2012

Privacy and security protection in cloud integrated sensor networks

Chuang Wang
Iowa State University

Privacy and security protection in cloud integrated sensor networks

by

Chuang Wang

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:

Wensheng Zhang, Major Professor

Carl K. Chang

Yong Guan

Daji Qiao

Johnny S. Wong

Iowa State University

Ames, Iowa

2012

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to all those people who helped and inspired me during my study and research in Iowa State University. First and foremost, I would like to sincerely appreciate my advisor Prof. Zhang for his guidance, patience and support throughout the course of my PhD research and the writing of the dissertation. His insights, wisdom, enthusiasm towards scientific research and words of encouragement have often inspired me and renewed my hopes for completing the PhD program. I would also like to thank my committee members Prof. Carl K. Chang, Prof. Yong Guan, Prof. Daji Qiao, and Prof. Johnny S. Wong for their efforts and advices to my PhD research.

I am proud that I have met many friends in Iowa State University who have been part of my life. I would like to take this opportunity to express my thanks to Wei Zhang, Taiming Feng, Danyu Liu, Hua Ming, Jia Tao, Ru He, Xia Wang, Hsinyi Jiang, Hua Qin, Liyuan Xiao, Ge Xu, Jiang Tian, Jing Liu, Zi Li, Yuheng Long, Yang Peng, Xuemin Liu and Ming Jia. I would also like to thank Yaqiang Wu, Yanping Chen, Fan Yang, Si Li, Qinxue Jin, Jing Li, and Peng Liu. Their kind help and concern encouraged me to overcome the difficulties and gave me such a beautiful memory for the rest of my life.

ABSTRACT

Wireless sensor networks have been widely deployed in many social settings to monitor human activities and urban environment. In these contexts, they acquire and collect sensory data, and collaboratively fuse the data. Due to resource constraint, sensor nodes however cannot perform complex data processing. Hence, cloud-integrated sensor networks have been proposed to leverage the cloud computing capabilities for processing vast amount of heterogeneous sensory data. After being processed, the sensory data can then be accessed and shared among authorized users and applications pervasively.

Various security and privacy threats can arise when the people-centric sensory data is collected and transmitted within the sensor network or from the network to the cloud; security and privacy remain a big concern when the data is later accessed and shared among different users and applications after being processed. Extensive research has been conducted to address the security and privacy issues without sacrificing resource efficiency. Unfortunately, the goals of security/privacy protection and resource efficiency may not be easy to accomplish simultaneously, and may even be sharply contrary to each other. Our research aims to reconcile the conflicts between these goals in several important contexts. Specifically, we first investigate the security and privacy protection of sensory data being transmitted within the sensor network or from the sensor network to the cloud, which includes: (1) efficient, generic privacy preserving schemes for sensory data aggregation; (2) a privacy-preserving integrity detection scheme for sensory data aggregation; (3) an efficient and source-privacy preserving scheme for catching packet droppers and modifiers. Secondly, we further study how to address people's security and privacy concerns when accessing sensory data from the cloud.

To preserve privacy for sensory data aggregation, we propose a set of generic, efficient and collusion-resilient privacy-preserving data aggregation schemes. On top of these privacy preserving schemes, we also develop a scheme to simultaneously achieve privacy preservation and detection of integrity attack for data aggregation. Our approach outperforms existing solutions in terms of generality, node compromise resilience, and resource efficiency.

To remove the negative effects caused by packet droppers and modifiers, we propose an efficient scheme to identify and catch compromised nodes which randomly drop packets and/or modify packets. The scheme employs an innovative packet marking techniques, with which selective packet dropping and modification can be significantly alleviated while the privacy of packet sources can be preserved.

To preserve the privacy of people accessing the sensory data in the cloud, we propose a new efficient scheme for resource constrained devices to verify people's access privilege without exposing their identities in the presence of outsider attacks or node compromises; to achieve the fine-grained access control for data sharing, we design privacy-preserving schemes based on users' affiliated attributes, such that the access policies can be flexibly specified and enforced without involving complicated key distribution and management overhead.

Extensive analysis, simulations, theoretical proofs and implementations have been conducted to evaluate the effectiveness and efficiency of our proposed schemes. The results show that our proposed schemes resolve several limitations of existing work and achieve better performance in terms of resource efficiency, security strength and privacy preservation.

CHAPTER 1. Introduction

1.1 Background

A wireless sensor network [1, 2] is a collection of low-cost, small-size sensor nodes that can sense their direct environment and transmit their sensory data via wireless channels without requiring any infrastructure. Wireless sensor networks have been widely deployed in many social settings to monitor human activities and urban environment [3, 4, 5, 6, 7, 8]. For example, sensor networks may be deployed in factories, office buildings, houses and hospitals to monitor the working status of machineries, the air, light, noise, or temperature conditions of rooms, the water and electricity consumption of homes, and the health condition of human beings. Among these applications, sensor networks are essentially used as a facility for real-time data acquisition and transmission. The data from different sensor applications need to be further fused and processed such that the information is useful for human beings. However, the sensor nodes cannot perform complex data processing due to their constrained computation and storage resources. To bridge the gap between data acquisition and data processing, cloud-integrated sensor networks have been proposed [9, 10, 11, 12]. By leveraging the powerful cloud computation capabilities, vast amount of heterogeneous sensor data can be processed, analyzed, and stored. After that, the sensor information can be further accessed and shared among authorized users and applications pervasively. Figure 1.1 shows the ecosystem of the cloud-integrated sensor networks.

Along with attractive features brought by cloud-integrated sensor networks, security and privacy issues have been raised as serious concerns particularly when the sensor networks are collecting data related to daily activities of human beings [13, 14, 15]. For example, the sensitive and personal information collected from home, hospitals and offices must be secured and protected. In this context, people-centric sensory data however is vulnerable to security and privacy breach when the data is processed and transmitted within a sensor network or from the network to the cloud. Due to the constrained

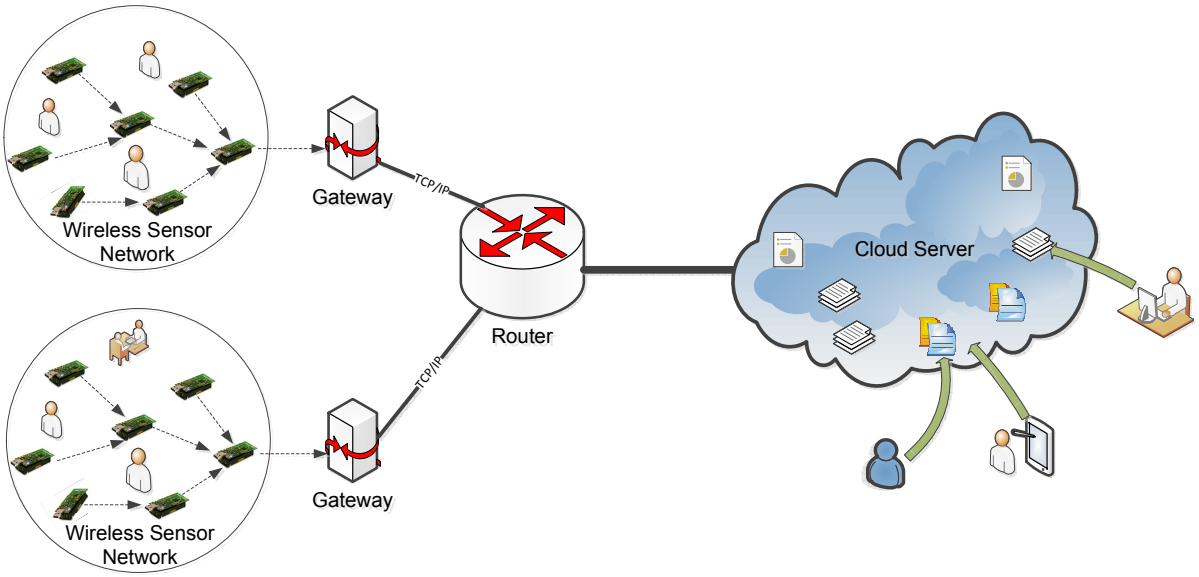


Figure 1.1 Cloud-integrated Sensor Networks

energy, computation, bandwidth and storage resources in sensor nodes, the employment of advanced privacy-preserving cryptographic primitives may be prohibitive. The people-centric sensory data is also exposed to security and privacy threats when being stored in the cloud and shared among users and applications. For instance, the sensory data may be exploited by intruders and malicious insiders to identify and obtain information related to particular users.

1.2 Research Problems

Based on the above description of the cloud-integrated sensor network system and the security and privacy threats that it faces, we identify the following important research problems.

A renowned research problem is the conflict among *in-network data aggregation* [16, 17, 18, 19, 20, 21], *data privacy protection* [22, 23, 24, 25], and *data integrity detection* [32, 33, 34]. With in-network data aggregation, each sensor node aggregates the data generated by itself and those that it receives, and forwards only the aggregated data. By employing in-network data aggregation, the amount of data communicated in the network can be significantly reduced, which consequently decreases bandwidth consumption and energy depletion. To perform in-network data aggregation, sensor nodes should be able to access data items that they forward. For this sake, sensory data may be either in plaintext, or encrypted with keys known by the forwarding nodes. However, data transmitted in plaintext can be

eavesdropped by both forwarding nodes and outsiders. Encrypting data with keys known by forwarding nodes can stop outsiders from eavesdropping but cannot prevent forwarding nodes from doing so, which conflicts the goal of privacy preservation. To solve the conflict between in-network aggregation and data privacy, data may be encrypted via homomorphic encryption so that sensor nodes are able to conduct in-network aggregation on the ciphertext [22, 23, 24, 25]. However, these schemes only support additive aggregation functions, such as sum/average, but cannot be used for other aggregation functions, such as max/min, median/percentile. Hence, a generic privacy preserving scheme for data aggregation is needed. Besides, conducting data aggregation on ciphertext may be abused by the adversaries to falsify the aggregation result at will. As discussed in references [32, 33], testing the integrity of aggregation usually needs the knowledge about the actual data being aggregated and results from aggregation, which conflicts the goal of preserving data privacy. Therefore, a desired privacy-preserving data aggregation scheme should also reconcile the conflicts between privacy preservation and integrity protection.

Identifying packet droppers and modifiers in resource constrained sensor networks remains as a challenge, which can be even more challenging when privacy preservation is factored in. An adversary may launch various attacks to disrupt in-network communication [46]. Among these attacks, two common ones are *dropping packets* [47, 59, 61] and *modifying packets* [75, 76, 77, 78, 79], i.e., compromised nodes drop or modify the packets that they are supposed to forward. Existing solutions do not consider protecting the privacy of packet sources. Without such consideration, the traffic pattern for a particular source can be easily inferred. Furthermore, the bad effects of dropping or modifying would be even more severe if the packet sources are not concealed, because the adversaries may selectively drop or modify packets from particular nodes, which may result in bigger difficulty in identifying intruders or even incorrect detection of innocent nodes. Therefore, an efficient scheme is required to identify and catch compromised nodes meanwhile conceal packet sources.

To enforce access control to the data stored in cloud and meanwhile preserve the privacy of authorized users, employing a privacy-preserving authentication is a natural approach. Numerous *privacy-preserving authentication and access control* schemes [90, 91, 92, 93, 94, 95] have been proposed for various scenarios. These schemes, however, are not designed for resource constrained devices. As the sensory data has been centralized into the cloud, authorized users should be able to access the data pervasively via their resource constrained devices, such as smart phones. Hence, a lightweight privacy-

preserving authentication scheme is desired. In addition, the access to different types of data may be regulated under different policies, and the policies may be defined dynamically. Attribute-based encryption (ABE) techniques have been widely employed to implement such diverse and dynamic access regulations. These techniques have realized privacy preservation and access control, but they do not provide accountability. Though it may be a remedy to employ these techniques together with group signature schemes, the communication efficiency, scalability and flexibility could be constrained. Therefore, it is also important to design more efficient and scalable schemes for accountable and privacy preserving access control under diverse and dynamic policies.

1.3 Overview of Our Approaches

In this section, we briefly discuss our approaches to the problems identified above: (1) efficient, generic privacy preserving schemes for sensory data aggregation; (2) a privacy-preserving integrity detection scheme for sensory data aggregation; (3) an efficient and source-privacy preserving scheme for catching packet droppers and modifiers; and (4) efficient privacy preserving authentication schemes for static groups and dynamic ad hoc groups.

1.3.1 Generic Privacy Preserving Schemes for Data Aggregation

Our generic privacy preserving schemes are inspired by existing schemes in references [22, 23, 24, 26]. In these schemes, each data item is encrypted using homomorphic encryption; hence, aggregation can be conducted on the ciphertext. However, existing homomorphic encryption schemes only support efficiently additive data aggregation, such as sum/average, but not other types of data aggregation, such as max/min, median/percentile, etc.

To address the above issue, we propose a set of generic, efficient and collusion-resilient privacy-preserving data aggregation solutions. Particularly, to preserve privacy for the queries targeted at special sensor data or sensor data distribution, we propose *perturbed histogram-based aggregation (PHA)* schemes. In the basic version of the PHA scheme (called *b-PHA*), data reported from sensor nodes are aggregated to form a histogram with a certain desirable granularity; node specific perturbations are added to the histogram such that any sensor node cannot see or infer either the histogram or the in-

dividual data items reported by sensor nodes. Based on received histograms, the sink can then derive the approximate results of particular queries. To keep the privacy preservation property of the b-PHA scheme and meanwhile reduce the bandwidth consumption, we further design advanced PHA schemes. The design is based on the idea of trading computational cost at the sink for less communication cost at sensor nodes, which is favorable for sensor networks where communication is much more costly than computation.

1.3.2 A Privacy and Integrity Protection Scheme for Data Aggregation

As mentioned above, designing a data aggregation scheme that can both preserve privacy and detect integrity attacks still remains as a challenging problem due to the apparent contradiction between the two goals. As in references [32, 33], to test the integrity of aggregation usually needs the knowledge about the actual data being aggregated and results from the aggregation, which contradict with the requirement of preserving data privacy; meanwhile, as in references [25, 26, 27], to preserve the privacy of data before and after aggregation can be abused by malicious aggregators to modify the aggregation results without being detected.

We address the above problem by proposing a simple yet effective generic aggregation scheme that can detect tampered data aggregation while preserving data privacy. Built on top of the aforementioned perturbed histogram-based privacy-preserving data aggregation scheme [27], our proposed scheme has leveraged a delicately designed mathematical structure, together with unique properties of histogram structure, the random perturbation technique, and information preloading mechanism. With the scheme, two necessary conditions derived from the unique property of histogram are leveraged to check the integrity of data. Through our scheme, each innocent non-leaf node on the aggregation tree can detect whether the data it has aggregated is tampered or not with a certain probability affected by several system parameters such as the percentage of intruders, the average scale of each malicious data modification, and so on. However, the detector cannot gain any knowledge about the actual data through the detection activities.

1.3.3 Catching Packet Droppers and Modifiers

We propose an efficient scheme to identify and catch packet droppers and modifiers without disclosing packet sources. According to the scheme, a dynamic routing tree rooted at the sink is first established. When sensor data is transmitted along the tree structure towards the sink, each packet sender or forwarder adds a small number of extra bits, called packet marks, to the packet. The format of the small packet marks is deliberately designed such that the adversary is unable to find out the source of packets, but the sink can obtain very useful information from the marks. Specifically, based on the packet marks, the sink can figure out the dropping rate associated with every sensor node, and then run our proposed *node categorization algorithm* to identify nodes that are droppers/modifiers for sure or are suspicious droppers/modifiers. As the tree structure dynamically changes every certain time interval, behaviors of sensor nodes can be observed in a large variety of scenarios. As the information of node behaviors has been accumulated, the sink periodically runs our proposed *heuristic ranking algorithms* to identify the most likely bad nodes from suspiciously bad nodes. This way, most of the bad nodes can be gradually identified with small false positive.

1.3.4 Accountable, Privacy Preserving Authentication Scheme for Static Groups

Many accountable privacy preserving authentication schemes [90, 91, 92, 93, 94, 95] have been proposed for controlled access to online services. Many of them are based on computationally intensive bilinear maps [103, 104, 109, 110]. Instead, we propose a Lightweight Accountable and Anonymous Authentication Scheme (LA^3) based on computationally-lightweight elliptic curves. It offers similar security properties as group signature does, namely, *Non-frameability*, *Traceability*, and *Selfless Anonymity*. The proposed LA^3 scheme assumes three types of entities in the system: a service provider (called verifier) that needs to verify whether a client has the privilege to access its service; a group of clients (called provers) that need to prove their access privileges; and a trusted authority responsible for choosing system parameters and initializing the verifier and provers. Following the protocol of LA^3, a prover and the verifier can interact with each other in an authentication transaction after they have been initialized by the authority. The authentication process involves only a few operations over a multiplicative cyclic group and a finite field, in addition to a small number of message exchanges. The

prover can keep anonymous to the verifier; but the authority is able to trace out the prover based on the authentication transcript when necessary.

1.3.5 Accountable, Privacy Preserving Authentication Scheme for Ad Hoc Groups

This research component is to meet the afore-mentioned demand for efficient and scalable schemes for accountable and privacy-preserving access control under diverse and dynamic policies. Particularly, we propose a new group signature scheme, named AdHocSign, for ad hoc groups that are defined dynamically according to the diverse access structure (i.e., access policy) of each shared resource, which is typically expressed as a combination of logic conjunctions and disjunctions of attributes. In this scheme, when certain data is posted to a host, the host is given certain auxiliary information that is computed by a trusted authority according to the access structure of the data. The auxiliary information serves as the public key of the ad hoc group. When a user needs to access a piece of data, it contacts the host of the data to obtain the access structure of the data and the afore-mentioned auxiliary information pre-loaded to the host by the authority. If the user's attributes satisfy the access structure (i.e., the user is a member of the ad hoc group defined by the access structure), the user can compute his/her own private key for the ad hoc group based on the auxiliary information and pre-loaded key materials, and authenticates himself/herself to the host. The user does not expose his/her identity or ownership of attributes during the authentication.

1.4 Organization

In the rest of this dissertation, related work is summarized in Chapter 2. Chapter 3, Chapter 4, and Chapter 5 present our privacy preserving schemes [27, 42, 83, 84] for collecting and delivering sensory data into the cloud. Chapter 6 and Chapter 7 describe our proposed schemes [97, 138] for protecting the privacy of accessing sensory data within the cloud. Chapter 8 summarizes the research results in this dissertation and discusses the future work.

CHAPTER 2. Related Work

In this chapter, we discuss the work related to our investigated problems: namely, privacy and integrity protection for sensory data aggregation, countermeasures for packet dropping and modifying attacks, and accountably privacy preserving authentication schemes.

2.1 Data Aggregation in Sensor Networks

Data aggregation [16, 17, 18, 20] is essential to save the energy expenditure for wireless sensor networks. With in-network data aggregation, every sensor node processes multiple raw sensory data items that it produces, or that it receives and is expected to forward. Here, typical aggregation functions can be classified as additive based aggregation (such as sum/average, count) and histogram based aggregation (such as max/min, percentile/median, and so on [16]). After being processed, only the aggregation result is transmitted. In this way, the amount of data communicated in the network can be decreased, which consequently reduces the communication bandwidth and energy consumption.

2.1.1 Integrity Protection Schemes for Data Aggregation

Due to the open nature of wireless communication channels and the lack of physical protection of individual sensor nodes which make the adversary easy to eavesdrop the communication and compromise sensor nodes, sensor networks have inherent security vulnerabilities. Many researchers have proposed schemes to detect the data modification attack and further filter the modified data within a certain number of hops [75, 76, 77, 78, 82]. However, these schemes cannot be used for integrity checking in data aggregation since the intermediately aggregated data keep being updated hop by hop, while these schemes are proposed for the application that the data will not be changed during the transmission. To detect the alteration of intermediately aggregated result, Hu and Evans [37] proposed a

secure hop by hop protocol to aggregate the data. However, it works only when two consecutive nodes are not compromised. Yang *et al* [33] proposed a probability based integrity checking scheme for the aggregation result. During the data aggregation phase, dynamic logic groups are formed. For each group, a hop by hop commitment (MAC) is computed and sent to the base station along with the data aggregation result within this group. The base station then launches the integrity verification attestation on suspicious groups based on the group aggregation results. Chan *et al* [32] proposed a scheme, which is guaranteed to detect any malicious modification of aggregation result. In this scheme, each sensor node checks whether its data contribute to the final aggregation result correctly in a distributed way based on the received off-path value, and then sends the verification result to the base station securely. Its communication overhead has been improved in reference [34]. FAIR [35] is a resilient aggregation framework that provides data integrity by introducing multiple witness nodes. By computing a value of information quality via fuzzy inference model, FAIR finally returns the accuracy level of aggregation result. Zhang *et al* [36] proposed a statistic approach to let the sink detect whether the aggregated data has been illegitimately altered based on watermarking technique. In this scheme, the sensory data from the network is regarded as an image, in which every sensor node is viewed as a pixel with its sensory data representing the pixel's intensity. Investigation to localize and remove compromised nodes in data aggregation are also conducted in references [38, 39].

2.1.2 Privacy Preserving Schemes for Data Aggregation

As sensor network applications expand to include increasingly sensitive measurement of everyday life, preserving data privacy becomes an increasingly important concern. For example, the details of household such as power and water usage will be collected through sensor networks. Without proper privacy protection, such applications will not be favored by consumers since people are becoming more and more alert to their privacy. Extensive research hence has been conducted to preserve the data privacy when performing data aggregation. Mykletun *et al* [29] proposed an end-to-end privacy preservation data aggregation. However, this scheme is based on additive homomorphic public key encryption, which is still too costly for current energy constrained sensor nodes. Existing literatures [22, 23, 24] proposed schemes to protect the end-to-end data privacy and confidentiality of aggregated data. Instead of using public key encryption, the symmetric additive homomorphic encryption is applied. Feng *et al* [26] pro-

posed a series of schemes to optimize the communication overhead along this line of research. Later, He *et al* [25] proposed two privacy-preserving data aggregation schemes, namely, the cluster-based private data aggregation (CPDA) scheme and the slice-mix-aggregate (SMART) scheme, for additive aggregation functions, both of which can only tolerate up to a certain threshold number of compromised nodes. Although the threshold can be raised by expanding the size of cluster for CPDA or increasing the number of slices for SMART, it will result in higher communication overhead. Moreover, one common limitation of existing schemes [22, 23, 24, 25, 26] is that they are only applicable for additive aggregation. It cannot perform other common aggregation functions such as max/min, percentile/median, and so on. The negative survey technique was used to collect sensitive data in reference [30]. Instead of transmitting the actual data, each node randomly selects a data item from the actual data's complement set, and transmits the complement data to the sink. The sink can recover the histogram of original sensor readings based on the negative samples. Ganti *et al* [31] proposed an architectural component (PoolView) for providing privacy guarantees on aggregated information of interests. PoolView deals with the time series data. It relies on client side data perturbation to break the data correlation so as to ensure individual's privacy. Meanwhile, the reconstruction of application specific statistic data can be performed at the server side with bounded accuracy.

