} \) which means that given \( g, g^a, h, h^b \in \mathbb{G} \), it is hard to calculate \( h^a \in \mathbb{G} \). Loosely state it is computationally infeasible to solve random instances of the CDH problem. We also assume \( x \) is randomly created. However, we do not assume the bilinear map in our scheme satisfies Decision Diffie-Hellman (DDH) in which given \( g, g^a, h, h^b \in \mathbb{G} \), it is hard to verify whether \( a = b \) holds. As the matter of fact, we find that DDH is solvable in our scheme. It means that \( a = b \mod p \iff \hat{e}(h, g^a) = \hat{e}(h^b, g) \). Thus, without knowing \( x \), it is computationally infeasible to forge signature \( \sigma \) if \( x \) is carefully chosen. Considering that \( x \) is known only by the smart meter and placed in a secret location, it is difficult for an adversary to derive signature \( \sigma \). Therefore, the signature is not forgeable and the authentication property is accomplished when a smart meter digitally signs the hash result of a package with its private key via BLS signature.

2) The batch verification scheme is also secure against bogus signature: assume the concentrator receives \( m \) valid signatures, \( \sigma_1, \sigma_2, ... , \sigma_m \) from a valid smart meter. When \( m = 1 \), it is the same as BLS signature which is already approved to be against the bogus attack earlier. Then, let us validate that the property still works well in case that \( m = n \):

\[
\hat{e}\left(\prod_{i=1}^{n} \sigma_i, g\right) = \hat{e}\left(\prod_{i=1}^{n} h_i^{x_i}, g\right) = \hat{e}\left(\prod_{i=1}^{n} h_i, g\right) = \hat{e}\left(\prod_{i=1}^{n} h_i, g^{x_i}\right) = \hat{e}\left(\prod_{i=1}^{n} h_i, y\right)
\]

Based on the property of bilinear map, we approve that the left hand of verification equation (1) in Appendix can be expanded and derived from the right hand of (1). Therefore, the batch verification is validated when \( m = n \). Next, we approve that it still holds when \( m = n + 1 \). Assume that the adversary can replace the \((n + 1)^{th}\) signature, \( \sigma_{n+1} \), with a fake signature \( \sigma'_{n+1} \) and assume that the concentrator cannot identify that \( \sigma'_{n+1} \) is bogus. Therefore, based on BLS, verification equation (1) in Appendix should hold even when the concentrator receives \( \sigma'_{n+1} \).

\[
\hat{e}\left(\prod_{i=1}^{n} \sigma_i \times \sigma'_{n+1}, g\right) = \hat{e}\left(\prod_{i=1}^{n} h_i \times h'_{n+1}, y\right) \iff \hat{e}\left(\prod_{i=1}^{n} \sigma_i, g\right) \times \hat{e}(\sigma'_{n+1}, g) = \hat{e}\left(\prod_{i=1}^{n} h_i, y\right) \times \hat{e}(h'_{n+1}, y)
\]

To analyze further, we divide the scenarios into two cases: a) \( h_{n+1} = h'_{n+1} \). We deduce that \( \hat{e}(\sigma'_{n+1}, g) = \hat{e}(h'_{n+1}, y) = \hat{e}(h_{n+1}, y) = \hat{e}(\sigma_{n+1}, g) \). Therefore, based on the property of bilinear map, \( \sigma'_{n+1} = \sigma_{n+1} \). It conflicts with the assumption. b) \( h_{n+1} \neq h'_{n+1} \). It means that after calculating \( h'_{n+1} \) and capturing \( \sigma'_{n+1} \), the adversary can derive \( \sigma'_{n+1} = (h'_{n+1})^x \). It leads to a conclusion that BLS signature is not hard on CDH problem which conflicts with the property of BLS. The discussion on both cases indicates that the assumption that the adversary can replace a legal signature.
with a bogus one cannot hold in batch verification. Therefore, batch verification is secure when $m = n + 1$. Consequently, batch verification is against bogus signature.

3) We can validate that signature aggregation scheme’s security against forgery signature with the similar security analysis. To save the space, we only demonstrate how to establish the validity of verification equation (2) in Appendix in the followings:

$$\hat{E}(\sigma_1, \ldots, \sigma_n, g) = \prod_{j=1}^{n} \hat{e}(\sigma_j, g)$$

### F. Experimental Results for Worst Case

Fig. Appendix-3 Experimental Results of $\alpha$Tree fault diagnosis algorithm to Pinpoint Bogus Signatures (1 to 50) among 100 Signatures: Worst Case