2.1.3 Privacy and Integrity Synergy Schemes for Data Aggregation

Existing secure data aggregation schemes are aimed to either detect the data integrity attack or preserve the data privacy. Very few works focus on preserving the data privacy meanwhile guaranteeing the data authenticity. He *et al* [41] firstly studied this problem by proposing a scheme called iPDA, which aims to simultaneously address the privacy and the integrity issues in aggregation. The privacy preservation is achieved by the slicing technique [25]. The data integrity issue is dealt with by asking each sensor node to report their sensory data to two disjoint aggregation trees. The sink can detect data modification if aggregation results from two trees do not agree with each other. However, iPDA is a threshold based privacy preserving scheme, which can only tolerate a small number of node compromises; data modification cannot be detected in many situations, for example, when two aggregators from the disjoint trees have been selectively compromised and do the same modification. iCPDA [40] was also designed to achieve the synergy of data privacy and integrity. The detection of integrity attacks

is achieved via peer monitoring, and the scheme only protects the privacy of raw data. The privacy of intermediate aggregation results cannot be preserved through iCPDA since the results should be used for peer monitoring.

2.2 Countermeasures for Packet Dropping and Modification Attacks

The wireless sensor networks are often deployed in an unattended and hostile environment to perform the monitoring and data collection tasks. When it is deployed in such an environment, it lacks physical protection and is subject to node compromise. After compromising one or multiple sensor nodes, an adversary may launch various attacks [85] to disrupt the in-network communication. Among these attacks, two common ones are *dropping packets* and *modifying packets*, i.e., compromised nodes drop or modify the packets that they are supposed to forward. Many schemes have been proposed to identify the packet dropping attack in Internet [68, 69, 71, 72, 73, 74]. However, these schemes cannot be applied to wireless sensor networks due to their limited capabilities for data acquisition and processing.

2.2.1 Countermeasures to Detect Packet Dropping Attacks

The countermeasures to detect packet dropping attack can be categorized as three approaches: multi-path forwarding approach, acknowledgement approach, and neighbor monitoring approach. Based on delivering redundant packets along multiple paths, multi-path forwarding [47, 50] is a widely adopted countermeasure to mitigate packet droppers. By obtaining responses from intermediate nodes, acknowledgment approach [65, 66, 87] is also used to detect packet droppers. The neighbor monitoring approach (i.e., the *watchdog* method) was originally proposed to mitigate routing misbehavior in mobile ad hoc networks [51], and then adopted to identify packet droppers in wireless sensor network [52, 53, 54]. When the watchdog mechanism is deployed, each node monitors its neighborhood promiscuously to collect the firsthand information from its neighboring nodes. A variety of reputation systems have been designed by exchanging each node's firsthand observations, which are further used to quantify node's reputation [55, 56, 57, 58]. Based on the monitoring mechanism, the intrusion detection systems are proposed in references [60, 61]. However, the watchdog method requires nodes to buffer the packets

and operate in the promiscuous mode. The storage overhead and energy consumption may not be affordable for sensor nodes. In addition, this mechanism relies on the bidirectional communication links, and hence, it may not be effective when directional antennas are used [62]. Particularly, this approach cannot be applied when the node does not know the expected output of the next hop since the node has no way to find a match for buffered packets and overheard packets. Note that, this scenario is not rare. For example, the packets may be processed, and then encrypted by the next hop node in many applications where security is required. Since the watchdog is a critical component of reputation systems, the limitations of the watchdog mechanism can also limit the reputation system. Besides, a reputation system itself may become the attacking target. It may either be vulnerable to bad mouthing attack or false praise attack [62].

2.2.2 Countermeasures to Detect Packet Modification Attacks

To deal with packet modifiers, most of existing countermeasures [75, 76, 77, 78] are to filter modified messages within a certain number of hops so that energy will not be wasted to transmit modified messages. The statistical en-route filtering (SEF) scheme [75] randomly preloads each sensor node with k secret keys from one of n partitions from the key pool. A stimulus report is endorsed by T nodes with T MACs. The interleaved hop-by-hop authentication scheme [76] relies on the periodic association process to make each node establish the pairwise keys with others which are $t + 1$ hops away. Each stimulus should be reported by multiple sensor nodes, and the cluster head forms the final report which contains $t + 1$ distinct MACs. When a message is forwarded along a path to the receiver, an en-route node may use its shared secret key to verify one of the MACs carried by the message. The modified messages will be filtered out whenever the MAC verification fails. The dynamic en-route scheme proposed by Yu and Guan [78] takes similar ideas of multiple MACs to filter the false message. Each node periodically updates its keys from a one-way hash chain. The authentication keys are disseminated by the cluster head to the forwarding nodes using hill climbing technique. Each forwarding node then validates the authenticity of the message and drops the modified one. Compared to references [75, 76], it achieves better filtering performance and could deal with the dynamic network topology in sensor networks. However, the common drawback of existing schemes [75, 76, 77, 78] is that each secret key is shared by multiple nodes and therefore, these schemes become ineffective or even useless if a large

number of nodes are compromised.

2.2.3 Countermeasures to Catch Packet Droppers and Modifiers

The effectiveness to detect malicious packet droppers and modifiers would be limited without identifying them and excluding them from the network. Researchers hence have been proposed schemes to localize and identify packet droppers. One approach is to employ the acknowledgement based scheme [68, 69, 70] to identify the problematic communication links. It can deterministically localize links to malicious nodes when every node reports ACK using onion report. However, this incurs large communication and storage overhead for sensor networks. The probabilistic ACK approaches [69, 70] are then proposed, which seek tradeoffs among detection rate, communication overhead and storage overhead. However, these approaches assume the packet sources are trustable, which may not a valid assumption in sensor networks as the base station typically is the only one that we can trust in sensor networks. Furthermore, these schemes need to set up pairwise keys among regular sensor nodes so as to verify the authenticity of ACK packets, which may cause considerable overhead for key management in sensor networks. A scheme called PNM [79] was designed to identify packet modifiers with a certain probability. However, the PNM scheme cannot be used together with the false packet filtering schemes [75, 76, 77, 78], because the filtering schemes will drop the modified packets which should be used by the PNM scheme as evidences to infer packet modifiers. This degrades the efficiency of deploying the PNM scheme.

2.3 Accountable Privacy Preserving Authentication Schemes

Group signature [98, 99, 100, 103, 104] has been widely used to as the cryptographic primitive to design privacy preserving authentication schemes [90, 91, 92, 93, 94, 95]. It allows a group member to sign a message anonymously on behalf of the group. Besides, it also allows the group manager to trace out who has signed the message, and thereafter group signature based schemes are capable to maintain accountability such that malicious users cannot abuse the privacy preservation schemes.

2.3.1 The State-of-the-art Group Signature Schemes

Since Chaum and Van Heyst introduced the group signature concept in [98], researchers have proposed a large number of group signature schemes [99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118].

Based on the cryptographic assumptions they rely on, these group signature schemes can be briefly summarized as follows. Some schemes [100, 101, 102, 107, 108] are based on the strong RSA and decisional Diffie-Hellman assumptions. An exemplified scheme was proposed by Ateniese *et al* [100], which was improved to support membership revocation in [101, 102] at the cost of degraded efficiency. Camenisch and Groth [107] then proposed a scheme with an order of magnitude better efficiency and supportability of revocation. Later on, Boneh *et al* designed short group signature schemes with approximately the same size of standard RSA signature based on bilinear maps [103, 104]. The security of the design relies on the strong Diffie-Hellman and the Decision Linear assumptions. Since then, various other group signatures built on bilinear maps have been proposed in [106, 115, 109, 119, 116, 117, 118, 110, 113, 114]. Particularly, schemes [109, 119] are designed to provide backward unlinkability for revoked users, i.e., even a member is revoked, signatures generated by this member before the revocation remain anonymous. Scheme [110] is proposed for scenarios where the membership of dishonest users can be revoked, but their identities remain anonymous after revocation. This is useful for anonymous authentication in anonymous BBS, blogs and so on.

Efficient revocation has been one research challenge in designing group signature scheme. In early group signature schemes, when revocation occurs, the authority needs to change the group public key, and redistributes the secret keys of all non-revoked members [99, 100]. An improved revocation mechanism was to broadcast a message to all signers and verifiers such that non-revoked users can update their secret keys while revoked users cannot [102, 104, 107]. Recently, schemes [113, 114], in which the revocation overhead is constant independently of the number of revoked members, have been proposed. However, the feature of constant revocation overhead in scheme [113] is achieved in the cost of large storage overhead, as the size of its public key is $O(\sqrt{N})$, where N is the total number of group members. Scheme [114] shifts the quadratic computational cost $O(NR)$ to the group manager, where N and R are the total number of group members and number of revoked members; both the

signer and verifier then need to update some information computed by the group manager for generating and verifying signature respectively. With a different approach, revocation can only affect the verifier [101, 103, 109, 110], provers do not need to know the revocation, and the authority needs not to be contacted by provers for revocation; however, the cost for verifying a signature has to be linearly dependent on the number of revoked users. Such approach is called verifier local revocation (VLR).

2.3.2 Accountable, Privacy Preserving Authentication Schemes for Static Groups

As group signature schemes provide anonymity and traceability, numerous accountable and anonymous authentication schemes have been proposed based on the application of group signature schemes. For example, EPID [90], AnonySense [91], PEACE [92], and schemes [93, 94, 95] are proposed on top of the short group signature schemes [103, 104] devised by Boneh *et al*.

EPID [90] is a privacy preserving authentication scheme for hardware devices. Builds on top of group signature scheme proposed in references [104, 105], EPID enables the hardware devices to prove their membership to a verifier via a group signature, the verifier can verify the membership of hardware devices without knowing their identity. In EPID, the private key is embedded into the hardware device; nobody except the hardware itself knows the private key. EPID does not support traceability for the group manager so as to provide the maximum privacy for the hardware devices. To revoke the compromised devices, EPID can revoke a device based on its private key when it is available for the group manager, i.e., the private key is extract from the hardware and available publicly. EPID can also revoke a device based on the signature issued by the device when its private key is not known to the group manager. AnonySense [91] is a privacy preserving framework to encourage people to take part in the collaborative and opportunistic task reporting on their surroundings via their mobile devices and carry-on sensors. Leveraging the group signature scheme [104], AnonySense allows applications to submit the collected sensor data while ensuring the anonymity of users. Furthermore, AnonySense ensures that multiple reports from the same user are unlinkable even under timing attacks on reported data.

In wireless mesh networks, Ren and Lou [92] proposed a security framework, called PEACE. PEACE enforces strict user access control to cope with both free riders and malicious users; on the other hand, PEACE offers user privacy protection against both adversaries and various other network entities. PEACE supports the mutual authentication between user and mesh router or user and user.

Based on the group signature technique [103], PEACE achieves the privacy preservation and accountability simultaneously.

In vehicle networks, privacy preserving schemes [93, 94] are proposed based on group signature technique. Lu *et al* [95] further proposed the ECPP protocol by jointly considering the efficiency of key storage, safety message verification and the traceability of malicious OBUs. The proposed protocol is characterized by the generation of on the fly short-time anonymous keys between OBUs and RSUs, which provides fast accountably anonymous authentication while minimizing the required storage for short time anonymous keys.

Built upon group signature primitive, these schemes [90, 91, 92, 93, 94, 95] need to conduct pairing and exponentiation operations over bilinear groups, which are computationally expensive. Hence, considerable delay and computational overheads may be introduced when the prover and/or the verifier are resource-constrained devices.

2.3.3 Accountable, Privacy Preserving Authentication Schemes for Ad Hoc Groups

Researchers have proposed attribute based encryption (ABE) [129, 130, 131] schemes and attribute based signature (ABS) [136, 137] schemes. With ABE, data are encrypted based on their access structures, where each access structure is a logical expression over attributes. A user is able to decrypt a piece of data if and only if she owns the access attributes that satisfy the access structure. Systems such as Persona [132] have been proposed to employ ABE for access control of shared outsourced data. With ABS, data's access privilege is also defined over access structures, and each user accessing the data is required to attest its access privilege by submitting a signature. The signature attests that the signer possesses certain attributes which satisfy the access structure. The signature guarantees the privacy of the signer as it reveals nothing about the identity or attributes of the signer. Both ABE and ABS approaches provide anonymous fine-grained access control of shared data, and the data host does not know users' identities when they access the shared data. These approaches however may not be feasible in certain scenarios where accountability is required, which allows the trust authority to trace out misconduct users and stop them from abusing the privacy preservation features. Attribute based group signature (ABGS) schemes [135, 134] are further designed based on existing group signature schemes [103, 104]. With ABGS, a signature authenticates a user belonging to a group and possessing

certain attributes. Particularly, it allows the trusted authority to reveal the identity of the signer in case of misbehaviors appearing. Unfortunately, ABGS schemes introduce considerable overhead for signature generation and verification when applied for resource constrained devices. For example, the ABGS scheme with revocation [134] is built upon group signature scheme called BS [103]. It inherits not only the overhead of BS scheme, but also involves extra pairing and exponentiation operations over bilinear groups, which is proportional to the number of attributes satisfying the access structure. The length of signatures also increases linearly. The ABGS schemes also lack the property of signer attribute privacy as the verifier needs to know what attributes are used by the signer during the verification process. Camenisch *et al* [133] proposed a scheme for oblivious data transfer with access control, which is also built upon group signature scheme. However, different from our proposed AdHocSign, the purpose of the scheme is not to provide accountable group signature mechanisms for dynamic groups, but to protect the confidentiality of data access policies as well as what data are accessed by users.

CHAPTER 3. Generic Privacy Preservation Solutions for Approximate Aggregation of Sensor Data

In-network data aggregation has been a well adopted practise for saving energy consumption when sensory data is transmitted to the sink. Preserving privacy during the course of in-network data aggregation poses new challenges due to the potential incompatibilities between the goals of privacy-preserving mechanisms and in-network data aggregation. To address this problem, we propose a set of new privacy-preserving data aggregation schemes. Different from existing work, our solutions have the following features: supporting a variety of aggregation functions; providing privacy protection for both individual data and aggregated data; being resilient to any number of node collusion; being highly efficient.

3.1 Introduction

As the increasing civilian sensing applications enabled by wireless sensor networks, the concern of private information leakage has also been raised. Existing schemes designed for achieving resource efficiency in sensor networks may not be compatible with the raised privacy concern. A renowned example is the conflict between *in-network data aggregation* [16, 32, 17, 18, 19, 20, 21] and *data confidentiality (privacy) protection* [22, 23, 24, 25]. With in-network data aggregation, the amount of data communicated in the network can be decreased, which consequently reduces the bandwidth consumption and the energy depletion for communication. However, to enable in-network data aggregation, sensor nodes should be able to access data items that they forward. For this sake, sensory data may be either in plaintext, or encrypted with keys that known by the forwarding nodes. However, data transmitted in plaintext can be eavesdropped by both forwarding nodes and outsiders. Encrypting data with keys known by forwarding nodes can stop outsiders from eavesdropping but cannot prevent forwarding nodes from doing so.

The problem of resolving the aforementioned incompatibility has been studied in [22, 23, 24, 25, 26]. However, all the schemes only support additive data aggregation, but cannot protect other types of data aggregation functions such as Max/Min, Percentile/Median. To address the above issues, we propose in this paper a set of generic, efficient and collusion-resilient privacy-preserving data aggregation solutions. Particularly, to preserve privacy for the queries targeted at special sensor data or sensor data distribution (e.g., Max/Min, Sum/Average, Percentile/Median, and so on), we propose in this paper *perturbed histogram-based aggregation (PHA)* schemes. In the basic version of the PHA scheme (called *b-PHA*), data reported from sensor nodes are aggregated to form a histogram with a certain desirable granularity; perturbations are added to the histogram such that any sensor node cannot see or infer either the histogram or the individual data items reported by sensor nodes. Based on received histogram, the sink can then derive the approximate results of particular queries such as MIN/MAX, Sum, Median, etc. To keep the perfect privacy preservation property of the b-PHA scheme and meanwhile reduce the bandwidth consumption, we further design advanced PHA schemes, namely, a function-assisted PHA scheme and a hybrid PHA scheme. The design is based on the idea for trading computational cost at the sink for less communication cost at sensor nodes, which is favorable for sensor networks where communication is much most costly than computation. Simulations are conducted to evaluate our proposed schemes. The results show that our proposed advanced schemes achieve nearly-ideal efficiency.

3.2 System Model

Network Assumptions We consider a sensor network that consists of N sensor nodes, each with a unique ID picked from $\{1, \cdots, N\}$, and a sink (e.g., a base station). Each sensor node monitors its direct environment and generates data. The sink is aware of the number and IDs of currently alive sensor nodes, and has much powerful computation, storage and communication capabilities than sensor nodes.

Each sensor data item is an integer ranging from 0 to some upper bound denoted as U_d. Note that, even though some data (e.g., temperature, humidity, noisiness, etc.) may not be integer in its original form, they can be transformed to integers. Sensor nodes and the sink form a tree. Responding to each query broadcast by the sink via the tree downwards, sensor nodes forward their replies back to the sink

via the tree upwards. During the course of forwarding replies, each non-leaf sensor node aggregates the data from all of its descendants together with its own data, and only forwards the aggregation result to its parent node. We assume that if some sensor nodes fail to reply, this will be detected by some upstream nodes on the tree, and the IDs of these fail-reporting sensor nodes will be sent to the sink. Therefore, our proposed schemes do not consider this issue.

Security Assumptions and Design Goals We assume that the sink is trustworthy while any sensor node could be compromised. After a sensor node is compromised, it may attack the network arbitrarily. Since this paper focuses on addressing the incompatibility between in-network data aggregation and data privacy protection, we only consider the attacks that *outsiders or compromised sensor nodes eavesdrop sensor data, and/or reveal the data they receive/forward to the adversary*. For other security issues in data aggregation, readers may refer to [33, 32, 43].

Design Goals We aim to achieve privacy preservation by accomplishing the following objectives.

- *Privacy/confidentiality of querying results*: The data as queried results shall not be exposed to any sensor node. For example, in the query for MIN/MAX, the actual minimum or maximum data item returned shall be kept secret from any sensor node; in the query for sensor data histogram, the actual distribution of sensor data (histogram) shall also be kept secret from any sensor nodes.

- *Privacy of raw and intermediately-aggregated data*: The data reported by any individual sensor node (we call raw data) shall be known only by the sensor node itself; this also implies that the links between reported data items and the reporting sensor nodes shall be kept secret. The actual intermediately-aggregated results (i.e., the aggregated result of the data reported by any set of sensor nodes) shall be kept secret from any sensor node.

- *Efficiency*: The designed privacy-preservation schemes shall incur as low overhead as possible. Based on our system assumption, the sink has much powerful computation, storage and communication capabilities than sensor nodes. Therefore, our designs are more concerned the overhead at sensor nodes rather than that at the sink.

3.3 The Proposed Privacy-Preserving Data Aggregation Schemes

In this section, we present privacy-preserving data aggregation schemes to support queries targeted at sensor data, for example, *querying the minimum/maximum value of all sensor readings (MIN/MAX)*, *querying the median value of all sensor readings (Median)*, and *querying the distribution of all sensor readings (Histogram)*.

3.3.1 b-PHA: Basic Perturbed Histogram-based Aggregation

With b-PHA, a data-targeted query is performed in two steps: first, the distribution of all sensor readings (i.e., sensor data histogram) is queried; second, the answer to the particular query is computed based on the histogram. During the query, perturbation technique is applied to hide the actual individual readings and the actual aggregate results sent by sensor nodes. In the following, we use the example of querying the median value among all sensor readings to describe the scheme.

System Preparation before Node Deployment The b-PHA scheme requires that, every sensor node (say, node u, where u is the unique ID of the node) is preloaded with a unique secret number, denoted as s_u. Here, s_u is known exclusively by the sink and the node u itself. In addition, each sensor node is preloaded with a secure one-way hash function, denoted as $h(.)$, which maps a bit string to a value between 0 and $N-1$.

Query Launch at Sink Suppose the sink wants to query the median value of all sensor readings with an accuracy requirement that, the difference between the queried median value and the actual value should not be greater than $\frac{\sigma}{2}$. The sink sends out the query message. $\langle Query, \sigma, X \rangle$, where X is a nonce uniquely associated with this query, and it is used to prevent the adversary from launching replay attacks.

Data Replying and Aggregation at Sensor Nodes Upon receiving the above query message, each sensor node u responds differently based on whether or not it is a leaf node.

Case I (if node u is a leaf node). Node u responds by replying the following message: $\langle Reply, \overline{N}_{u,0}, \cdots, \overline{N}_{u,n-1} \rangle$, where $n = \lceil \frac{U_d}{\sigma} \rceil$, and each $\overline{N}_{u,i}$ $(i = 0, \cdots, n-1)$ is computed as follows:

- If the reading at node u is in range $(i * \sigma, (i+1) * \sigma]$ (or $[0, \sigma]$ for $i = 0$), then $\overline{N}_{u,i} = [1 + h(u|X|i)]$ MOD N.
- Otherwise, $\overline{N}_{u,i} = h(u|X|i)$ MOD N.

Case II (if node u is not a leaf node). Node u first waits until it has received replies from all its children, denoted as $\langle Reply, \overline{N}_{j,0}, \cdots, \overline{N}_{j,n-1} \rangle$, where $j = 0, \cdots, m_u - 1$ and m_u is the number of children of node u. Then, node u replies message $\langle Reply, \overline{N}_{u,0}, \cdots, \overline{N}_{u,n-1} \rangle$, where each $\overline{N}_{u,i}$ $(i = 0, \cdots, n-1)$ is computed as follows:

- If the reading at node u is in range $(i * \sigma, (i+1) * \sigma]$ (or $[0, \sigma]$ for $i = 0$), then $\overline{N}_{u,i} = [1 + h(u|X|i) + \sum_{j=0}^{m_u - 1} \overline{N}_{j,i}]$ MOD N.
- Otherwise, $\overline{N}_{u,i} = [h(u|X|i) + \sum_{j=0}^{m_u - 1} \overline{N}_{j,i}]$ MOD N.

Post-Query Processing at Sink Suppose the sink has m_0 children nodes, and thus it will receive m_0 messages denoted as $\langle Reply, \overline{N}_{j,0}, \cdots, \overline{N}_{j,n-1} \rangle$, where $j = 0, \cdots, m_0 - 1$. Based on these messages, the sink can figure out the distribution of sensor readings. Specifically, the number of sensor readings belonging to range $(i * \sigma, (i+1) * \sigma]$, denoted as N_i $(i = 0, \cdots, n-1)$, is

$$\{ \sum_{j=0}^{m_0 - 1} \overline{N}_{j,i} - \sum_{u=1}^{N} h(u|X|i) \} \text{ MOD } N.$$

From the obtained distribution of sensor readings, the median value can be found out as

$$(k + \frac{1}{2}) * \sigma, s.t., \sum_{j=0}^{k-1} N_j < \frac{N}{2} \wedge \sum_{j=k}^{n} N_j \geq \frac{N}{2}.$$

Note that, the actual median value must be in the range of $(k * \sigma, (k+1) * \sigma]$. Therefore, the difference between $(k + \frac{1}{2}) * \sigma$, the obtained median value, and the actual median value is not greater than $\frac{\sigma}{2}$.

Discussion With b-PHA, each sensor node needs to forward a $reply$ message, of which the size is

$$n * \lceil log_2 N \rceil \tag{3.1}$$

Table 3.1 Comparing the bandwidth consumption of b-PHA against that of the ideal design (Note: for each item x(y), x represents the consumption of b-PHA and y represents the consumption of the ideal design; the unit is bit.)

	N=128	N=256	N=512	N=1024
n=16	112 (70)	128 (85)	144 (101)	160 (116)
n=32	224 (112)	256 (142)	288 (172)	320 (204)
n=64	448 (172)	512 (227)	576 (286)	640 (347)
n=128	896 (251)	1024 (348)	1152 (458)	1280 (575)

bits. On the other hand, the total number of possibilities for distributing N sensor readings into n intervals is $\binom{N+n-1}{n}$. That is, we need at least, and ideally it is possible to use only

$$log_2[\binom{N+n-1}{n}] \tag{3.2}$$

bits to represent the distribution of sensor readings. As shown in Table 3.1, the b-PHA scheme consumes much higher bandwidth than this ideal design. To shorten the performance gap and improve the bandwidth efficiency, we propose new PHA schemes in the following.

3.3.2 f-PHA: Function-aided Perturbed Histogram-based Aggregation

The motivation for designing f-PHA is to reduce the amount of information that sensor nodes have to send to the sink, and meanwhile, it is still guaranteed that the sink can figure out the data distribution. For this purpose, in f-PHA, sensor nodes collaboratively aggregate some equations to the sink, instead of directly sending the histogram. After obtaining the equations, the sink then solves the equations to discover the distribution information. Following the notations used in the b-PHA scheme, we let N be the total number of sensor readings and n be the number of ranges. In addition, let N_i ($i = 0, \cdots, n-1$) be the number of readings in range i. The f-PHA includes the following steps:

System Preparation before Node Deployment Every sensor node u is preloaded with a unique secret number s_u. In addition, the sink constructs a certain number (denoted as system parameter α) of n-variable linear functions denoted as $f_i(x_0, \cdots, x_{n-1})$, where $i = 0, \cdots, \alpha - 1$. Specifically,

$$f_i(x_0, \cdots, x_{n-1}) = \sum_{j=0}^{n-1} a_{i,j} x_j, \tag{3.3}$$

where each $a_{i,j}$ is a number randomly picked from $\{0, \cdots, 2^\gamma - 1\}$, and γ is another system parameter. All these coefficients are preloaded to every sensor node. Finally, each sensor node is preloaded a secure one-way hash function $h(.)$, which maps a string to a number belonging to $\{0, \cdots, N * 2^\gamma - 1\}$.

Query Launch at Sink This step is the same as that in b-PHA.

Data Replying and Aggregation at Sensor Nodes Upon receiving the above query message, each sensor node u responds in one of the following ways.

Case I (if node u is a leaf node). Node u replies message $\langle Reply, \overline{N}_{u,0}, \cdots, \overline{N}_{u,\alpha-1} \rangle$, where each $\overline{N}_{u,i}$ $(i = 0, \cdots, \alpha - 1)$ is computed as follows.

- If the reading at node u is in range $(j * \sigma, (j+1) * \sigma]$ (or $[0, \sigma]$ for $j = 0$), then $\overline{N}_{u,i} = [a_{i,j} + h(u|X|i)]$ MOD $(N * 2^\gamma)$.
- Otherwise, $\overline{N}_{u,i} = h(u|X|i)$ MOD $(N * 2^\gamma)$.

Case II (if node u is not a leaf node). Node u first waits until it has received replies from all of its m_u children. Let these replies be $\langle Reply, \overline{N}_{j,0}, \cdots, \overline{N}_{j,\alpha-1} \rangle$, where $j = 0, \cdots, m_u - 1$. Then, node u replies message $\langle Reply, \overline{N}_{u,0}, \cdots, \overline{N}_{u,\alpha-1} \rangle$, where each $\overline{N}_{u,i}$ $(i = 0, \cdots, n-1)$ is computed as follows:

- If the reading at node u is in range $(k * \sigma, (k+1) * \sigma]$ (or $[0, \sigma]$ for $k = 0$), then $\overline{N}_{u,i} = [a_{i,k} + h(u|X|i) + \sum_{j=0}^{m-1} \overline{N}_{j,i}]$ MOD $(N * 2^\gamma)$.
- Otherwise, $\overline{N}_{u,i} = [h(u|X|i) + \sum_{j=0}^{m-1} \overline{N}_{j,i}]$ MOD $(N * 2^\gamma)$.

Post-Query Processing at Sink Similar to the b-PHA scheme, the sink will receive from its children m_0 messages: $\langle Reply, \overline{N}_{j,0}, \cdots, \overline{N}_{j,\alpha-1} \rangle$ $(j = 0, \cdots, m_0 - 1)$. Let

$$c_i = \{ \sum_{j=0}^{m_0-1} \overline{N}_{j,i} - \sum_{u=1}^{N} h(X|u|i) \} \text{ MOD } (N * 2^\gamma),$$

where $i = 0, \cdots, \alpha - 1$. Then, we can get the following system of linear equations

$$\sum_{j=0}^{n-1} a_{i,j} N_j = c_i, \ i = 0, \cdots, \alpha - 1,$$

where N_j ($j = 0, \cdots, n-1$) are n non-negative integers, each represents the number of sensor readings in range $(j * \sigma, (j+1) * \sigma]$. In addition, $\sum_{j=0}^{n} N_j = N$.

Table 3.2 Choice of parameter α (s.t. the sink can identify a unique sensor reading distribution with a probability $\geq 99\%$; system parameter γ is fixed at 5)

	N=16	N=32	N=64
n=16	$\geq$4	$\geq$5	$\geq$9
n=32	$\geq$5	$\geq$7	$\geq$12
n=64	$\geq$7	$\geq$9	$\geq$14
n=128	$\geq$9	$\geq$12	N/A

When the system parameter α is large enough, the sink can find out N_j ($j = 0, \cdots, n-1$) correctly with high probability. Table 3.2 shows the results we obtained from experiments. For example, if $r = 5$, $N = 32$ and $n = 128$, α should be at least 12 so that the sink can find out the correct distribution of sensor readings with probability 99%. In this setting, the f-PHA schemes requires each sensor node to send out a message of size $\alpha * (log_2 N + \gamma) = 120$ bits. Note that if b-PHA is employed, the size of reply is $n * log_2 N = 640$ bits. Therefore, f-PHA scheme can significantly reduce the bandwidth consumption.

Discussion From our experiments, we also discovered that f-PHA is not applicable to sensor networks (e.g., $N \geq 128$ and $n \geq 32$). In these cases, the sink may take long time to figure out the distribution of sensor readings, which incur unacceptably high query delay. To address this issue, we propose a hybrid PHA scheme in the following.

3.3.3 h-PHA: Hybrid Perturbed Histogram-based Data Aggregation

In h-PHA, the number of readings each range i is represented as $N_i = N' * x_i + N'_i$, where N'‘N is a system parameter, and all N'_i are sent by using b-PHA and all x_i are sent by using f-PHA. Detailed description is as follows:

System Preparation before Node Deployment This is the same as in f-PHA.

Query Launch at Sink The sink sends out query message

$$\langle Query, \sigma, N', X \rangle,$$

where X is a nonce uniquely associated with this query to thwart replaying attacks, and N' ($N' < N$ and N' is a factor of N) is a system parameter.

Data Replying and Aggregation at Sensor Nodes Upon receiving the query message, each sensor node u responds as follows.

Case I (if node u is a leaf node). Node u replies

$$\langle Reply, \overline{N}_{u,0}, \cdots, \overline{N}_{u,\alpha-1}, \overline{N'}_{u,0}, \cdots, \overline{N'}_{u,n-1} \rangle.$$

Here, the computation of $\overline{N}_{u,0}, \cdots, \overline{N}_{u,\alpha-1}$ is the same as in the f-PHA scheme. The computation of $\overline{N'}_{u,0}, \cdots, \overline{N'}_{u,n-1}$ is similar to but different from that in the b-PHA scheme. Specifically, each $\overline{N'}_{u,i}$ ($i = 0, \cdots, n-1$) is computed as follows:

- If the reading at node u is in range $(i * \sigma, (i+1) * \sigma]$ (or $[0, \sigma]$ for $i = 0$), then $\overline{N'}_{u,i} = [1 + h(u|X|i)]$ MOD N'.
- Otherwise, $\overline{N'}_{u,i} = h(u|X|i)$ MOD N'.

Note that, each $\overline{N'}_{u,i}$ is less than N' (not N as in the b-PHA). Hence, the size of $\overline{N'}_{u,i}$ is $log_2 N'$. If system parameter N' is set to be much smaller than N, the size for this part of message can be reduced significantly.

Case II (if node u is not a leaf node). Node u first waits until it has received replies from all its children, denoted as $\langle Reply, \overline{N}_{j,0}, \cdots, \overline{N}_{j,\alpha-1}, \overline{N'}_{j,0}, \cdots, \overline{N'}_{j,n-1} \rangle$, where $j = 0, \cdots, m_u - 1$ and m_u is the number of children of node u. Then, node u replies message $\langle Reply, \overline{N}_{u,0}, \cdots, \overline{N}_{u,\alpha-1}, \overline{N'}_{u,0}, \cdots, \overline{N'}_{u,n-1} \rangle$. Here, the computation of $\overline{N}_{u,0}, \cdots, \overline{N}_{u,\alpha-1}$ is the same as in f-PHA. The computation of $\overline{N'}_{u,0}, \cdots, \overline{N'}_{u,n-1}$ is as follows:

- If the reading at node u is in range $(i * \sigma, (i+1) * \sigma]$ (or $[0, \sigma]$ for $i = 0$), then $\overline{N'}_{u,i} = [1 + h(u|X|i) + \sum_{j=0}^{m-1} \overline{N'}_{j,i}]$ MOD N'.
- Otherwise, $\overline{N'}_{u,i} = [h(u|X|i) + \sum_{j=0}^{m-1} \overline{N'}_{j,i}]$ MOD N'.

Post-Query Processing at Sink The sink waits until it has received replies from all of its m_0 children. Let these messages be $\langle Reply, \overline{N}_{j,0}, \cdots, \overline{N}_{j,\alpha-1}, \overline{N'}_{j,0}, \cdots, \overline{N'}_{j,n-1} \rangle$, where $j = 0, \cdots, m_0 - 1$.

The sink goes through the following steps to figure out N_i ($i = 0, \cdots, n-1$), i.e., the distribution of sensor readings.

- First, the sink computes $N'_i = \{\sum_{j=0}^{m_0-1} \overline{N'}_{j,i} - \sum_{j=1}^{N} h(j|X|i)\}$ MOD N'. Then, it holds that $N'_i = N_i$ MOD N'. In other words,

$$N_i = N'_i + x_i * N', \tag{3.4}$$

  where x_i is an unknown non-negative integer. Note that, if $\sum_{i=0}^{n-1} N'_i = N$, it must hold that $N_i = N'_i$. Otherwise, the following step should be executed to find out x_i and hence N_i.

- Second, the same as in the f-PHA scheme, the sink obtains the following equations about N_i ($i = 0, \cdots, n-1$):

$$\sum_{i=0}^{n-1} a_{j,i} N_i = c_j,\ j = 0, \cdots, \alpha - 1, \tag{3.5}$$

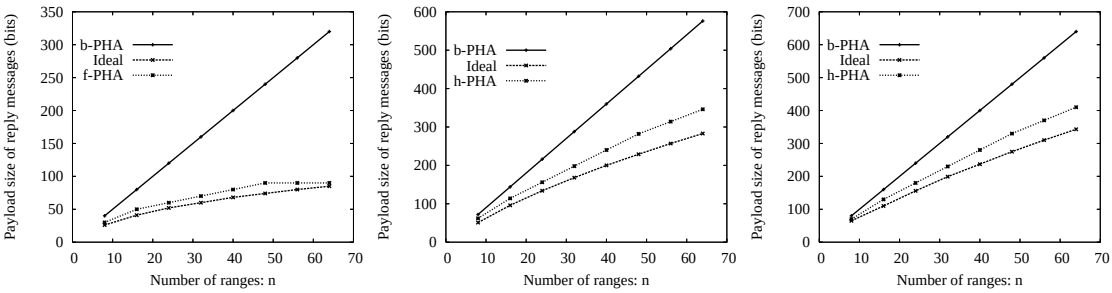
  where

$$c_j = \{\sum_{k=0}^{m_0-1} \overline{N}_{k,j} - \sum_{u=1}^{N} h(u|X|i)\} \text{ MOD } (N * 2^{\gamma}).$$

- Third, combining equations (3.4) and (3.5), the sink can get the following linear equations about x_i ($i = 0, \cdots, n-1$):

$$N' * \sum_{i=0}^{n-1} a_{j,i} x_i = c_j - \sum_{i=0}^{n-1} a_{j,i} N'_i,\ j = 0, \cdots, \alpha - 1.$$

  By solving x_i from these equations, the sink can also solve N_i.



(a) N=32 (b) N = 512, N'=16 and γ=5 (c) N = 1024, N'=32 and γ=5

Figure 3.1 Comparing b-PHA, f-PHA, h-PHA and the ideal design

3.4 Performance Evaluation

Simulations are conducted to compare the performance between the proposed schemes and the ideal design. In the simulations, the number of sensor nodes (which is also the number of sensor readings) varies between 32 and 1024. The number of ranges (i.e., n) varies from 8 to 64. We focus on evaluating the bandwidth consumption of our proposed schemes. Particularly, we use the *payload size of reply messages* that sent by each sensor node as the metric for evaluating the bandwidth consumption. The results are depicted in Figure 3.1, which are also explained in the following.

Figure 3.1 (a) compares the performance of b-PHA, f-PHA and ideal design in the context of a small-scale sensor network (the number of sensor nodes N is 32). As we can see, the ideal design outperforms f-PHA, which in turn outperforms b-PHA. The bandwidth consumed by these schemes all increase as the number of ranges (n) increases. The increase for b-PHA is the most fast, and its required payload size is linear to n, while the increases of other two schemes are much slower. Therefore, the performance advantage of f-PHA over b-PHA becomes more and more significant as n goes up. We can also perceive that the performance difference between f-PHA and the ideal design is small. However, our experiments (not shown here) also have indicated that, f-PHA is not applicable to large-scale networks, in which the computation complexity experienced by the sink becomes very high and may rend the query delay to be unacceptable.

Figure 3.1 (b) and Figure 3.1 (c) compare the performance between b-PHA , h-PHA and the idea design. Both figures show that the ideal design outperforms h-PHA, which outperforms b-PHA. Also, the performance difference between the ideal design and h-PHA is much smaller than that between h-PHA and b-PHA. In addition, the differences increase as the number of ranges n increases. Comparing these two figures with Figure 3.1 (a), we can find out that, the difference between h-PHA and the idea design is greater than that between f-PHA and the idea design. This is because, by replacing f-PHA with h-PHA, we have made tradeoffs between computational complexity at the sink and the communication overhead at sensors. That is, we reduce the computational complexity at the cost of increasing the bandwidth consumption. However, the performance of h-PHA is still much better than b-PHA.

3.5 Conclusion

In this chapter, we proposed a set of novel privacy-preserving data aggregation schemes to support a variety queries in sensor networks. Specifically, these include schemes to support data aggregation in queries targeted at special sensor data or the distribution of sensor data and queries targeted at particular sensor nodes. Simulations have been conducted to evaluate our schemes and the results show that our proposed advanced schemes can achieve nearly-ideal performance in addition to providing perfect privacy protection.

CHAPTER 4. Privacy-Preserving Detection of Integrity Attacks for Data Aggregation in Wireless Sensor Networks

We propose a set of generic privacy-preservation solutions for data aggregation in previous chapter. However, the property of concealing data privacy may be abused by the adversaries to falsify the aggregation result at will. Nobody in the network is able to detect the data integrity attacks after deploying the privacy preserving schemes. Privacy and integrity appear to be conflicting requirements for data aggregation. In this chapter, we propose a simple yet effective scheme, which can protect data integrity and preserve data privacy simultaneously.

4.1 Introduction

Wireless sensors are more and more widely deployed to monitor the working or life conditions of people [3, 4]. As the data collected and processed by these sensors may reveal peoples privacy, such as daily life patterns or health conditions, it is desired to conceal the actual content of the data for privacy preservation. The data concealment feature, however, may be abused by compromised sensors to modify or ill-process data without being caught. Hence, reconciling privacy preservation and intrusion detection, which apparently conflict with each other, is important.

This chapter studies the above problem in the context of sensory data aggregation [16, 32, 17, 18, 20], which has been one of the fundamental primitives necessary for efficient operation of sensor networks. If some sensors are compromised, they may ill-perform data aggregation to distort the aggregation result. Chan *et al* [32] and Yang *et al* [33] have proposed intrusion detection schemes to identify ill-performed aggregations. But these schemes do not attempt to conceal the content of sensory data and thus do not preserve the privacy of people whom the sensory data is associated with. On the other hand, Castelluccia *et al* [23], He *et al* [25] and others [24, 26, 27, 28] have proposed sensory data con-

cealment schemes to preserve the privacy of people who or whose environments have been monitored by sensors. But these schemes cannot detect ill-performed aggregations. To the best of our knowledge, no effective scheme has been designed to simultaneously accomplish data concealment and detection of ill-performed aggregations.

The major barrier in simultaneous accomplishment of data concealment and detection of ill-performed aggregations is that, existing detection schemes [33, 32] need to know the actual content of data that is aggregated and the aggregated result to find out if the aggregation is performed correctly, which disables the employment of existing data concealment schemes [23, 24, 26, 27, 28], and vice versa. To eliminate the barrier, we propose a scheme that can detect ill-performed aggregations without knowing the actual content of data that is aggregated or the actual aggregation result, and therefore allow data to be kept concealed. The proposed scheme is designed on top of a generic framework for data aggregation [27, 28]. Assuming a tree structure is used for sensory data collection, mathematical constructs are delicately devised to enable every non-leaf sensor in the tree to test if its descendant nodes have ill-performed aggregation based on the reports received from these descendants. The reports transmitted in the tree can be concealed (i.e., the actual content of the report is perturbed) and thus are unknown to the detecting sensors.

Analysis, simulation and prototype system implementation have been conducted. The results show that, the actual content of raw and aggregated sensory data that are transmitted in the network can be well concealed and thus the goal of privacy preservation is accomplished. Meanwhile, most of ill-performed aggregations can be detected; the ill-performed aggregations that have escaped from detection have only negligible impact on the final aggregation results. Besides, the proposed scheme has low computational and communication overheads, and the storage overhead is affordable by current generation of wireless sensor motes.

4.2 System Model

4.2.1 Network Assumptions

Aligned with existing work on sensory data aggregation [33, 32, 25, 27], the following assumptions are made: A sensor network consists of a sink and N static sensors, where each sensor has a unique

ID picked from $\{1, \cdots, N\}$ and the ID of the sink is 0. Each sensor monitors its direct environment, generates sensory data and responses to queries from the sink. The sink is aware of the deployed network topology, and has powerful computation, storage and communication capabilities compared to sensors. Each sensory data item is an integer ranging from 0 to some upper bound denoted as U_d. Note that, even though some data (e.g., temperature, humidity, noisiness, etc.) may not be integers in its original form, they can be converted to integers. As in a typical data aggregation protocol, a tree rooted at the sink is formed by connecting sensors and the sink [32, 33]. The sink broadcasts each query via the tree downwards, and sensors forward their replies back to the sink via the tree upwards. During the course of forwarding replies, each non-leaf sensor aggregates its own data with the data from its children, and only forwards the aggregation result to its parent. We assume each sensor u knows how many descendants it has, which is denoted as N_u.

We assume a relatively static network topology as privacy critical applications are often deployed inside a building where people's activities are monitored. Note that, the topology of the data aggregation tree may change due to node or link failures. We assume these failures can be detected and the sink is aware of them. We provides detailed discussion for this problem at the end of Sec 4.4.3. And the failed nodes can further be replaced manually due to the indoor deployment of the network. Furthermore, a reliable transmission mechanism is assumed to be adopted as in [33, 32, 25, 27].

4.2.2 Security Assumptions and Attack Model

We assume the sink is trustworthy while any sensor could be compromised. As our study focuses on data concealment and detection of ill-performed data, only the following attacks are mainly considered: (i) outsiders or compromised sensors eavesdrop packets in transmission; and (ii) compromised sensors modify aggregation results, and attempt to make the sink accept the false aggregation results (i.e., ill-performed aggregation).

We do not consider the attack in which a compromised node forges its own sensory data. As pointed out by [33, 32], the final aggregation results will not drift far away from the true results if compromised sensors can only forge their own data.

4.3 Preliminaries

For different statistics queried by the sink, there are different data aggregation algorithms. To make our solution generic, we build our solution on top of a generic data aggregation framework, i.e., the histogram-based framework [27]. In this section, the histogram-based framework is first presented, which is followed by the scheme for data concealment on top of the framework and attacks on the scheme.

4.3.1 Histogram-based Framework for Data Aggregation

The valid range of sensory data, i.e., $[0, U_d]$, is uniformly divided into a certain number (denoted as n) of buckets. Given a query requested by the sink, assume the actual sensory data that should be reported by a sensor u is denoted as d_u, and d_u falls into the i^{th} bucket (i.e., $\lfloor \frac{d_u}{U_d/n} \rfloor = i$).

- If sensor u is a leaf node, it sends to its parent n-tuple histogram $D_u = \langle D_{u,0}, \cdots, D_{u,n-1} \rangle$, where $D_{u,i} = 1$, and for every $j \neq i$, $D_{u,j} = 0$.

- If sensor u is not a leaf node, it first collects reports from all of its children, denoted as D_{v_k} ($k = 0, \cdots, m_u - 1$). Then, it sends to its parent n-tuple histogram $D_u = \langle D_{u,0}, \cdots, D_{u,n-1} \rangle$, where $D_{u,i} = \sum_{k=0}^{m_u-1} D_{v_k,i} + 1$, and for every $j \neq i$, $D_{u,j} = \sum_{k=0}^{m_u-1} D_{v_k,j}$.

After collecting and aggregating histograms reported by its children, the sink can approximately compute the *sum*, *average*, *max/min*, *median*, *standard deviation*, etc. of all the sensory data generated in the network. For example, assuming the resulting histogram is $\langle D_{0,0}, \cdots, D_{0,n-1} \rangle$, the *median* of all the sensory data can be approximated as

$$(k + \frac{1}{2})\frac{U_d}{n}, \text{where} \sum_{j=0}^{k-1} D_{0,j} < \frac{N}{2} \text{and} \sum_{j=0}^{k} D_{0,j} \geq \frac{N}{2}.$$

Note that, the actual median value must be in the range of $[k\frac{U_d}{n}, (k+1)\frac{U_d}{n})$. Therefore, the difference between $(k + \frac{1}{2})\frac{U_d}{n}$, i.e., the obtained median value, and the actual median value is not greater than $\frac{U_d}{2n}$.

Note: To ease presentation, hereafter we treat a n-tuple histogram $D_u = \langle D_{u,0}, \cdots, D_{u,n-1} \rangle$ as an integer $D_u = \sum_{j=0}^{n-1} D_{u,j} \cdot 2^{bj}$, where $b = \lceil \log_2 N \rceil$ and N is the total number of nodes in the network.

4.3.2 Data Concealment in Histogram-based Aggregation

A scheme to conceal the actual content of data on top of the histogram-based aggregation framework has been proposed [27], which is briefly described as follows.

To conceal the actual content of data, instead of reporting original histogram D_u as described in Section 4.3.1, sensor u reports $\tilde{D}_u$, where $\tilde{D}_u = D_u + P_u \bmod q$, and $q = 2^{bn}$. Here, P_u is a node-specific secret number computed as $P_u = \sum_{j=0}^{n-1} p_{u,j} \cdot 2^{bj}$, where each $p_{u,j} = h(K_u|X|j) \mod 2^b$, K_u is a secret key shared between node u and the sink, X is the nonce uniquely associated with current query, and $h(.)$ is the secure one-way hash function.

After the sink has aggregated histograms from all of its children and obtains $\tilde{D}_0$, the original histogram for all sensory data in the network (still denoted as D_0) can be computed as $D_0 = \tilde{D}_0 - \sum_{j=1}^{N} P_j \bmod q$.

4.3.3 Ill-performed Data Aggregation

With the data concealment scheme, histograms transmitted in network are perturbed and thus become unintelligible to everyone but the sink. A compromised node may take advantage of this fact by not conducting aggregation honestly and such abuse of data concealment cannot be caught. This problem is to be addressed in our proposed scheme.

4.4 Proposed Scheme

In this section, we first present high-level ideas of our design, which are followed by the roadmap which shows how our final design is derived step by step and finally our proposed scheme is elaborated.

4.4.1 High-level Ideas of Our Design

On top of the generic histogram-based data aggregation framework, we aim to propose a scheme that enables non-leaf sensors to check if their descendants have ill-performed aggregation though the detecting sensors do not know the actual histograms which should be concealed. The scheme is designed based on the following ideas.

4.4.1.1 Perturbation for Data Concealment

As in [27], each sensor shares a unique secret key with the sink. A unique nonce is disseminated along with a query from the sink. Responding to the query, each sensor reports a histogram which is added by a perturbation number determined together by its unique secret key and the unique nonce of the query. The perturbation is unknown by anyone other than the reporting sensor and the sink, and it is not reused. The perturbation mechanism conceals the actual histograms and thus conceals the actual sensory data.

4.4.1.2 Leveraging Unique Properties of Histogram Data Structure for Detection of Ill-performed Aggregations

Responding to each query, each sensor generates one and only one data item which falls into one and only one bucket of the histogram. Therefore, for any non-leaf sensor u, supposing the number of its descendants is N_u and each of its children sends up an aggregated histogram, it can obtain an aggregated histogram which is the sum of all the received histograms, and the aggregated histogram should satisfy the following necessary conditions if its descendants have not ill-performed aggregations:

- *Necessary Condition 1*. The sum of the numbers in all buckets of the aggregated histogram should be N_u.
- *Necessary Condition 2*. The number in each bucket of the aggregated histogram should be no less than 0 and no greater than N_u.

These two necessary conditions are leveraged to check aggregation integrity. As these two are necessary conditions, it does not guarantee that there is no ill-performed aggregation if the conditions are both satisfied. However, as shown by the evaluation results in Section 4.6, using these two conditions can detect most of ill-performed aggregations, and the impact of undetected ones is highly limited.

4.4.2 Roadmap: How to Reconcile Data Concealment and Detection of Ill-performed Aggregations?

The above two ideas, however, cannot be directly applied simultaneously because of the following contradiction: if the histogram is perturbed (i.e., applying the first idea), the exact numbers in each

bucket become unknown which makes it hard to check the two necessary conditions (i.e., applying the second idea). Our proposed scheme aims to reconcile the contradiction. In the following, we present the roadmap that we have followed, which leads to our final approach that can eventually reconcile the contradiction.

4.4.2.1 Naive Approach

To conceal the actual data, each sensor v reports a perturbed histogram $\tilde{D}_v$, which is obtained by adding a node specific perturbation denoted as P_v to the original histogram D_v. The perturbed histogram $\tilde{D}_v$ is aggregated hop by hop as it is forwarded to the sink. Therefore, at any intermediate sensor u, the sum of perturbed histograms received from all of its children, denoted as R_u, should be:

$$\begin{aligned} R_u &= \sum_{v \in \Gamma_u} (D_v + P_v) \mod q \\ &= \sum_{v \in \Gamma_u} D_v + \sum_{v \in \Gamma_u} P_v \mod q, \end{aligned}$$

where Γ_u is the set of all descendants of u. Hence, $\sum_{v \in \Gamma_u} D_v \mod q$ is the actual intermediate aggregation result. Hereafter, let

$$S_u = \sum_{v \in \Gamma_u} P_v \mod q,$$

and

$$H_u = \sum_{v \in \Gamma_u} D_v \mod q.$$

As a naive approach, sensor u can be preloaded with S_u. Knowing S_u, sensor u can compute H_u (i.e., the actual histogram of the sensory data generated by all of its descendants), and thus can test the two necessary conditions. However, the intermediate aggregation result is also exposed, which is not desired.

4.4.2.2 Enhanced Approach

To eliminate the privacy-leakage problem in the naive approach, for each non-leaf sensor u, we can preload a *perturbed* sum of the perturbations used by its descendants. Specifically, for each sensor u, instead of preloading S_u directly, we preload $\tilde{S}_u = (S_u - W_u) \mod q$, where W_u is a number randomly picked for sensor u but is unknown by u. Suppose the concatenation format of W_u is $W_u =$

$\langle w_{u,n-1}, \cdots, w_{u,0} \rangle$, where each $w_{u,j} \in \{0, \cdots, 2^b - 1\}$ ($j = 0, \cdots, n-1$), the sink constructs W_u such that each $w_{u,j}$ satisfies $0 \leq w_{u,j} < 2^b - N_u$. In addition, $\bar{W}_u = \sum_{j=0}^{n-1} w_{u,j}$ is also preloaded to sensor u. Based on preloaded $\tilde{S}_u$ and $\bar{W}_u$, as well as the sum of received perturbed histograms (i.e., R_u), sensor u can obtain the perturbed histogram $\tilde{H}_u$ as

$$\begin{aligned} \tilde{H}_u &= (R_u - \tilde{S}_u) \mod q \\ &= [R_u - (S_u - W_u)] \mod q \\ &= [(R_u - S_u) + W_u] \mod q \\ &= (H_u + W_u) \mod q. \end{aligned}$$

Let the concatenation format of H_u be $H_u = \langle h_{u,n-1}, \cdots, h_{u,0} \rangle$, where each $h_{u,j} \in \{0, \cdots, 2^b - 1\}$ ($j = 0, \cdots, n-1$). If there is no ill-performed aggregation, it holds that $\sum_{j=0}^{n-1} h_{u,j} = N_u$ (i.e., the first necessary condition) and $0 \leq h_{u,j} \leq N_u$ for $j = 0, \cdots, n-1$ (i.e., the second necessary condition). Furthermore, because $0 \leq w_{u,j} < 2^b - N_u$, it holds that $0 \leq h_{u,j} + w_{u,j} < 2^b$. That is, there will be no overflow caused by adding perturbation at each bucket. Consequently, the following equation should hold:

$$\sum_{j=0}^{n-1} \tilde{h}_{u,j} = \sum_{j=0}^{n-1} h_{u,j} + \sum_{j=0}^{n-1} w_{u,j} = N_u + \bar{W}_u.$$

The above equation can be used to test the necessary conditions of no ill-performed aggregation. However, this approach has the following limitation: for each bucket j, $2^b - 1 \geq \tilde{h}_{u,j} = h_{u,j} + w_{u,j} \geq h_{u,j} \geq 0$. That is, sensor u knows that the actual number of sensory data falling in bucket j (i.e., $h_{u,j}$) should be no greater than $\tilde{h}_{u,j}$, which exposes the range of $h_{u,j}$.

4.4.2.3 Key Ideas of Our Final Approach

In order not to expose the range of the number in each bucket, we propose our final approach. Our final approach inherits the ideas of the enhanced approach. Moreover, in the final approach, the information preloaded to each sensor is delicately constructed such that *the number in each bucket j (i.e., $h_{u,j}$) can be less than, equal to, or greater than $\tilde{h}_{u,j}$; while in the afore-described enhanced approach, $h_{u,j}$ is always no greater than $\tilde{h}_{u,j}$*. This way, less information about the histogram is exposed. The detailed description of final approach is elaborated in the following.

4.4.3 Detailed Description of Our Proposed Scheme

The detailed description of our proposed scheme is broken down to four parts, namely, system initialization, query launched at the sink, response at sensors, and processing at the sink.

4.4.3.1 System Initialization

For a sensor network with N sensors, we use $b = \lceil \log_2 N \rceil + 1$ bits to record the number of sensory readings in each bucket. Recall that, previous approaches use $\lceil \log_2 N \rceil$ bits. We introduce one more bit to provide larger range of perturbations for better data concealment. Based on n and b, a finite field $\mathbb{F}_q$ ($q = 2^{n*b} = 2^{n*(\lceil \log_2 N \rceil + 1)}$) is constructed.

The sink prepares t nonces, denoted as $X_1, X_2, \cdots, X_t$, where X_i is used in the i-th query, and no nonce is reused. For each sensor u and for each nonce X_i, the sink computes sensor u's perturbation number P_u^i as a large number in $\mathbb{F}_q$ as follows:

$$P_u^i = \sum_{j=0}^{n-1} p_{u,j} \cdot 2^{bj}, \tag{4.1}$$

where $p_{u,j} = h(K_u|X_i|j) \mod 2^b$, K_u is a secret key exclusively shared between sensor u and the sink, and $h(.)$ is a one way secure hash function.

Based on the known aggregation tree, the sink also computes for each non-leaf sensor u the sum of perturbation numbers used by all of its descendants, which is denoted as S_u^i. Specifically, $S_u^i = (\sum_{e \in \Gamma_u} P_e^i) \mod q$, where Γ_u is the set of all descendants of sensor u. Similar to the enhanced approach, the sum S_u^i is not directly loaded to sensor u; instead a perturbed version of the sum denoted as $\tilde{S}_u^i$ is computed and loaded to sensor u. Specifically, $\tilde{S}_u^i = (S_u^i - W_u^i) \mod q$, where $W_u^i = \sum_{j=0}^{n-1} w_{u,j}^i \cdot 2^{bj}$ and $0 \leq w_{u,j}^i < 2^b$ for each j. The sink also loads sensor u with $\bar{W}_u^i = \sum_{j=0}^{n-1} w_{u,j}^i$.

To avoid exposing the range of $h_{u,j}$ as in the enhanced approach, W_u^i is constructed as follows:

- We first randomly select buckets $\tau_1, \tau_2, \cdots, \tau_m$ from all buckets, and for $\forall k, 1 \leq k \leq m$, we let $w_{u,\tau_k}^i = 0$.
- From the remaining $n - m$ buckets, we further randomly select m' buckets: $\tau_1', \tau_2', \cdots, \tau_{m'}'$, and randomly generate $w_{u,\tau_k'}^i$ such that

$$2^b - N_u < w_{u,\tau_k'}^i < 2^b.$$

- For each bucket j belonging to the rest $n-m-m'$ buckets, the value of $w_{u,j}^i$ is arbitrarily chosen such that

$$0 < w_{u,j}^i < 2^b - N_u.$$

To summarize, for each $i \in \{1, \cdots, t\}$, each sensor u is loaded with

- $\tilde{S}_u^i$: the perturbed sum of the perturbation numbers that will be used by all the descendants of sensor u during the i-th query (note: if sensor u is leaf node, $\tilde{S}_u^i = 0$)

- $\bar{W}_u^i$.

4.4.3.2 Query Launched at the Sink

For the i-th query, the sink sends out a query message containing nonce X_i.

4.4.3.3 Responses by Sensor u to the i-th Query

Upon receiving the query message containing nonce X_i, each sensor u prepares its sensory reading D_u, computes the perturbation P_u used for current round, computes its own perturbed data $\tilde{D}_u = (D_u + P_u) \mod q$, and replies the query differently depending on whether it is a leaf node or not:

- If sensor u is a leaf node, it only reports its individual perturbed data $A_u = \tilde{D}_u$ to its parent. Note that, its actual histogram report D_u has been concealed by the secret perturbation P_u, which is only known by sensor u itself and the sink.

- If sensor u is a non-leaf node, it waits until it has received reply A_{v_j} from every child v_j, where $j = 0, \cdots, m_u - 1$ and m_u is the number of children of sensor u. Then, sensor u checks if its descendants have ill-performed aggregations. The checking procedure is shown by Algorithm 1. If no ill-performed aggregation is detected, sensor u aggregates its own perturbed data $\tilde{D}_u$ with the sum of received perturbed data (i.e., R_u), and sends the sum of these two parts, i.e., $A_u = (R_u + \tilde{D}_u) \mod q$, to its parent.

The correctness of Algorithm 1 is based on the following Theorem 4.4.1.

Algorithm 1 Detection of Ill-performed Aggregation (Run by Non-leaf Sensor u):

1: Aggregate the received data $R_u = \sum_{j=0}^{m_u-1} A_{v_j} \mod q$;
2: $\tilde{H}_u = (R_u - \tilde{S}_u^i) \mod q$
3: Find $\tilde{h}_{u,j} \in \{0, \cdots, 2^b - 1\}$ for $j = 0, \cdots, n-1$ such that $\tilde{H}_u = \sum_{j=0}^{n-1} \tilde{h}_{u,j} \cdot 2^{bj}$
4: $Y = \bar{W}_u + N_u - \sum_{j=0}^{n-1} \tilde{h}_{u,j}$
5: **if** there exists $y \in \{0, \cdots, m'\}$ such that $Y = y * (2^b - 1)$ or $Y = y * (2^b - 1) + 1$ **then**
6: No ill-performed aggregation is detected
7: **else**
8: Ill-performed aggregation is detected; sink is notified

Theorem 4.4.1 *Suppose sensor u is a non-leaf node. It is preloaded with $\tilde{S}_u^i$ and $\bar{W}_u^i$ ($i = 1, \cdots, t$). For the i-th query issued by the sink, sensor u has received perturbed histogram A_{v_j} from each of its children v_j ($j = 0, \cdots, m_u - 1$).*

Let the number of descendants of sensor u be N_u, $\tilde{H}_u = (R_u - \tilde{S}_u^i) \mod q = \sum_{j=0}^{n-1} \tilde{h}_{u,j} \cdot 2^{bj}$, where $\tilde{h}_{u,j} \in \{0, \cdots, 2^b - 1\}$, and $Y = \bar{W}_u + N_u - \sum_{j=0}^{n-1} \tilde{h}_{u,j}$.

If no any descendant of sensor u has ill-performed aggregations, then the following condition should be satisfied:

$$Y = y * (2^b - 1), \text{or } Y = y * (2^b - 1) + 1, \tag{4.2}$$

where $y \in \{0, \cdots, m'\}$.

Proof According to the definition of $\tilde{S}_u^i$, we have

$$\begin{aligned} \tilde{H}_u &= (R_u - \tilde{S}_u^i) \mod q \\ &= [R_u - (S_u^i - W_u^i)] \mod q \\ &= [(R_u - S_u^i) + W_u^i] \mod q \\ &= (H_u + W_u^i) \mod q, \end{aligned}$$

where, as defined before, H_u is the actual aggregated histogram.

Let $H_u = \sum_{j=0}^{n-1} h_{u,j} \cdot 2^{bj}$, where each $h_{u,j} \in \{0, \cdots, 2^b - 1\}$; and $W_u^i = \sum_{j=0}^{n-1} w_{u,j}^i \cdot 2^{bj}$, where each $w_{u,j}^i \in \{0, \cdots, 2^b - 1\}$. Assuming that descendants of sensor u have not ill-performed aggregations, we have

$$\sum_{j=0}^{n-1} h_{u,j} = N_u,$$

according to the aforementioned two necessary conditions.

Also, we have

$$\sum_{j=0}^{n-1} \tilde{h}_{u,j} \cdot 2^{bj} = \sum_{j=0}^{n-1} [h_{u,j} + w_{u,j}^{i}] \cdot 2^{bj}.$$

If each $(h_{u,j} + w_{u,j}^{i})$ is less than 2^b, it holds that $\sum_{j=0}^{n-1} \tilde{h}_{u,j} = \sum_{j=0}^{n-1} [h_{u,j} + w_{u,j}^{i}] = N_u + \bar{W}_u^i$. However, among $w_{u,j}^{i}$ ($j = 0, \cdots, n-1$), m' of them are greater than $2^b - N_u$. Hence, there exist $\{j_1, \cdots, j_y\} \subset \{0, \cdots, n-1\}$ and $y \in \{0, \cdots, m'\}$ such that, $h_{u,k} + w_{u,k}^{i} \geq 2^b$ ($k \in \{j_1, \cdots, j_y\}$). Intuitively, this means adding perturbations cause y buckets of the histogram to generate overflow. According to the basic arithmetic, it follows that

$$\sum_{j=0}^{n-1} \tilde{h}_{u,j} = N_u + \bar{W}_u^i - y * (2^b - 1),$$

if $n - 1 \notin \{j_1, \cdots, j_y\}$ (that is, the leftmost bucket does not have overflow); or

$$\sum_{j=0}^{n-1} \tilde{h}_{u,j} = N_u + \bar{W}_u^i - [y * (2^b - 1) + 1],$$

if $n - 1 \in \{j_1, \cdots, j_y\}$ (that is, the leftmost bucket has overflow).

Because $Y = \bar{W}_u + N_u - \sum_{j=0}^{n-1} \tilde{h}_{u,j}$, the above two conditions can be rewritten as

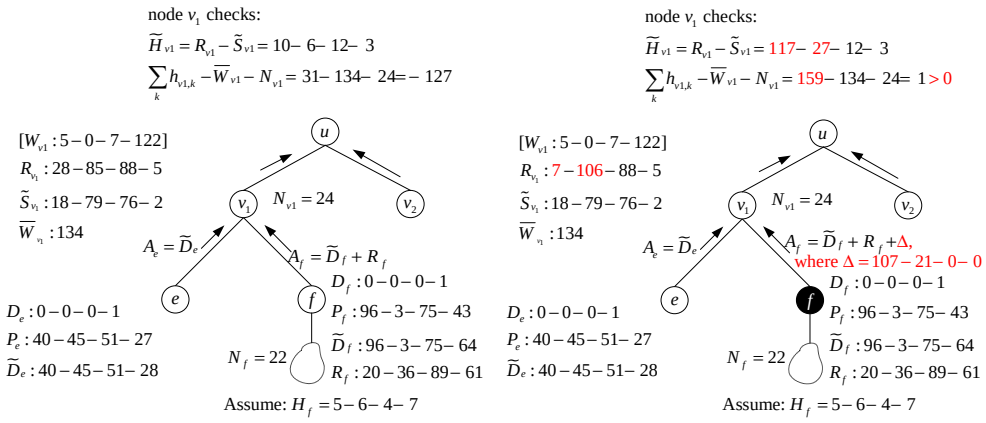
$$Y = y * (2^b - 1), \text{or } Y = y * (2^b - 1) + 1,$$

where $y \in \{0, \cdots, m'\}$.

Fig.4.1 shows an example of applying the proposed algorithm to test if there is ill-performed aggregation. Here, $b = 7$, the number of bucket $n = 4$, and hence $q = 2^{28}$. Fig. 4.1(a) shows an example without ill-performed aggregation. Sensor v_1 is preloaded with $\tilde{S}_{v_1}$ and $\bar{W}_{v_1}$ shown in the figure. For the ease of presentation, all numbers, except $\bar{W}_u$, are represented in the concatenation format. For example, $\tilde{S}_{v_1} : \langle 18, 79, 76, 2 \rangle$ means $\tilde{S}_{v_1} = 18 * 2^{21} + 79 * 2^{14} + 76 * 2^7 + 2$.

Let us consider sensor v_1. S_{v_1} is computed in the system initialization phase; in this example, it is $\langle 23, 79, 83, 124 \rangle$, and W_{v_1} is $\langle 5, 0, 7, 122 \rangle$. Hence $\bar{W}_{v_1} = 5 + 0 + 7 + 122 = 134$. Sensor v_1 is only preloaded with $\tilde{S}_{v_1}$ and $\bar{W}_{v_1}$, S_{v_1} and W_{v_1} are unknown.

When sensor v_1 receives the perturbed histograms from sensor e and sensor f, it first sums up the received histograms to obtain $R_{v_1} = A_e + A_f \mod q$. To check if there is ill-performed aggregation, sensor v_1 computes $\tilde{H}_{v_1} = (R_{v_1} - \tilde{S}_{v_1}) \mod q$, which results in $\langle 10, 6, 12, 3 \rangle$. Then, sensor v_1



(a) Without Ill-performed Aggregation (b) With Ill-performed Aggregation

Figure 4.1 Illustration of Data Aggregation and Detection of Ill-performed Aggregations

computes:

$$Y = \bar{W}_{v_1} + N_{v_1} - \sum_{i=0}^{3} \tilde{h}_{v_1,i} = 134 + 24 - (10 + 6 + 12 + 3) = 127,$$

which satisfies the condition in the algorithm. Hence, no ill-performed aggregation is detected.

Fig. 4.1(b) shows an example with ill-performed aggregation. Here, sensor f is compromised, and it modifies its intermediate aggregation result by Δ, which is detected by sensor v_1 since the condition specified in the algorithm cannot be satisfied.

Specifically, sensor f computes the perturbation number P_f, which is $\langle 96, 3, 75, 43 \rangle$. As the actual histogram generated by it, denoted as D_f, is $\langle 0, 0, 0, 1 \rangle$, the perturbed version of the histogram, denoted as $\tilde{D}_f$, should be $\langle 96, 3, 75, 44 \rangle$. Furthermore, sensor f receives perturbed histogram $R_f = \langle 20, 36, 89, 61 \rangle$ from its children. Hence, if aggregation is performed correctly, it should report $A_f = \tilde{D}_f + R_f$. But, as a malicious sensor, it reports $A_f = \tilde{D}_f + R_f + \Delta$, where $\Delta = \langle 107, 21, 0, 0 \rangle$. However, this attack can be detected by its parent v_1 because the necessary conditions of correct aggregation are not satisfied. Besides, from this example we can see that, sensor v_1 receives the perturbed data and computes the perturbed histogram $\tilde{H}_{v_1}$ based on preloaded $\tilde{S}_{v_1}$ and $\bar{W}_{v_1}$. The data is concealed from sensor v_1, meanwhile, the detection of ill-performed aggregation is enabled.

4.4.3.4 Processing at Sink

Suppose the sink has m_0 children $v_0, \cdots, v_{m_0-1}$. It will receive A_{v_j} from each of them, where $j = 0, \cdots, m_0 - 1$. The sink checks if there is ill-performed aggregation in each subtree rooted at v_j and recovers the tree-wide aggregated histogram using Algorithm 2.

Algorithm 2 Processing at Sink

Let $\Psi = \emptyset$;
for Each child v_j, $0 \leq j < m_0$ **do**
  $S_{\Gamma_{v_j}} = \sum_{e \in \Gamma_{v_j}} P_e^i \mod q$, where Γ_{v_j} includes all sensors in the tree rooted at v_j
  $H_{\Gamma_{v_j}} = (A_{v_j} - S_{\Gamma_{v_j}}) \mod q$
  Find $h_j \in \{0, \cdots, 2^b - 1\}$ for $j = 0, \cdots, n-1$ such that $H_{\Gamma_{v_j}} = \sum_{j=0}^{n-1} h_j \cdot 2^{bj}$.
  if $\sum_{j=0}^{n-1} h_j = N_{v_j} + 1$ and $0 \leq h_j \leq N_{v_j} + 1$ for every $j \in \{0, \cdots, n-1\}$ **then**
    $\Psi = \Psi \cup \{v_j\}$
  else
    Ill-performed aggregation detected in subtree rooted at v_j
Compute final histogram $H_0 = \sum_{v_j \in \Psi} H_{\Gamma_{v_j}}$

Discussion: In the wireless sensor network, network topology may be changed because of failed sensors. To deal with the dynamics of network topology, we seek for neighboring sensors to notify the sink of failed sensors. As shown in Fig. 4.2, supposing sensor D fails, its children A or/and B may detected this failure. Then, they report the failure to the sink by sending the report through another path to the sink. For example, they may send the report via neighboring sensor C. The failure reporting may be encrypted via each sensor's secret key with the sink, and corresponding MACs may also be attached so as to let the sink verify the authenticity of such reports. When the sink receives the failure report, it generates faked reports for the failed sensors and sends them back. Particularly, the sink may generate a faked report for sensor D and sends it to one child of sensor D, say sensor B. When performing data aggregation for the i-th query, sensor A may forward to sensor B. Sensor B aggregates what it received with the faked report for sensor D, and then sends the intermediate aggregation result to neighboring sensor C. Sensor C just forwards it to its parent. The data aggregation and checking procedure can then be resumed at sensor F.

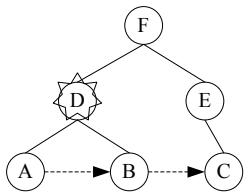


Figure 4.2 Deal with Network Dynamics

4.4.4 Probabilistic Version of the Proposed Scheme

The proposed scheme described in Section 4.4.3 requires every non-leaf sensor u stores $\langle \tilde{S}_u^i, \bar{W}_u^i \rangle$ for every query i, and takes part in detection for every query. This may require too much storage overhead. To reduce storage cost, for each query i, each sensor u is preloaded with $\langle X_i, \tilde{S}_u^i, \bar{W}_u^i \rangle$ with probability p, which is called its *detection participation probability* or *participation probability* for short. Sensor u takes part in detection for i-th query with nonce X_i only when it has been preloaded with detection knowledge $\langle \tilde{S}_u^i, \bar{W}_u^i \rangle$. In the following performance analysis and evaluation, we assume the probabilistic version of the proposed scheme is adopted, and study its performance as the participation probability p varies.

4.5 Performance Analysis

In this section, we mainly analyze the strength of data concealment and the storage overhead for the proposed scheme. The trade-off between various system parameters is studied in Section 4.6, and the computational overhead is evaluated in the Section 5.4.4.

4.5.1 Concealment of Individual Sensory Data

An outsider/insider attacker may intercept perturbed data $\tilde{D}_u$ reported by a sensor u. However, as the perturbation P_u is only known by sensor u and the sink, the probability for the attacker to correctly derive D_u from $\tilde{D}_u$ is as low as $1/q$, recalling $q = 2^{(\lceil \log_2 N \rceil + 1)n}$, n is the number of buckets and N is the number of nodes in the network. Typically, q is selected as equal or greater than 2^{80}, and therefore the probability of successful derivation is negligibly low.

Even the attacker compromises some sensors, there is still no help for the attacker to derive the secret perturbation of an innocent sensor since the secret used by each sensor is independent to each

other. Hence, the individual data D_u is concealed.

4.5.2 Concealment of Intermediate/Final Aggregation Results

4.5.2.1 Concealment of the Actual Histogram

The intermediate aggregation result computed by a node is the distribution of sensory data (i.e., histogram) from all of its descendants. Our proposed scheme can conceal the actual intermediate aggregation results; hence, the final aggregation result is also concealed. Specifically, in our proposed scheme, sensor u is able to compute perturbed histogram $\tilde{H}_u$ so as to check the integrity of received perturbed histogram. But, it is computationally hard to recover the actual histogram H_u from $\tilde{H}_u$, even if the relation $\tilde{H}_u = (H_u + W_u^i) \mod q$ is known. This is because, H_u is protected by the secret number $W_u^i = \langle w_{u,n-1}^i, \cdots, w_{u,0}^i \rangle$, which is unknown to any node but the sink. It is also hard to figure out W_u^i from $\bar{W}_u^i$ because the probability is $\frac{1}{C(\bar{W}_u^i+n-1,n-1)}$, which is very low as long as $\bar{W}_u^i$ is large.

4.5.2.2 Concealment of the Actual Number in a Bucket

As recovering the whole histogram from a perturbed one is hard, the attacker may attempt to find out the number in a certain bucket of the histogram. Next, we study how to evaluate the effectiveness of such attack.

Recall that the number in a bucket of the original histogram, i.e., $h_{u,k}$, may be less than, equal to or greater than $\tilde{h}_{u,k}$, which is exposed. If $h_{u,k} > \tilde{h}_{u,k}$, the possible value of $h_{u,k}$ should be in the set $C_1 = \{\tilde{h}_{u,k} + 1, \cdots, N_u\}$; if $h_{u,k} = \tilde{h}_{u,k}$, the possible value of $h_{u,k}$ should be in the set $C_2 = \{\tilde{h}_{u,k}\}$; if $h_{u,k} < \tilde{h}_{u,k}$, the possible value of $h_{u,k}$ should be in the set $C_3 = \{0, 1, \cdots, \min\{\tilde{h}_{u,k} - 1, N_u\}\}$.

The adversary can use $\tilde{h}_{u,k}$ to guess $h_{u,k}$. Specifically, for each bucket k, $0 \leq k < n$, the attacker knows $\tilde{h}_{u,k}$. Also, the attacker knows the system parameters m and m', and thus can guess the probabilities for $\tilde{h}_{u,k} > h_{u,k}$, $\tilde{h}_{u,k} = h_{u,k}$ and $\tilde{h}_{u,k} < h_{u,k}$. To simplify estimation of the difficulty for the attacker to guess $h_{u,k}$, the probabilities are approximated by $(n - m' - m)/n$, m/n and m'/n, respectively, in this paper. Based on this knowledge, it can naturally guess $h_{u,k}$ as its expected value $E(h_{u,k})$. We then use the average distance between $h_{u,k}$ and $E(h_{u,k})$ as our metric to measure the strength of concealing histogram $h_{u,k}$. For sensor u, the strength of data concealment is defined as

$\frac{\sum_{k=0}^{n-1}|h_{u,k}-E(h_{u,k})|}{n}$. We also utilize entropy to measure the uncertainties of $h_{u,k}$. Let $E_{h_{u,k}}$ be the entropy of $h_{u,k}$, then, $E_{h_{u,k}} = -\sum_{i\in C} p_i \log_2 p_i$, where $C = C_1 \cup C_2 \cup C_3$. We thereafter define the entropy of histogram E_H as $E_H = \frac{\sum_{k=0}^{n-1} E_{h_{u,k}}}{n}$ to measure the uncertainly of truth histogram H. Note that, the actual probabilities for the cases of $\tilde{h}_{u,k} > h_{u,k}$ and $\tilde{h}_{u,k} < h_{u,k}$ are hard to find; hence, the strength of data concealment estimated here serves as an estimation. In Section 4.6, the above-defined strength of data concealment is evaluated as system parameters vary.

4.5.3 Storage Overhead

If a non-leaf node u is preloaded with $\langle X_i, \tilde{S}_u^i, \bar{W}_u^i\rangle$, it can verify whether its descendant nodes have ill-performed aggregation for the i-th query. $\tilde{S}_u^i$ is a large number with nb bits. And,

$$\begin{aligned}\bar{W}_u^i &\leq m'(2^b-1)+(n-m-m')(2^b-N_u-1)\\ &\leq m'(2^b-1)+(n-m-m')(2^b-1)\\ &\leq (n-m)(2^b-1)\leq 2^b n\end{aligned}$$

From the above analysis, $\bar{W}_u^i$ should not take more than $b+\log_2 n$ bits. Suppose X_i is 10 bytes, then there are 2^{80} available nonces. Hence, each tuple $\langle X_i, \bar{W}_u^i, \tilde{S}_u^i\rangle$ will consume $\frac{(n+1)b+\log_2 n}{8}+10$ bytes. Considering the limitation in storage capacity, each sensor is only preloaded with $\langle X_i, \bar{W}_u^i, \tilde{S}_u^i\rangle$ with participation probability p. If we use $100KB$ external flash to store the preloaded information, noting that the external flash size is $1MB$ for TelosB mote, the participation probability for each sensor is 0.05. Table 4.1 shows that the number of nonces that can be used per day if the typical lifetime of a node is 3 years [45].

Table 4.1 Number of Nonces can be Used per Day in Mote's Life Time

	N=128	N=256	N=512	N=1024
n=10	87	84	80	73
n=14	76	71	65	62
n=18	65	62	58	52

From Table 4.1, we can see that there are 87 nonces available each day within 3 years when the number of nodes N is 128 and the number of buckets n is 10. Though the number of nonces can be used per day becomes less when increasing N and n, it is still practicable under typical N and n

settings. Therefore, the proposed scheme is feasible in terms of storage overhead within mote's typical life time.

4.6 Performance Evaluation with Simulation

4.6.1 Experiment Setup

The proposed scheme is simulated to evaluate the effectiveness and efficiency of detecting ill-performed aggregation. We simulate a sensor network with 400 sensors distributed over a $1000 \times 1000m^2$ field. We assume that valid sensory data is in range $\{0, \cdots, 99\}$. By default, the number of buckets is 10, and the system parameters m and m' are 1 and 2, respectively. But these parameters vary in the simulations studying their impacts. In the figures of simulation results, each plotted value is an average over 100 simulation runs with the same set of system parameters.

4.6.2 Performance Metric

We introduce the metric of *detection probability* to measure the probability that the proposed scheme successfully detects an ill-performed aggregation. Particularly, it is measured as the number of rounds which detects the attack over the overall number of rounds in our experiment, where in each round at least one attack is simulated.

We also define a series of *percentage of deviation* for commonly used aggregation functions to measure the impact of attack when the attack is not detected. Specifically, let f and f' be the final aggregation result based on the actual histogram and the histogram obtained by the sink, respectively, for an aggregation function (e.g., *sum/average*, *median*, *standard deviation*, and *max/min*). The percentage of deviation is measured as $d = \frac{|f-f'|}{f}$.

4.6.3 Attack Models

Compromised sensors are randomly selected from non-leaf nodes. Each of them adds or subtracts δ at one or multiple buckets of the correct histogram that it is supposed to compute and send up to its parent. Here, δ is a random number between $[1, \rho * N)$. ρ is a system parameter, called *change scale*, N is the number of nodes in the network. Particularly, we identify the following three attack models:

- *Bad Behavior I*: Sensor u adds δ at randomly picked bucket j of the correct histogram it is supposed to send, and subtracts δ at another randomly picked bucket k of the histogram, where $k \neq j$.

- *Bad Behavior II*: Sensor u adds δ at a randomly picked bucket j of the correct histogram that it is supposed to send. It then randomly selects r buckets, namely, $k_1, \cdots, k_r$, such that $k_i \neq j$ for all $1 \leq i \leq r$. After then, sensor u subtracts δ_{k_i} from buckets k_i, where each δ_{k_i} is a randomly chosen number such that $1 \leq \delta_{k_i} \leq \delta$ and $\sum_{i=1}^{r} \delta_{k_i} = \delta$.

- *Bad Behavior III*: Sensor u randomly selects r buckets, namely, $k_1, \cdots, k_r$, and adds δ_{k_i} to bucket k_i, where $1 \leq \delta_{k_i} \leq \delta$ and $\sum_{i=1}^{r} \delta_{k_i} = \delta$. In the next step, sensor u randomly selects r' buckets, which are different from the previous r buckets, and subtracts $\delta_{k'_i}$ from these r' buckets, where $1 \leq \delta_{k'_i} \leq \delta$ and $\sum_{i=1}^{r'} \delta_{k'_i} = \delta$.

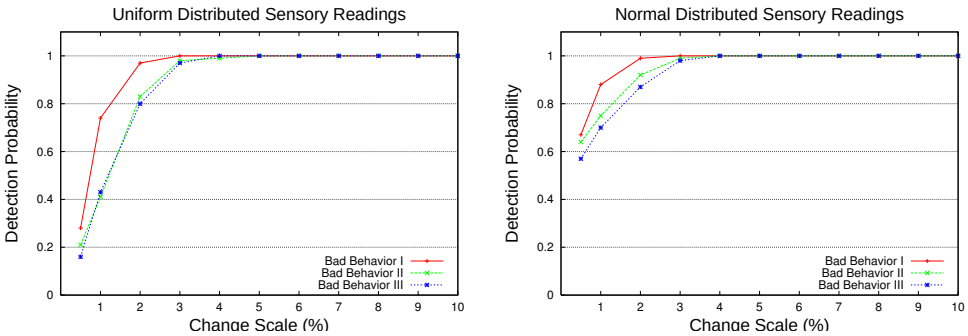
The rationality of the above attack models are based on the following observation: to avoid being detected via Line 6 of Algorithm 2, the adversary needs add δ at one or multiple buckets, meanwhile subtract δ from other buckets. However, it is highly possible to cause borrowing when subtraction is performed. The adversary does not have effective strategies to prevent borrowing from happening because the distribution of sensory readings from its subtree is unknown to the adversary.

Besides, unless otherwise stated, we set the percentage of bad nodes to 10%, the participation probability of each node to 5%, and the change scale ρ to 1%, respectively.

4.6.4 Performance Evaluation

4.6.4.1 Performance under Different Attack Models

Fig. 4.3 shows the result of detection probability under different attack models. For each attack model, the larger is the change scale, the easier is it to be detected. Among the three bad behaviors, bad behavior I is the easiest to be detected. And bad behavior II and III do not have much difference on detection probability. When the change scale is small, the detection probability is higher when the sensory data follow the normal distribution than when the data follow the uniform distribution. This is because the distribution of numbers in buckets is imbalanced when the sensory data follow a normal

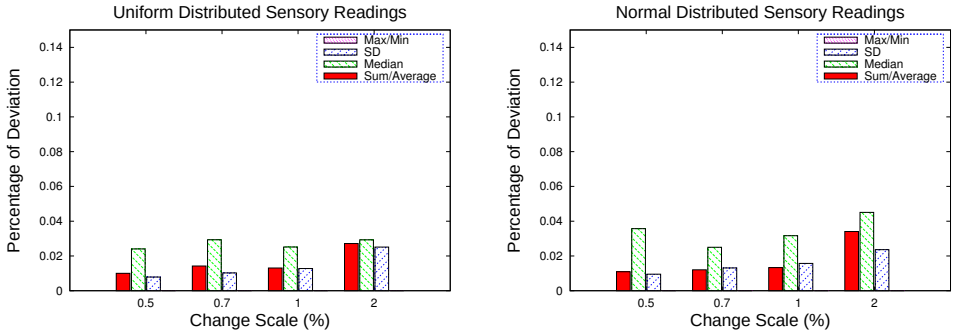


(a) (b) Mean $\mu = 49.5$, Deviation $\sigma = 30$

Figure 4.3 Performance under Different Attack Models

distributed data. This causes the overflow to be generated more easily when extra numbers are added into buckets by compromised nodes.

Considering that bad behavior I is easier to be detected than the other two and bad behavior III is the generalized case of bad behavior II, in the rest of the simulations, we study only the performance of the proposed scheme under bad behavior III. We also assume the sensory data follow uniform distribution as it is more difficult to be detected than other distributions where distribution of numbers in buckets is more imbalanced.



(a) (b) Mean $\mu = 49.5$, Deviation $\sigma = 30$

Figure 4.4 Impact of Undetected Attacks

From Fig. 4.3, we can see that the detection probability is not close to 1 when the change scale is less than 2%. We hence study the impact of undetected attacks on the aggregation results obtained by the sink. Fig. 4.4 shows the results. As we can see, the undetected attacks barely change the results

of *max/min* aggregation function. The percentages of deviation for aggregation functions *sum/average*, *median* and *standard deviation* are also quite small. Therefore, the impact of undetected attacks is tolerable for these common aggregation functions since the aggregation results do not deviate far away from the actual results.

As a large change scale makes an attack to be easily detected, in the rest of the simulation, we set change scale as 1% by default.

4.6.4.2 Detection Probability vs. Participation Probability

Considering the storage overhead, each sensor node is preloaded with information to check the data integrity attacks with certain probability. Recall that, we called it *participation probability*. Fig. 4.5 shows how the participation probability affects the detection probability. Generally, the higher is the participation probability, the higher detection probability is achieved. However, when the change scale ρ is larger than 3%, dishonest aggregation can be detected even if the participation probability is as low as 5%. Hence, we set the participation probability to 5% by default in the rest of simulation.

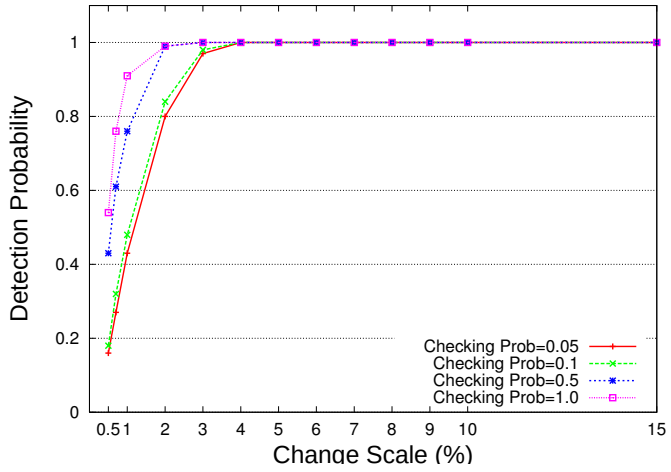


Figure 4.5 Detection Probability vs. Participation Probability

4.6.4.3 Detection Performance vs. Percentage of Compromised Nodes

Fig. 4.6 shows the detection probability as the percentage of compromised nodes varies. As for other parameters, the change scale is 1% and the participation probability is 5%. As we can see, it becomes easier to detect ill-performed aggregation as the percentage of compromised nodes increases.

On the other hand, if no ill-performed aggregation is detected, the deviation between the obtained aggregation results and the actual ones keeps at a very low level even when the number of compromised nodes increases. Particularly, the percentage of deviation is less than 7% in the worse case when the percentage of compromised nodes is no greater than 30%.

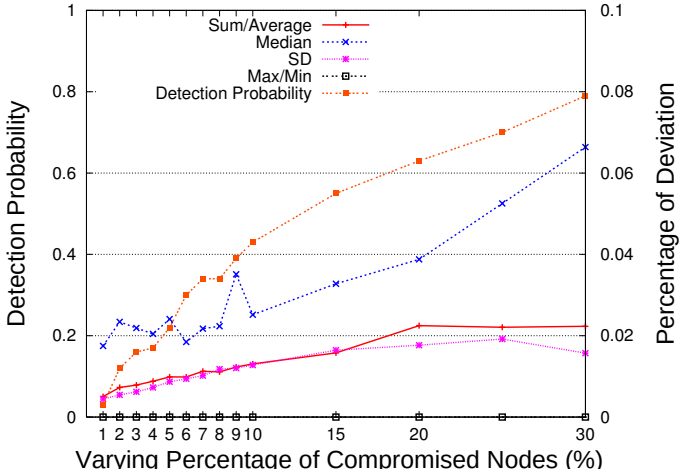
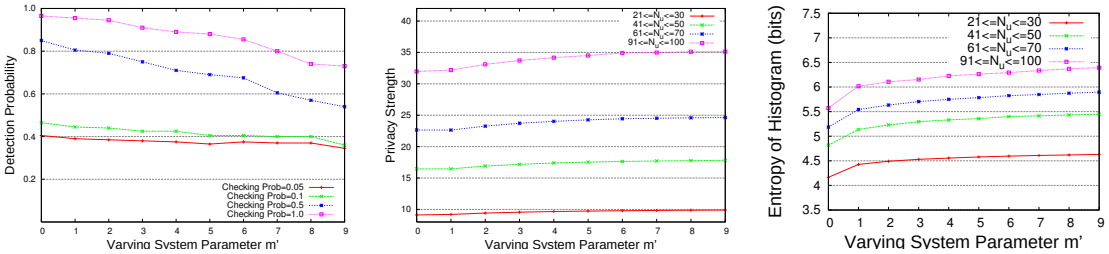


Figure 4.6 Detection Performance vs. Percentage of Bad Nodes (change scale adopted by attackers is 1%)

4.6.4.4 Detection Probability vs. System Parameter m'



(a) Detection Probability (b) Strength of Data Concealment (c) Entropy of Histogram

Figure 4.7 Detection Probability, Strength of Data Concealment, and Entropy of Histogram as m' Varies (change scale adopted by attackers is 1%)

Fig. 4.7(a) shows the detection probability as system parameter m' varies. Note that, system parameter m is fixed to 1 and the change scale is 1%. From this figure, we can see that the performance of detection probability degrades as m' increases. This is because, as m' increases, the range of $y = \{0, \cdots, m'\}$ in Eq. (4.2) also increases. That is, the necessary condition used to verify correct ag-

gregations becomes looser, which in turn increases the probability to miss detecting some ill-performed aggregations. Even though, we can see that, if the participation probability is low (say, 0.05, which is adopted by default in our simulation), the degradation is small.

Reading the absolute value of detection probability shown in Fig. 4.7(a), it may be thought that the detection probability is low and the proposed scheme is ineffective. In fact, this is untrue. The results shown in this figure is for the case that the attackers change the aggregation results slightly (i.e., the changing scale is just 1%). Recalling the study presented in Fig. 4.4, when the changing scale is small, the impact of undetected dishonest aggregation is also small.

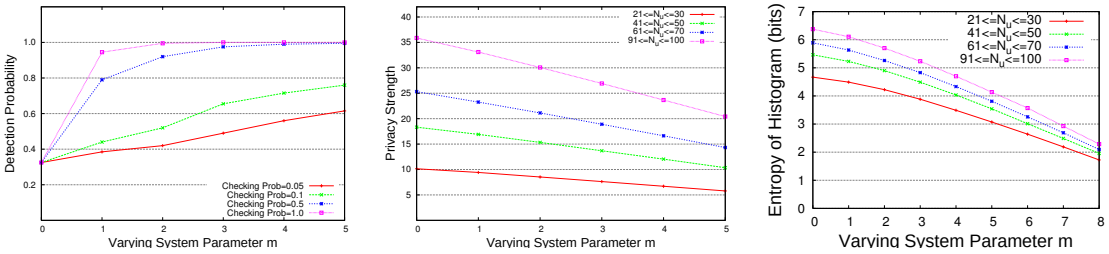
Fig. 4.7(b) shows that as m' increases, the strength of concealing numbers in individual bucket also slightly increases. This is because, increasing m' enlarges the difference between perturbed histogram and the actual histogram. Note that, if $m' = 0$, our proposed scheme is degraded to the afore-described enhanced approach, in which $\tilde{h}_{u,k}$ is always greater than or equal to $h_{u,k}$. Particularly, for nodes which have 21 to 30 descendants, the strength of data concealment is about 10, i.e., the average distance between the guessed number in a bucket and the actual number in that is around 10.

Further note that, the nodes with a large number of descendants are usually the main attacked targets. Our scheme nicely deals with this problem because, as we can see in Fig. 4.7(b), the larger is the number of descendants, the higher strength of data concealment is achieved. For example, when $m' = 5$, the strength of data concealment for a node with 21 to 30 descendants is about 10, while it is about 35 for a node with 91 to 100 descendants.

Fig. 4.7(c) plots the entropy of histogram versus system parameter m'. The experimental results are consistent with the strength of data concealment.

4.6.4.5 Detection Probability vs. System Parameter m

In this study, system parameter m varies, while m' is fixed at 2. As shown in Fig. 4.8(a), as the m increases, the detection probability also increases. This is because, the difference between perturbed histograms and actual histograms shrinks as m increases. However, as a trade-off between the detection probability and the strength of data concealment, Fig. 4.8(b) shows that, the larger is m, the weaker is the strength of data concealment. Also, Fig. 4.8(c) plots the entropy of histogram versus system parameter m. The experimental results are consistent with the strength of data concealment.



(a) Detection Probability (b) Strength of Data Concealment (c) Entropy of Histogram

Figure 4.8 Detection Probability, Strength of Data Concealment, and Entropy of Histogram as m Varies (change scale adopted by attackers is 1%)

4.6.4.6 Detection Performance vs. Query Precision

The precision of queried data is affected by the number of buckets: the more buckets are used, the higher precision is achieved. We hence vary the number of buckets in the system to see the variance of detection probability, the result is shown in Fig. 4.9. The detection probability increases as the increase of query precision. And the percentage of deviation still keeps at a very low level when the number of buckets varies.

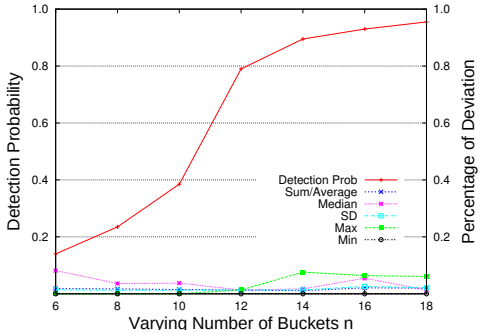


Figure 4.9 Detection Performance vs. Report Precision (change scale adopted by attackers is 1%)

Summary: The simulation results demonstrate that the proposed scheme has the following features:

- The detection probability is very high even when the participation probability is as low as 5%.
- The undetected attacks have little impact on aggregation results of aggregation functions such as max/min, standard deviation, median, sum and average.
- The data being aggregated and aggregation results are well concealed.

- Tradeoffs exist between the detection probability and the strength of data concealment as system parameters m and m' vary.
- The detection probability increases with query precision (i.e., the number of buckets), and meanwhile the deviation keeps at a low level. But, using more buckets increases the communication cost.

4.7 Prototype Implementation

We implemented the proposed scheme on TelosB Motes, which are widely used resource-constrained sensor motes produced by CrossBow. Each TelosB mote has a CPU running at 4MHz, a RAM of 10KB size, and a flash storage of 1MB size. The preloaded secrets needed for privacy-preserving detection of dishonest data aggregation is stored in the flash memory. The code size run on TelosB varies in terms of the number of buckets n. Table 4.2 shows our implementation results.

Table 4.2 Code Size (Bytes)

n	8	10	12	14	16	18	20
RAM	418	430	454	466	480	502	514
ROM	10330	10324	10354	10360	10354	10370	10378

Table 4.3 Time for Detection (Milliseconds)

Number of Buckets	8	10	12	14	16	18	20
Computational Time	15	21	32	42	52	68	81

The time for detecting ill-performed aggregation conducted by each non-leaf node is evaluated, and the results are shown in Table 5.1, where the number of buckets (i.e., n) varies from 8 to 20. By employing bit operations to compute $\tilde{h}_{u,j}$ from $\tilde{H}_u$ in Algorithm 1, the computational overhead for detecting ill-performed aggregation is very small.

4.8 Conclusion

In this chapter, we propose a simple yet effective scheme, which can protect data integrity and preserve data privacy simultaneously. Extensive analysis and experiments have been conducted and verified the effectiveness and efficiency of the scheme.

CHAPTER 5. Catching Packet Droppers and Modifiers in Wireless Sensor Networks

Packet dropping and modifying are common attacks that can be launched by an adversary to disrupt communication in wireless multi-hop sensor networks. Many schemes have been proposed to mitigate the attacks but none can effectively and efficiently identify the intruders. To address the problem, we propose a simple yet effective scheme, which can identify misbehaving forwarders that drop or modify packets.

5.1 Introduction

In wireless sensor networks, an adversary may launch various attacks [85] to disrupt the in-network communication. Among these attacks, two common ones are *dropping packets* and *modifying packets*, i.e., compromised nodes drop or modify the packets that they are supposed to forward.

To locate and identify packet droppers and modifiers, it has been proposed that nodes continuously monitor the forwarding behaviors of their neighbors [51, 52, 59, 61, 63, 64] to determine if their neighbors are misbehaving, and the approach can be extended by using the reputation-based mechanisms to allow nodes to infer whether a non-neighbor node is trustable [55, 56, 57, 58]. This methodology may be subject to high energy cost incurred by the promiscuous operating mode of wireless interface; moreover, the reputation mechanisms have to be exercised with cautions to avoid or mitigate bad mouth attacks and others. Recently, Ye *et al* proposed a probabilistic nested marking (PNM) scheme [79]. But with the PNM scheme, modified packets should not be filtered out en-route because they should be used as evidence to infer packet modifiers; hence, it cannot be used together with existing packet filtering schemes.

In this chapter, we propose a simple yet effective scheme to catch both packet droppers and modifiers. In this scheme, a routing tree rooted at the sink is first established. When sensor data is transmitted

along the tree structure towards the sink, each packet sender or forwarder adds a small number of extra bits, which is called packet marks, to the packet. The format of the small packet marks is deliberately designed such that the adversary is unable to differentiate the source of packets but the sink can obtain very useful information from the marks. Specifically, based on the packet marks, the sink can figure out the dropping rate associated with every sensor node, and then runs our proposed *node categorization algorithm* to identify nodes that are droppers/modifiers for sure or are suspicious droppers/modifiers. As the tree structure dynamically changes every time interval, behaviors of sensor nodes can be observed in a large variety of scenarios. As the information of node behaviors has been accumulated, the sink periodically runs our proposed *heuristic ranking algorithms* to identify most likely bad nodes from suspiciously bad nodes. This way, most of the bad nodes can be gradually identified with small false positive.

Our proposed scheme has the following features: (i) being effective in identifying both packet droppers and modifiers, (ii) being able to conceal packet source, (iii) low communication and energy overheads, and (iv) being compatible with existing false packet filtering schemes; that is, it can be deployed together with the false packet filtering schemes [75, 76, 77, 78], and therefore it can not only identify intruders but also filter modified packets immediately after the modification is detected. Extensive simulation has been conducted to verify the effectiveness and efficiency of the proposed scheme in various scenarios.

5.2 System Model

5.2.1 Network Assumptions

We consider a typical deployment of sensor networks, where a large number of sensor nodes are randomly deployed in a two dimensional area. Each sensor node generates sensory data periodically and all these nodes collaborate to forward packets containing the data towards a sink. The sink is located within the network. We assume all sensor nodes and the sink are loosely time synchronized [86], which is required by many applications. Attack-resilient time synchronization schemes, which have been widely investigated in wireless sensor networks [88, 89], can be employed. The sink is aware of the network topology, which can be achieved by requiring nodes to report their neighboring nodes right

after deployment.

5.2.2 Security Assumptions and Attack Model

We assume the network sink is trustworthy and free of compromise, and the adversary cannot successfully compromise regular sensor nodes during the short topology establishment phase after the network is deployed. This assumption has been widely made in existing work [77, 70]. After then, the regular sensor nodes can be compromised. Compromised nodes may or may not collude with each other. A compromised node can launch the following two attacks:

- Packet dropping: a compromised node drops all or some of the packets that it is supposed to forward. It may also drop the data generated by itself for some malicious purpose such as framing innocent nodes.

- Packet modification: a compromised node modifies all or some of the packets that it is supposed to forward. It may also modify the data it generates to protect itself from being identified or to accuse other nodes.

5.3 The Proposed Scheme

Our proposed scheme consists of a system initialization phase and several equal-duration rounds of intruder identification phases.

- In the initialization phase, sensor nodes form a topology which is a directed acyclic graph (DAG). A routing tree is extracted from the DAG. Data reports follow the routing tree structure.

- In each round, data is transferred through the routing tree to the sink. Each packet sender/ forwarder adds a small number of extra bits to the packet and also encrypts the packet. When one round finishes, based on the extra bits carried in the received packets, the sink runs a node categorization algorithm to identify nodes that must be bad (i.e., packet droppers or modifiers) and nodes that are suspiciously bad (i.e., suspected to be packet droppers and modifiers).

- The routing tree is reshaped every round. As a certain number of rounds have passed, the sink will have collected information about node behaviors in different routing topologies. The information

includes which nodes are bad for sure, which nodes are suspiciously bad, and the nodes' topological relationship. To further identify bad nodes from the potentially large number of suspiciously bad nodes, the sink runs heuristic ranking algorithms.

In the following sub-sections, we first present the algorithm for DAG establishment and packet transmission, which is followed by our proposed categorization algorithm, tree structure reshaping algorithm, and heuristic ranking algorithms. To ease the presentation, we first concentrate on packet droppers and assume no node collusion. After that, we present how to extend the presented scheme to handle node collusion and detect packet modifiers, respectively.

5.3.1 DAG Establishment and Packet Transmission

All sensor nodes form a DAG rooted at the sink and extracts a routing tree from the DAG. The sink knows the DAG and the routing tree, and shares a unique key with each node. When a node wants to send out a packet, it attaches to the packet a sequence number, encrypts the packet only with the key shared with the sink, and then forwards the packet to its parent on the routing tree. When an innocent intermediate node receives a packet, it attaches a few bits to the packet to mark the forwarding path of the packet, encrypts the packet, and then forwards the packet to its parent. On the contrary, a misbehaving intermediate node may drop a packet it receives. On receiving a packet, the sink decrypts it, and thus finds out the original sender and the packet sequence number. The sink tracks the sequence numbers of received packets for every node, and for every certain time interval, which we call a *round*, it calculates the packet dropping rate for every node. Based on the dropping rate and the knowledge of the topology, the sink identifies packet droppers based on rules we derive. In detail, the scheme includes the following components, which are elaborated in the following.

5.3.1.1 System Initialization

The purpose of system initialization is to set up secret pair-wise keys between the sink and every regular sensor node, and to establish the DAG and the routing tree to facilitate packet forwarding from every sensor node to the sink.

Preloading Keys and Other System Parameters Each sensor node u is preloaded the following information:

- K_u: a secret key exclusively shared between the node and the sink.
- L_r: the duration of a round.
- N_p: the maximum number of parent nodes that each node records during the DAG establishment procedure.
- N_s: the maximum packet sequence number. For each sensor node, its first packet has sequence number 0, the N_s^{th} packet is numbered $N_s - 1$, the $(N_s + 1)^{th}$ packet is numbered 0, and so on and so forth.

Topology Establishment After deployment, the sink broadcasts to its one-hop neighbors a 2-tuple $\langle 0, 0 \rangle$. In the 2-tuple, the first field is the ID of the sender (We assume the ID of sink is 0.) and the second field is its distance in hop from the sender to the sink. Each of the remaining nodes, assuming its ID is u, acts as follows:

(i) On receiving the first 2-tuple $\langle v, d_v \rangle$, node u sets its own distance to the sink as $d_u = d_v + 1$.

(ii) Node u records each node w (including node v) as its parent on the DAG if it has received $\langle w, d_w \rangle$ where $d_w = d_v$. That is, node u records as its parents on the DAG the nodes whose distance (in hops) to the sink is the same and the distance is one hop shorter than its own. If the number of such parents is greater than N_p, only N_p parents are recorded while others are discarded. The actual number of parents it has recorded is denoted by $n_{p,u}$.

(iii) After a certain time interval[1], node u broadcasts 2-tuple $\langle u, d_u \rangle$ to let its downstream one-hop neighbors to continue the process of DAG establishment. Then, among the recorded parents on the DAG, node u randomly picks one (whose ID is denoted as P_u) as its parent on the routing tree. Node u also picks a random number (which is denoted as R_u) between 0 and $N_p - 1$. As to be elaborated later, random number R_u is used as a short ID of node u to be attached to each

[1]The length of the interval is a predefined system parameter that is large enough for each node to receive an enough number of broadcasts from the nodes closer to the sink.

packet node u forwards, so that the sink can trace out the forwarding path. Finally, node u sends P_u, R_u and all recorded parents on the DAG to the sink.

After the above procedure completes, a DAG and a routing tree rooted at the sink is established. The routing tree is used by the nodes to forward sensory data until the tree changes later; when the tree needs to be changed, the new structure is still extracted from the DAG.

The life time of the network is divided into rounds, and each round has a time length of L_r. After the sink has received the parent lists from all sensor nodes, it sends out a message to announce the start of the first round, and the message is forwarded hop by hop to all nodes in the network. Note that, each sensor node sends and forwards data via a routing tree which is implicitly agreed with the sink in each round, and the routing tree changes in each round via our tree reshaping algorithm presented in Sec. 5.3.3.

5.3.1.2 Packet Sending and Forwarding

Each node maintains a counter C_p which keeps track of the number of packets that it has sent so far. When a sensor node u has a data item D to report, it composes and sends the following packet to its parent node P_u:

$$\langle P_u, \{R_u, u, C_p \text{ MOD } N_s, D, pad_{u,0}\}_{K_u}, pad_{u,1}\rangle,$$

where C_p MOD N_s is the sequence number of the packet. R_u ($0 \leq R_u \leq N_p - 1$) is a random number picked by node u during the system initialization phase, and R_u is attached to the packet to enable the sink to find out the path along which the packet is forwarded. $\{X\}_Y$ represents the result of encrypting X using key Y.

Padding $pad_{u,0}$ and $pad_{u,1}$ are added to make all packets equal length, such that forwarding nodes cannot tell packet sources based on packet length. Meanwhile, the sink can still decrypt the packet to find out the actual content. To satisfy these two objectives simultaneously, the padding is constructed as follows:

- For a packet sent by a node which is h hops away from the sink, the length of $pad_{u,1}$ is $\log(N_p) * (h-1)$ bits. As to be described later, when a packet is forwarded for one hop, $\log(N_p)$ bits information will be added and meanwhile, $\log(N_p)$ bits will be chopped.

- Let the maximum size of a packet be L_p bits, a node ID be L_{id} bits and data D be L_D bits. $pad_{u,0}$ should be $L_p - L_{id} * 2 - \log(N_p) * h - \log(N_s) - L_D$ bits, where $L_{id} * 2$ bits are for P_u and u fields in the packet, field R_u is $\log(N_p)$ bits long, field $pad_{u,1}$ is $\log(N_p) * (h-1)$ bits long, and C_p MOD N_s is $\log(N_s)$. Setting $pad_{u,0}$ to this value ensures that all packets in the network have the same size L_p.

When a sensor node v receives packet $\langle v, m\rangle$, it composes and forwards the following packet to its parent node P_v:

$$\langle P_v, \{R_v, m'\}_{K_v}\rangle,$$

where m' is obtained by trimming the rightmost $\log(N_p)$ bits off m. Meanwhile, R_v, which has $\log N_p$ bits, is added to the front of m'. Hence, the size of the packet keeps unchanged. Suppose on a routing tree, node u is the parent of node v and v is a parent of node w. When u receives a packet from v, it cannot differentiate whether the packet is originally sent by v or w unless nodes u and v collude. Hence, the above deliberate packet sending and forwarding scheme results in the difficulty to launch selective dropping, which is leveraged in locating packet droppers. We take special consideration for the collusion scenario, which is to be elaborated later.

5.3.1.3 Packet Receiving at the Sink

We use node 0 to denote the sink. When the sink receives a packet $\langle 0, m'\rangle$, it conducts the following steps:

(i) Initialization: We introduce two temporary variables u and m. Let $u = 0$ and $m = m'$.

(ii) The sink attempts to find out a child of node u, denoted as v, such that $dec(K_v, m)$ results in a string starting with R_v, where $dec(K_v, m)$ means the result of decrypting m with key K_v.

(iii) If the attempt fails for all children nodes of node u, the packet is identified as have been modified and thus should be dropped.

(iv) If the attempt succeeds, it indicates that the packet was forwarded from node v to node u. Now, there are two cases:

- If $dec(K_v, m)$ starts with $\langle R_v, v\rangle$, it indicates that node v is the original sender of the packet. The sequence number of the packet is recorded for further calculation and the receipt procedure completes.
- Otherwise, it indicates that node v is an intermediate forwarder of the packet. Then, u is updated to be v, m is updated to be the string obtained by trimming R_v from the leftmost. Then, steps (ii)-(iv) are repeated.

The process of packet receipt at the sink can be formalized as Algorithm 3

Algorithm 3 Packet Receipt at the Sink

```
1: Input: packet ⟨0, m⟩.
2: u = 0, m' = m;
3: hasSuccAttemp = false;
4: for each child node v of node u do
5:    P=dec(K_v, m');
6:    if decryption fails then
7:       continue;
8:    else
9:       hasSuccAttemp = true;
10:      if P starts with ⟨R_v, v⟩ then
11:         record the sequence number; /* v is the sender */
12:         break;
13:      else
14:         trim R_v from P and get m'; /*v is a forwarder */
15:         u ← v, hasSuccAttemp = false; go to line 4;
16: if hasSuccAttemp = false then
17:    drop this packet;
```

5.3.1.4 An Example

Fig 5.1 shows an example sensor network with 7 nodes, node $0-6$. Node ID is represented by 3 bits. Suppose the maximum packet sequence number N_s is 16 and 4 bits are used to represent the counter C_p. N_p, the maximum number of parents that each sensor node should record during the tree establishment, is 4. We assume that the length of sensory data L_D is 8 bits. In this figure, we illustrate the procedure when node 5, which is 2 hops away from the sink, generates sensory data 96, how the data is sent to the sink node 0. Assume currently, data from node 5 follows path 5->2->0 and $C_p = 3$.

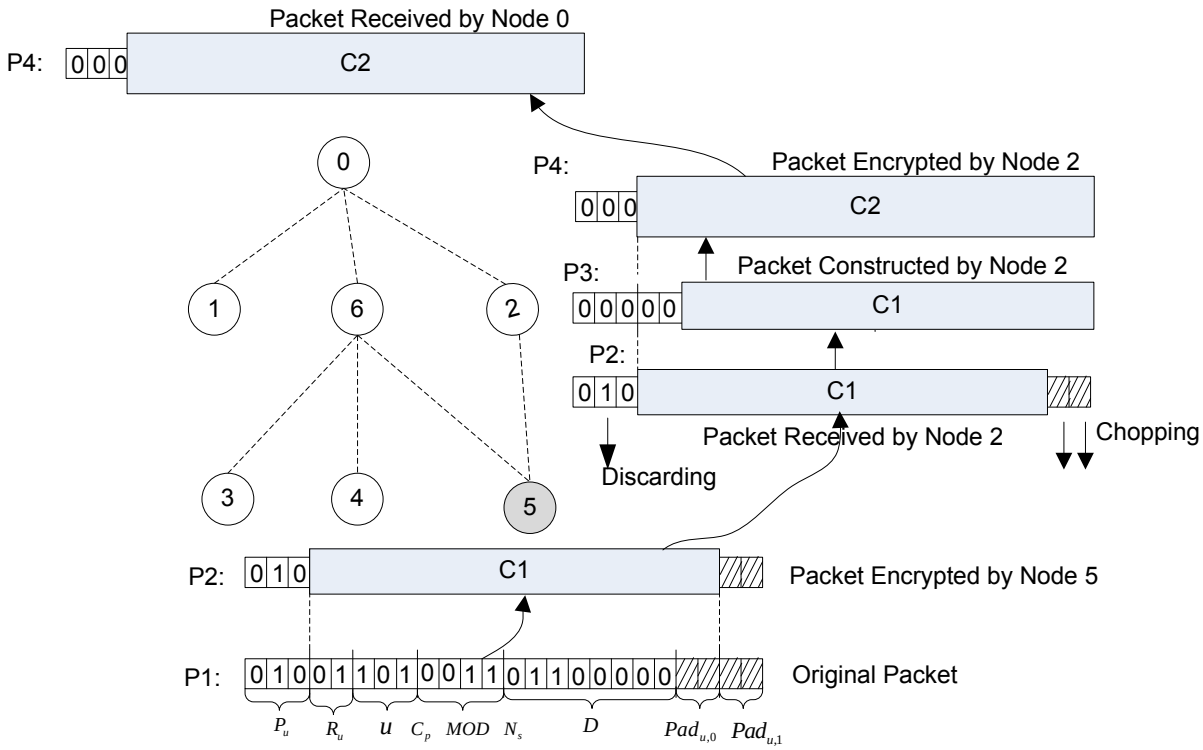


Figure 5.1 Example of Packet Sending, Forwarding

Node 5 constructs packet

$$< P_u, \{R_u, u, C_p \; MOD \; N_s, D, pad_{u,0}\}_{K_u}, pad_{u,1} >,$$

The plain-text of the packet is shown in the figure as P1. Specifically, $P_u = 2(010)$, $R_u = 1(01)$, $u = 5(101)$, $C_p = 3(0011)$, and $D = 96(01100000)$. The length of the paddings is calculated as follows. Assume that the maximum packet size L_p is 24 bits, the length of $pad_{5,1}$ should be $\log N_p * (h-1)$ bits, that is 2 bits. The length of $pad_{5,0}$ should be $L_p - L_{id} * 2 - \log N_p * h - \log N_s - L_D$, that is, $24 - 3 * 2 - 2 * 2 - 4 - 8 = 2$ bits. Based on P1, node 5 uses its secret key K_5 to encrypt part of $P1$, $\{R_5, 5, C_p \; MOD \; N_s, D, pad_{5,0}\}$. The cipher-text is represented by $C1$ and the encrypted packet $P2$ is constructed accordingly. P_2 is sent to node 2.

When node 2 receives packet $P2$, it first chops the rightmost $\log N_p$ bits, which is 2 bits of the paddings. Next, node 2 constructs packet $P3$ by adding its parent ID and the random number R_2 to the front of cipher-text $C1$. Note that the packet length is kept the same since the right most 2 bits is chopped, and 2 bits random number R_2 is added. Next, node 2 uses its secret key K_2 to encrypt information $\{R_2, C1\}$ in packet $P3$ and generates packet $P4$. $P4$ is then sent to the sink.

After the sink receives the packet $P4$ from its children, the sink tries to figure out the sender. The sink tries to decrypt the cipher-text $C2$ by using its children's secret keys one by one. The sink finds that the packet is from node 2 after $C2$ is decrypted by using K_2. The sink also recovers the decrypted $C2$ which does not starts with $\{R_2, 2\}$. (Note that, the sink and each sensor node are synchronized and they follow an implicit tree reshaping algorithm. The random number R_2 is also known by the sink.) The sink concludes that node 2 is an intermediate node. It continues this process and finds out the source of the data is node 5.

5.3.2 Node Categorization Algorithm

In every round, for each sensor node u, the sink keeps track of the number of packets sent from u, the sequence numbers of these packets and the number of flips in the sequence numbers of these packets, (i.e., the sequence number changes from a large number such as $N_s - 1$ to a small number such as 0). In the end of each round, the sink calculates the dropping rate for each node u. Suppose $n_{u,max}$ is the most recently seen sequence number, $n_{u,flip}$ is the number of sequence number flips and $n_{u,rcv}$ is the number of received packets. The dropping ratio in this round is calculated as follows:

$$d_u = \frac{n_{u,flip} * N_s + n_{u,max} + 1 - n_{u,rcv}}{n_{u,flip} * N_s + n_{u,max} + 1}.$$

Based on the dropping rate of every sensor node and the tree topology, the sink identifies the nodes that are droppers for sure and that are possibly droppers. For this purpose, a threshold θ is first introduced. We assume that if a node's packets are not intentionally dropped by forwarding nodes, the dropping rate of this node should be lower than θ. Note that θ should be greater than 0, taking into account droppings caused by incidental reasons such as collisions. The first step of the identification is to mark each node with "+" if its dropping ratio is lower than θ, or with "-" otherwise. After then, for each path from a leaf node to the sink, the nodes' mark pattern in this path can be decomposed into any combination of the following basic patterns, which are also illustrated by Fig. 5.2:

- $+\{+\}$: a node and its parent node are marked as "+".
- $+-\{-\}^*$: a node is marked as "+", but its one or more continuous immediate upstream nodes are marked as "-".

- $-\{+\}$: a node is marked as "-", but its parent node is marked as "+".
- $-\{-\}$: a node and its parent node are marked as "-".

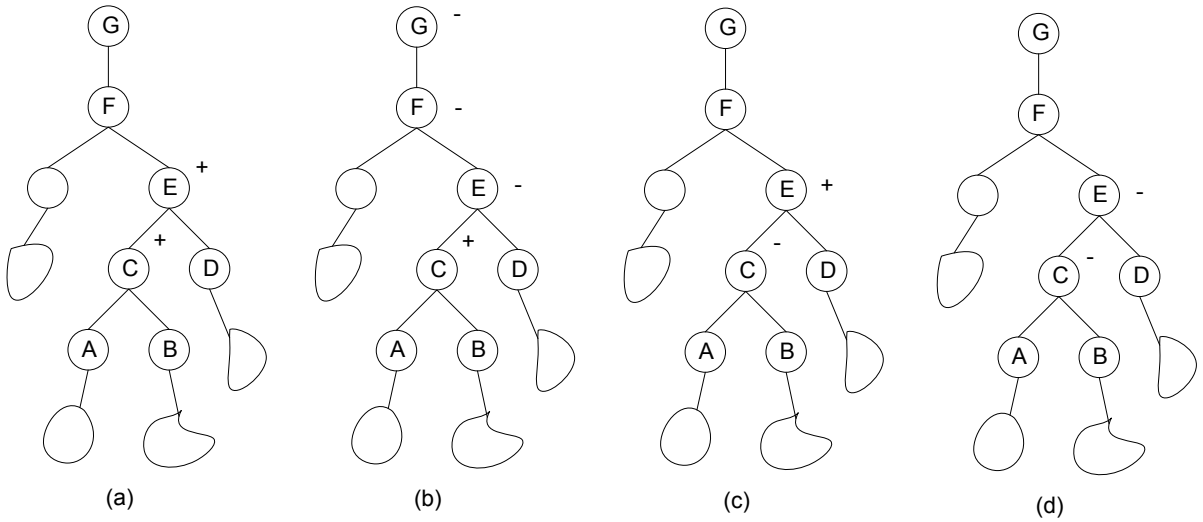


Figure 5.2 Node Status Pattern

For each of the above cases, we can infer whether a node (i) has dropped packets (called *bad for sure*), (ii) is suspected to have dropped packets (called *suspiciously bad*), (ii) has not been found to drop packets (called *temporarily good*), or (iv) must have not dropped packets (called *good for sure*):

Case 1: $+\{+\}$. The node and its parent node do not drop packets along the involved path, but it is unknown whether they drop packets on other forwarding paths. Therefore, the sink infers that these nodes are *temporally good*. For example, in Fig. 5.2(a), node C, E are marked "+" and are regarded as temporally good. A special case is, if a leaf node is marked as "+", it is safe to infer it is good since it cannot drop other's packets.

Case 2: $+-\{-\}^*$. In the case, all nodes marked as "-" must be *bad for sure*. To show the correctness of this rule, we prove it by contradiction. Without loss of generality, we examine the scenario illustrated in Fig 5.2(b), where node C is marked as "+", and node E, F, G are marked as "-". If our conclusion is incorrect and node E is good, E must not drop its own packets. Since E is marked as "-", there must be some upstream nodes of E dropping E's packets. Note that the bad upstream nodes are at least one hop above E, i.e., at least two hops above C. It is impossible for them to differentiate packets from E and C, so they cannot selectively drop the packets from E while forwarding the packets from C. Even if C and

the bad upstream node collude, they cannot achieve this result. This is because every packet from C must go through and be encrypted by E, and therefore the bad upstream node cannot tell the source of the packet to perform selective dropping. Note that, if a packet is forwarded to the bad upstream node without going through E, the packet cannot be correctly decrypted by the sink and thus will be dropped. Therefore, E must be bad. Similarly, we can also conclude that F and G are also bad.

Case 3: $-\{+\}$. In this case, either the node marked as "-" or its parent marked as "+" must be bad. But it cannot be further inferred whether (i) only the node with "-" is bad, (ii) only the node with "+" is bad, or (iii) both nodes are bad. Therefore, it is concluded that both nodes are *suspiciously bad*. The correctness of this rule can also be proved by contradiction. Without loss of generality, let us consider the scenario shown in Fig. 5.2(c), where node C is marked as "-", and node E is marked as "+". Now suppose both C and E are good, hence there must be at least one upstream node of E which is the bad node that drops the packets sent by C. However, it is impossible to find such an upstream node since node F, G, and other upstream nodes cannot selectively drop packets from node C while forwarding packets from node E. Hence, either node C is bad or node E is bad in this case.

Case 4: $-\{-\}$. In this case, every node marked with "-" could be bad or good. Conservatively, they have to be considered as *suspiciously bad*. Specifically, suppose v is the highest-level node that is marked as "-", and u is its parent node. If u is the sink, v must be *bad for sure*, otherwise, both u and v are *suspiciously bad*. On the other hand, suppose v is a child of u and they are both marked as "-", if the dropping rate of u is larger than that of v by at least θ (i.e., $d_v < d_u$ and $d_u - d_v > \theta$, recalling that θ is a threshold used to tolerate incidental droppings), then node u is *bad for sure*. Otherwise, both u and v are *suspiciously bad*.

Based on the above rules, we develop a node categorization algorithm to find nodes that are *bad for sure* or *suspiciously bad*. The formal algorithm is presented in Algorithm 4.

5.3.3 Tree Reshaping and Ranking Algorithms

The tree used to forward data is dynamically changed from round to round, which enables the sink to observe the behavior of every sensor node in a large variety of routing topologies. For each of these scenarios, node categorization algorithm is applied to identify sensor nodes that are bad for sure or suspiciously bad. After multiple rounds, sink further identifies bad nodes from those are suspiciously

Algorithm 4 Tree-Based Node Categorization Algorithm

Input: Tree T, with each node u marked by "$+$" or "$-$", and its dropping rate d_u.
for each leaf node u in T **do**
  $v \leftarrow u$'s parent;
  while u is not the Sink **do**
    if $u.mark =$ " $+$ " **then**
      if $v.mark =$ "$-$" **then**
        $b \leftarrow v$;
        repeat
          $e \leftarrow v$;
          $v \leftarrow v$'s parents node;
        until $v.mark =$ " $+$ " or v is Sink
        Set nodes from b to e as *bad for sure*;
    else
      if v is Sink **then**
        Set u as *bad for sure*;
      if $v.mark =$ " $+$ " **then**
        if v is not *bad for sure* **then**
          Set u and v as *suspiciously bad*;
      else
        if $d_v - d_u > \theta$ **then**
          Set v as *bad for sure*;
        else if $d_u - d_v > \theta$ **then**
          Set u and v as *suspiciously bad*;
    $u \leftarrow v$, $v \leftarrow v$'s parents node

bad by applying several proposed heuristic methods.

5.3.3.1 Tree Reshaping

The tree used for forwarding data from sensor nodes to the sink is dynamically changed from round to round. In other words, each sensor node may have a different parent node from round to round. To let the sink and the nodes have a consistent view of their parent nodes, the tree is reshaped as follows. Suppose each sensor node u is preloaded with a hash function $h(.)$ and a secret number K_u which is exclusively shared with the sink. At the beginning of each round i ($i = 1, 2, \cdots$), node u picks the $[h^i(K_u)$ MOD $n_{p,u}]^{th}$ parent node as its parent node for this round, where $h^i(K_u) = h(h^{i-1}(K_u))$ and $n_{p,u}$ is the number of candidate parent nodes that node u recorded during the tree establishment phase. Recall that node u's candidate parent nodes are those which are one hop closer to the sink and within node u's communication range. Therefore, if node u chooses node w as its parent in a round, node w will not select node u as its parent, and the routing cycle will not occur in our scheme. Note that, how the parents are selected is predetermined by both the preloaded secret K_u and the list of parents recorded in the tree establishment phase. The selection is implicitly agreed between each node and the sink. Therefore, a misbehaving node cannot arbitrarily select its parent in favor of its attacks.

5.3.3.2 Identifying Most Likely Bad Nodes from Suspiciously Bad Nodes

We rank the suspiciously bad nodes based on their probability to be bad, and identify part of them as most likely bad nodes. Specifically, after a round ends, the sink calculates the dropping rate of each node, and runs node categorization algorithm specified as Algorithm 4 to identify nodes that are bad for sure or suspiciously bad. Since the number of suspiciously bad nodes is potentially large, we propose how to identify most likely bad nodes from the suspiciously bad nodes as follows. By examining the rules in Case 3 and Case 4 for identifying suspiciously bad nodes, we can see that, in each of these cases (i) there are two nodes, denoted as u and v, which have the same probability to be the bad nodes and (ii) at least one of them must be bad. We call these two nodes as a *suspicious pair*. For each round i, all identified suspicious pairs are recorded in a *suspicious set* denoted as $S_i = \{\langle u_j, v_j\rangle | \langle u_j, v_j\rangle$ is a suspicious pair and $\langle u_j, v_j\rangle = \langle v_j, u_j\rangle\}$. Therefore, after n rounds of detection, we can obtain a series of suspicious sets: $S_1, S_2, \cdots, S_n$.

We define $\overline{S}$ as the ***set of most likely bad nodes*** identified from $S_1, S_2, \cdots, S_n$, if $\overline{S}$ has the following properties:

- *Coverage*. $\forall \langle u, v \rangle \in S_i$ $(i = 1, \cdots, n)$, it must hold that either $u \in \overline{S}$ or $v \in \overline{S}$. That is, for any identified suspicious pair, at least one of the nodes in the pair must be in the set of most likely bad nodes.
- *Most-likeliness*. $\forall \langle u, v \rangle \in S_i$ $(i = 1, \cdots, n)$, if $u \in \overline{S}$ but $v \notin \overline{S}$, then u must have higher probability to be bad than v based on n rounds of observation.
- *Minimality*. The size of $\overline{S}$ should be as small as possible in order to minimize the probability of mis-accusing innocent nodes.

Among the above three conditions, the first one and the third one can be relatively easily implemented and verified. For the second condition, we propose several heuristics to find nodes with *most-likeliness*.

Global Ranking-Based (GR) Method

The GR method is based on the heuristic that, the more times a node identified as suspiciously bad are, the more likely it is a bad node. With this method, each suspicious node u is associated with an *accused account* which keeps track of the time that the node has been identified as suspiciously bad nodes. To find out the most likely set of suspicious nodes after n rounds of detection, as described in Algorithm 5, all suspicious nodes are ranked based on the descending order of the values of their accused accounts. The node with the highest value is chosen as a most likely bad node and all the pairs that contain this node are removed from $S_1, \cdots, S_n$, resulting in new sets. The process continues on the new sets until all suspicious pairs have been removed. The GR method is formally presented in Algorithm 5.

Stepwise Ranking-Based (SR) method

It can be anticipated that the GR method will falsely accuse innocent nodes that have frequently been parents or children of bad nodes: as parents or children of bad nodes, according to previously-described rules in Cases 3 and 4, the innocents can often be classified as suspiciously bad nodes. To reduce false accusation, we propose the SR method. With the SR method, the node with the highest accused account value is still identified as a most likely bad node. However, once a bad node u is

Algorithm 5 The Global Ranking-Based Approach

1: Sort all suspicious nodes into queue Q according to the descending order of their accused account values
2: $\overline{S} \leftarrow \emptyset$
3: **while** $\bigcup_{i=1}^{n} S_i \neq \emptyset$ **do**
4: $u \leftarrow deque(Q)$
5: $\overline{S} \leftarrow \overline{S} \wedge \{u\}$
6: remove all $\langle u, * \rangle$ from $\bigcup_{i=1}^{n} S_i$

identified, for any other node v that has been suspected together with node u, the value of node v's accused account is reduced by the times that u and v have been suspected together. This adjustment is motivated by the possibility that v has been framed by node u. After the adjustment, the node that has the highest value of accused account among the rest nodes is identified as the next mostly like bad node, which is followed by the adjustment of the accused account values for the nodes that have been suspected together with the node. Note that, similar to the GR method, after a node u is identified as bad, all suspicious pairs with format $\langle u, * \rangle$ are removed from $S_1, \cdots, S_n$. The above process continues until all suspicious pairs have been removed. The SR method is formally presented in Algorithm 6.

Algorithm 6 The Stepwise Ranking-Based Approach

1: $\overline{S} \leftarrow \emptyset$
2: **while** $\bigcup_{i=1}^{n} S_i \neq \emptyset$ **do**
3: $u \leftarrow$ the node has the maximum times of presence in $S_1, \cdots, S_n$
4: $\overline{S} \leftarrow \overline{S} \wedge \{u\}$
5: remove all $\langle u, * \rangle$ from $\bigcup_{i=1}^{n} S_i$

Hybrid Ranking-Based (HR) Method

The GR method can detect most bad nodes with some false accusations while the SR method has fewer false accusations but may not detect as many bad nodes as the GR method. To balance the trade-off, we further propose the HR method, which is formally presented in Algorithm 7. According to HR, the node with the highest accused account value is still first chosen as most likely bad node. After a most likely bad node has been chosen, the one has the highest accused account value among the rest is chosen only if the node has not always been accused together with the bad nodes that have been identified already. Thus, the accusation account value is considered as an important criterion in identification, as in the GR method; meanwhile, the possibility that an innocent node being framed by bad nodes is also

considered by not choosing the nodes who have always being suspected together with already-identified bad nodes, as in the SR method. The HR method is formally presented in Algorithm 7.

Algorithm 7 The Hybrid Ranking-Based Approach

1: Sort all suspicious nodes into queue Q according to the descending order of their accused account values
2: $\overline{S} \leftarrow \emptyset$
3: **while** $\bigcup_{i=1}^{n} S_i \neq \emptyset$ **do**
4: $u \leftarrow deque(Q)$
5: **if** there exists $\langle u, * \rangle \in \bigcup_{i=1}^{n} S_i$ **then**
6: $\overline{S} \leftarrow \overline{S} \wedge \{u\}$
7: remove all $\langle u, * \rangle$ from $\bigcup_{i=1}^{n} S_i$

5.3.4 Handling Collusion

Because of the deliberate hop by hop packet padding and encryption, the packets are not distinguishable to the upstream compromised nodes as long as they have been forwarded by an innocent node. The capability of launching collusion attacks is thus very limited by the design of the proposed scheme. However, compromised nodes that are located close with each other may collude to render the sink to accuse some innocent nodes. We discuss the possible collusion scenarios in this section and propose our neutralized strategies to mitigate the effects of collusion.

As the four cases described in section 5.3.2, the attackers do not gain any benefit if the collusion triggers the scenarios of Case 1 and Case 2. However, the attackers may accuse honest nodes if the collusion triggers the scenarios of Case 3 and Case 4. By exploiting the rules used by the node categorization algorithm and rank algorithm, there are two possible collusion strategies to make the sink accuse innocent nodes. We use Fig. 5.3 as a general example to discuss the collusion scenarios.

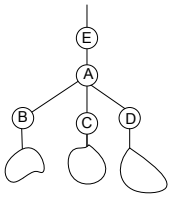


Figure 5.3 Collusion Scenarios

- Horizontal Collusion: if nodes B, C and D are compromised and collude, they will drop all or

some of the packets of their own and their downstream nodes. Consequently, according to the rules in Case 3, $\langle A, B\rangle$, $\langle A, C\rangle$ and $\langle A, D\rangle$ are all identified as pairs of suspiciously bad nodes. Since A has been suspected for more times than B, C and D, it is likely that A is falsely identified as bad node.

- Vertical Collusion: if nodes B and E are compromised and collude, B may drop some packets of itself and its downstream nodes, and then E further drops packets from its downstream nodes including B and B's downstream nodes. Note that, E cannot differentiate the packets forwarding/generating by B since they are encrypted by node A. Consequently, the dropping rates for B and its downstream nodes are higher than that for node A. According to Case 4, $\langle E, A\rangle$ and $\langle A, B\rangle$ are both identified as pairs of suspiciously bad nodes. Since A has been suspected for more times than B and E, it is likely to be identified as bad node.

To defeat collusion that may lead to false accusation, our scheme is extended as follows:

- The concept of *suspicious pair* is extended to *suspicious tuple* which is a non-ordered sequence of suspicious nodes. Note that, a suspicious pair is a special case of suspicious tuple, i.e., suspicious 2-tuple.

- A new rule is introduced: for each round i, if there exists multiple suspicious tuples of which each contains a certain node u, $\langle u, v_{1,1}, \cdots, v_{1,m_1}\rangle, \cdots, \langle u, v_{n,1}, \cdots, v_{n,m_n}\rangle$, all these tuples should be combined into a single tuple without duplication. For example, if the original tuples are $\langle u, v_1\rangle$, $\langle u, v_2, v_3\rangle$ and $\langle u, v_3\rangle$, these tuples will be replaced with $\langle u, v_1, v_2, v_3\rangle$, where each of the four nodes is suspected for only once.

As to be shown in our simulation results, the above enhancement can deal with collusion at the cost of slightly degraded detection rate.

5.3.5 An Extension for Identifying Packet Modifiers

If a compromised node modifies the packets that it is supposed to forward, the node can be detected with the afore-described scheme. This is because modified packets will be detected by the sink and thus be dropped (detailed in step (iii) of the packet receipt procedure at sink). This is equivalent to

that the packets are dropped by the modifier; hence, the packet modifier can be identified as a packet dropper. However, detecting modifiers in this way is not ideal because modified packets cannot be identified earlier by en-route nodes to save energy and bandwidth consumption. To enable en-route detection of modifications, the afore-described procedures for packet sending and forwarding can be slightly modified as follows.

When a node u has a data item D to report, it can obtain endorsement message authentication codes (MACs) from its neighbors, which are denoted as $MAC(D)$, following existing en-route filtering schemes such as the statistical en-route filtering scheme (SEF) [75] and the interleaved hop-by-hop authentication scheme [76]. The source node u generates and sends the following packet to its parent node P_u:

$$\langle P_u, D, MAC(D), \{R_u, u, C_p \text{ MOD } N_s, pad_{u,0}\}_{K_u}, pad_{u,1}\rangle.$$

When packet $\langle v, D, MAC(D), m\rangle$ is received by an en-route node v, node v can check the integrity of D in the same way as in existing packet filtering schemes [75, 76]. If a packet is found modified, it is immediately dropped; otherwise, the following packet is forwarded by v:

$$\langle P_v, D, MAC(D), \{R_v, m'\}_{K_v}\rangle,$$

where m' is constructed the same way from m as in the scheme to identify packet droppers.

Therefore, by integrating with the existing schemes [75, 76], the modified packets will be dropped by honest nodes on the way to the sink. Modified packets dropped by honest nodes are equivalent to packets dropped at the modifier nodes, which can be explained via Fig.5.4. Suppose node A is a compromised node and it modifies the packets it forwards randomly due to our deliberate packet encryption and padding techniques. And suppose node C, node B and node D as well as node B and node D's downstream nodes are honest nodes. Node E detects that some packets it receives have been modified, node E then drops the modified packets. Since honest nodes only drop modified packets and forward unmodified packets correctly, dropping the modified packets then will only affect the marks of nodes whose packets pass through the modifiers. In this example, only the marks (i.e., "$+$" or "$-$") of node A and its downstream nodes will be affected by node E's dropping behavior and the marks of node C, node B and node D as well as node B and node D's downstream nodes will not be affected by honest node E's dropping behavior. Therefore, modified packets dropped by honest nodes are equivalent to

packets dropped at the modifier nodes in terms of the marks of each node. The sink then does not need to differentiate honest nodes' behavior of dropping modified packets and compromised nodes' behavior of dropping correct packets.

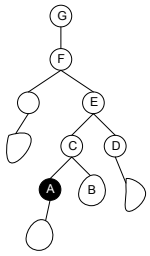


Figure 5.4 Detect Packet Modifiers

5.4 Performance Evaluation

5.4.1 Objectives, Metrics, and Methodology

Our packet dropper/modifier identification scheme is implemented in the ns-2 simulator (version 2.30) to evaluate the effectiveness and efficiency of the proposed scheme. Our objectives in conducting this evaluation study are four-fold: firstly, testing the effectiveness and efficiency of our scheme in identifying packet droppers and modifiers; secondly, studying the impacts of various system parameters (i.e., sensor data reporting interval, round length, percentage of bad nodes, network scale, presence of node collusions, etc.) on the performance of our scheme; thirdly, testing the effectiveness of our scheme under six different attack models; finally, comparing the proposed global ranking (GR), stepwise ranking (SR), and hybrid ranking (HR) algorithms to provide insights on how to choose the ranking algorithm for different situations.

We measure the performance of our scheme from two aspects: *the detection rate*, defined as the ratio of successfully identified bad nodes, and *the false positive probability*, defined as the ratio of mis-accused innocent nodes over all innocent nodes.

We run simulations on a $400 \times 400m^2$ network with randomly generated network topology. Unless otherwise stated, we set the percentage of bad nodes to 10%, the network size to 100 sensor nodes, the per-node packet reporting interval to 3 seconds, and the length of each round to 300 seconds. Also, when a bad node decides to drop packet in a round, it drops 30% of the packets. All the results are

measured and averaged from simulation over 50 random networks.

Attack Model: We assume smart attackers who perform traffic analysis beforehand and selectively compromise non-leaf nodes, because compromised non-leaf nodes can attack the system more effectively than compromised leaf nodes. Compromised nodes might treat packets generated by themselves and those by other nodes differently. For their own packets, a compromised node may (1) drop the packets at each round, (2) drop the packets in some randomly rounds, or (3) do not drop the packets all the time. For other nodes' packets that it is supposed to forward, a compromised node may (1) drop the packets in each round, or (2) drop the packets in some randomly rounds. Consider the combination of dropping behaviors in the above two categories, we obtain six attack models in total, namely, attack models 1-1, 1-2, 2-1, 2-2, 3-1 and 3-2, where the first index represents the dropping behavior towards the packets of the bad node itself and the second index represents the dropping behavior towards others' packets. For example, attack model $1-2$ means, own packets are dropped at each round, and the packets of others are dropped at some selected rounds. This is the easiest to identify because this type of attacks will result in the case of $+-\{-\}^*$, from which the bad node can be immediately identified using our node categorization algorithm. On the other hand, attack model $3-2$ means own packets are not dropped but packets from others are dropped at some selected rounds, and experimental results demonstrate that this attack model renders intruder identification to be the hardest.

To simulate the attack behaviors in ns-2 simulator, we mimic the attacking behavior by asking each compromised node to drop packets based on a particular attacking model described above. The compromised nodes are randomly selected beforehand. After initializing the simulator, each node in the network has a flag indicating whether it is a compromised node or not. If a node is a compromised node, it will mimic the attacking behavior by dropping packets, otherwise, it honestly forwards or sends the packets.

5.4.2 Simulation Results

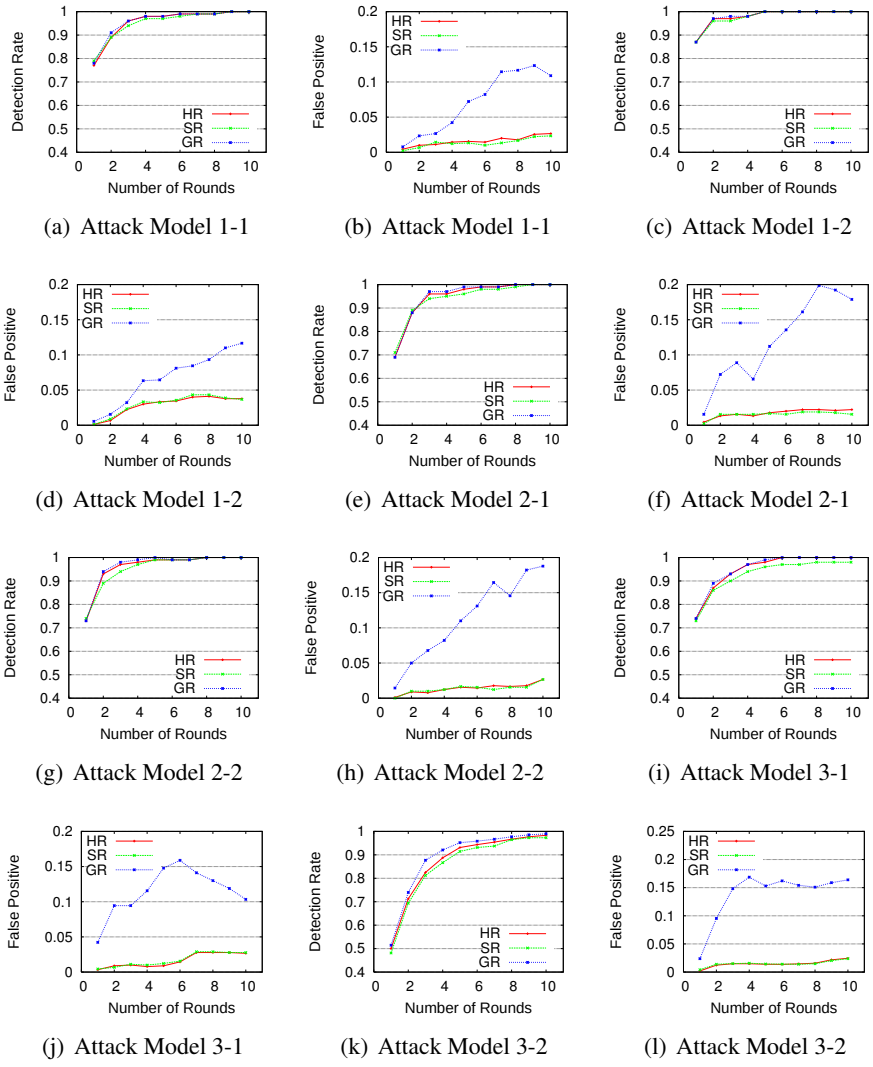
We first report the simulation results when there is no node collusion and then the results when there is collusion.

5.4.2.1 Evaluation of Ranking Algorithms

Fig 5.5 shows the detection rate and false positive probability of our scheme under different attack models. From the figure, we can see that the stepwise ranking (SR) algorithm provides a bit lower detection rate than the other two ranking algorithms in the first several rounds, but after 8 rounds, the three ranking algorithms achieve almost the same detection rate. In terms of false positive probability, the global ranking (GR) algorithm introduces much higher false positive probability than the other two, while the other two algorithms result in almost the same number of false positives. This is because the global ranking (GR) algorithm identifies bad nodes only based on the number that a node is suspected. Therefore, if an innocent node does not have many choices to select its parents in different rounds, or many of its possible parent nodes are actually compromised, the times that this innocent node is suspected will be large. On the contrary, the hybrid ranking (HR) and the stepwise ranking (SR) algorithms do not select a node which is suspected many times when that node has always been suspected together with some already-identified bad nodes, which results in less number of false accusations. Consider both the metrics, it is determined that the hybrid ranking is the best ranking algorithm among the three for its high detection rate and low false positive.

5.4.2.2 Impact of the Number of Rounds

We study the number of rounds needed to collect information such that a stable and high detection rate as well as a low false positive probability is achieved. We use the hybrid ranking (HR) algorithm here and first plot the detection rate under the six attack models in each round in Fig. 5.6. From the figure, we can see almost all bad nodes can be identified after 8 rounds regardless of the attack model. Among them, under attack model 1-2, the bad nodes will be detected quickly after 5 rounds. This is because a bad node does not drop packets from its downstream nodes at some intervals, which results in the $+-\{-\}^*$ case and the bad nodes can be identified immediately according to our proposed rule. On the contrary, under attack model 3-2, more rounds are needed to achieve a higher detection rate. In this case, bad nodes are sly and do not drop their self-generated packets. Consequently, they are only categorized as suspiciously bad nodes. More rounds are needed before they are eventually identified via a ranking algorithm.



(a) Attack Model 1-1 (b) Attack Model 1-1 (c) Attack Model 1-2

(d) Attack Model 1-2 (e) Attack Model 2-1 (f) Attack Model 2-1

(g) Attack Model 2-2 (h) Attack Model 2-2 (i) Attack Model 3-1

(j) Attack Model 3-1 (k) Attack Model 3-2 (l) Attack Model 3-2

Figure 5.5 Comparing Ranking Strategy under Various Attack Models

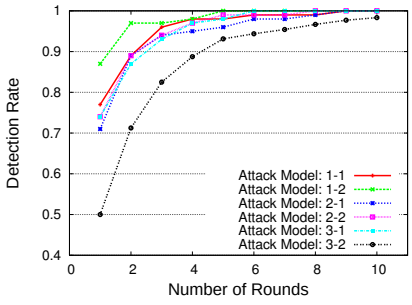


Figure 5.6 Number of Rounds vs. Detection Rate

Since the attack model 3-2 is the most difficult one, we study the standard deviations of the detection rate and the false positive probability under this attack model. The data used to compute the standard deviations are obtained from the simulations run over 50 random network topologies. The simulation results are shown in Fig. 5.7. As we can see, the standard deviation of detection rate becomes smaller and smaller as the number of rounds increases. It becomes stable after 8 rounds at about 0.125. The standard deviation of the false positive probability is higher than that of detection rate, but it is still as low as 0.15.

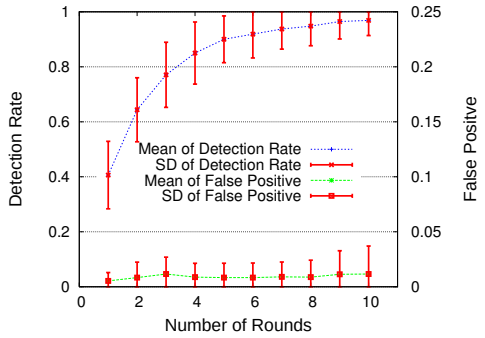


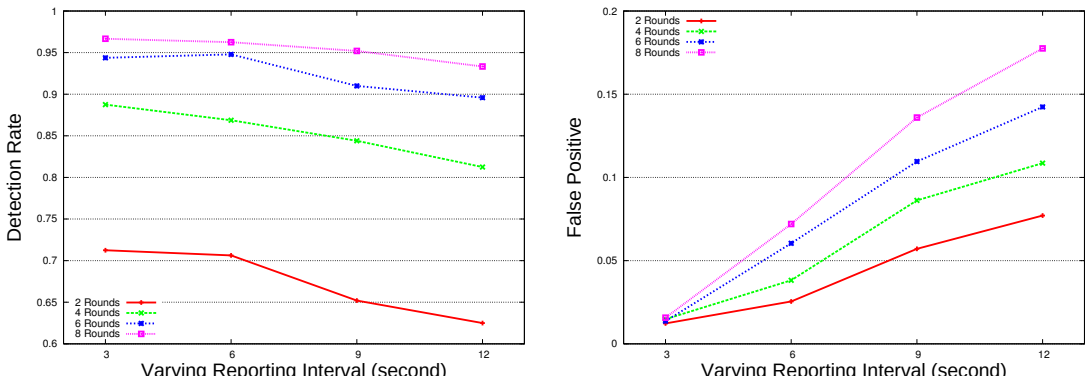
Figure 5.7 Mean and Standard Deviation for the HR Method

From the previous experimental study, attack model 3-2 is the most hidden attack, which renders great challenges to our proposed detection scheme. Also, the hybrid ranking algorithm gives the most effective detection performance towards all attack models. Therefore, in the following experiments, we study various impacts of system parameters based on attack model 3-2 with hybrid ranking algorithm.

5.4.2.3 Impact of Reporting Interval

Given a fixed time length of a round, the longer the report interval, the fewer packets are sent out. In the proposed scheme, several heuristics are based on the dropping ratio of the suspected nodes. For example, when a bad node blindly drops the forwarding packets, it drops the packets from all its downstream nodes randomly and hence the percentages of its downstream nodes' packets it drops should be similar. However, when the sample space is small, the variance of the dropping rate could be large, resulting in large false positive probability. As shown in Fig. 5.8(b), the false positive probability goes up when the reporting interval increases. In addition, the detection rate decreases when the number of rounds is small. This is because fewer packets are sent out. However, as the number of detection

rounds increases, the detection rate will approach 100% regardless of reporting interval.

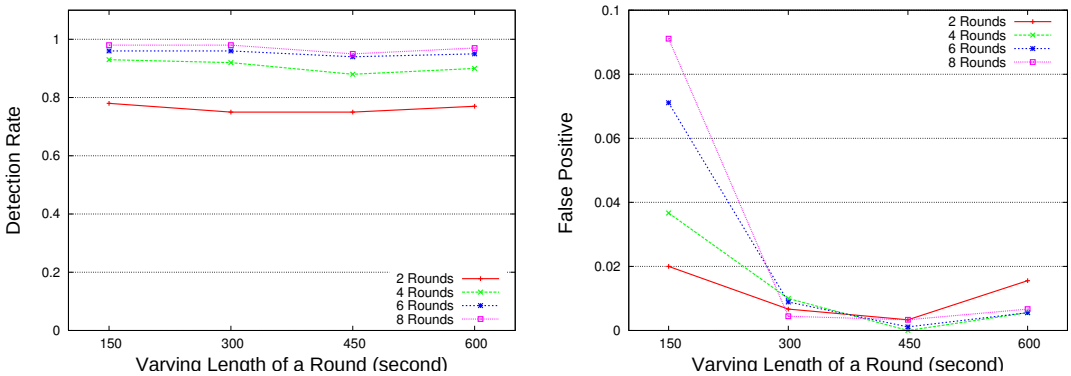


(a) Detection Rate (b) False Positive

Figure 5.8 Impact of Reporting Frequency

5.4.2.4 Impact of Round Length

Considering the delay for transmitting a packet from a source node to the sink, the round duration affects the number of packets received at the sink in each round, which in turn affects the detection performance. Fig. 5.9 shows the relationship between round duration and the detection performance. It can be seen that round duration mainly affects the false positive probability. As shown by Fig. 5.9(b), when the length is 150 seconds, the false positive probability becomes high though the detection rate is similar under different round length. This is because when the length of a round is small, there are not enough packets being generated and sent to the sink and the number of packets sent by different downstream nodes may not be dropped at the similar level. For example, when a bad node drops its forwarding packets, it is supposed to randomly drop the packets from all its downstream nodes. If the number of packets is small, it may drop more packets from some downstream node than others. In this situation, statistical analysis is not accurate and it causes relatively high false positive. As for detection rate, it is not sensitive to round length. Experimentally, we find a round length of about 300 seconds is the best.

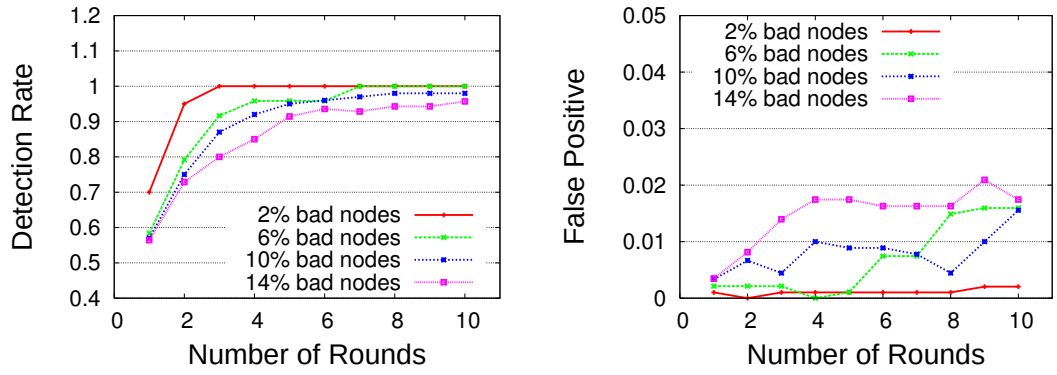


(a) Detection Rate (b) False Positive

Figure 5.9 Impact of Round Length

5.4.2.5 Impact of Percentage of Bad Nodes

Fig. 5.10 shows the detection performance as the percentage of bad nodes varies. Generally, the less the number of bad nodes is, the easier it is to identify these nodes. However, after a multiple rounds of identification, the detection rates under different percentage of bad nodes become similar, and all of them achieve very high detection rate.

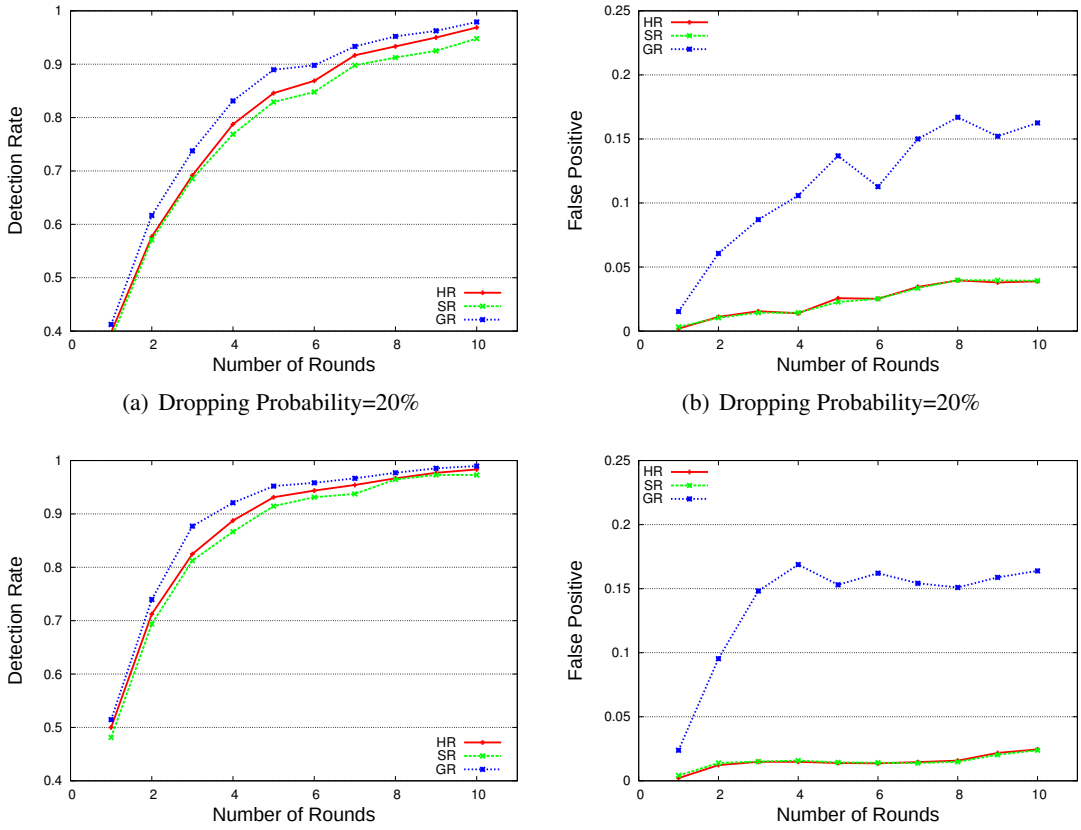


(a) Detection Rate (b) False Positive

Figure 5.10 Impact of Percentage of Bad Nodes

5.4.2.6 Impact of Dropping Probability

Fig. 5.11 shows the performance sensitivity to bad nodes' dropping percentage (i.e., the percentage of packets that will be dropped if a bad node decides to drop packets in a round). We vary the dropping probability between 20% and 80%. From Fig. 5.11, we can see the all the three ranking algorithms have a similar sensitivity to the dropping probability. In addition, under a high dropping probability, all the three algorithms achieve a higher detection rate in the early rounds, which means they can detect bad nodes quicker, and can achieve a lower false positive generally. This is because frequent misbehaviors can quickly distinguish bad nodes from innocent nodes.



(a) Dropping Probability=20%

(b) Dropping Probability=20%

(c) Dropping Probability=80%

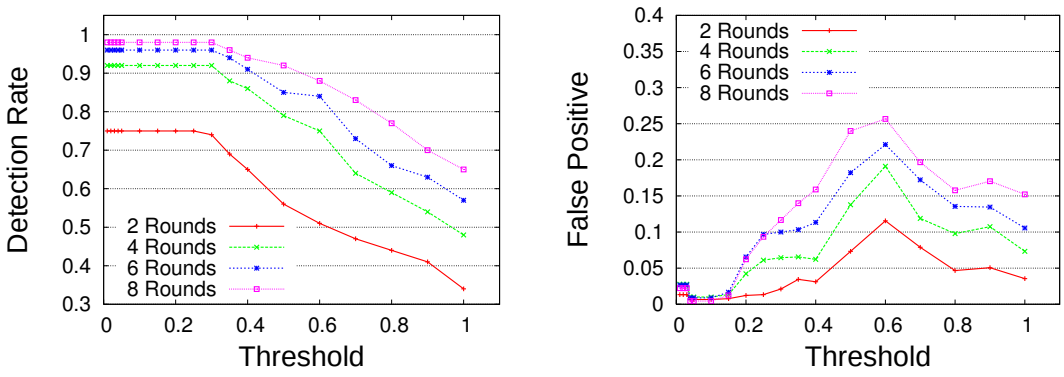
(d) Dropping Probability=80%

Figure 5.11 Comparing Ranking Strategy under Various Dropping Probability

5.4.2.7 Impact of Thresholds

(1)Threshold for Differentiating "+" Nodes and "-" Nodes. In order to make our scheme to tolerate natural packet loss, we use a threshold θ when marking each node with "+" or "-". Fig. 5.12 shows the impact of this threshold on the detection performance. As depicted in Fig. 5.12(a), the larger is the threshold, the lower is the detection rate. This is because, fewer nodes will be marked as "-" as the threshold increases; hence, a part of bad nodes will escape from being detected.

As shown in Fig. 5.12(b), when the threshold increases, the false positive probability increases first and then decreases after the threshold reaches a certain value (turning point). Hence, we select the threshold to be 0.1, with which high detection rate and low false positive can be achieved simultaneously, as shown in Fig. 5.12.

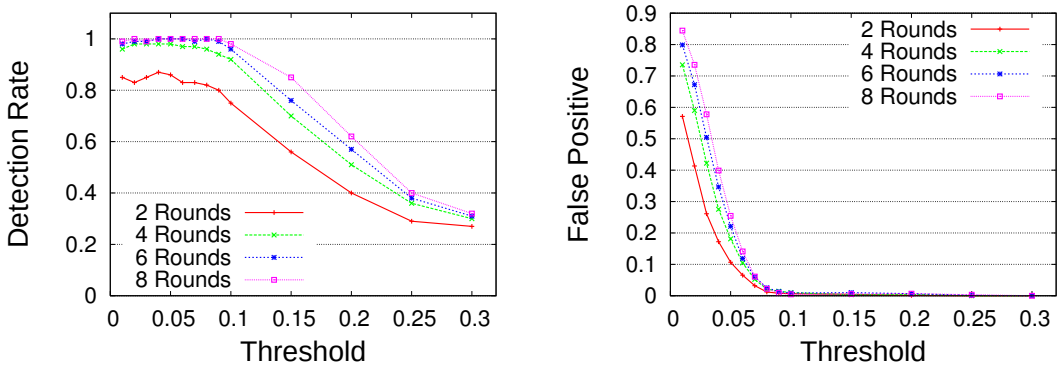


(a) Detection Rate (b) False Positive

Figure 5.12 Threshold for Differentiating "+" Nodes and "-" Nodes

(2)Threshold for Identifying Nodes with Dropping Rates. Considering that incidental collisions may cause two nodes to have different dropping rates, we use a threshold to differentiate the case that two nodes really have different dropping rates with the case there are incidental collisions. Fig. 5.13 shows the impact of this threshold on the detection rate and the false positive. We can see that, the larger the threshold, the lower the detection rate and the false positive probability. This is because the difference in dropping rates between two nodes is an important parameter for our ranking algorithms to differentiate the behaviors between parent-child nodes. If the threshold is too large, our algorithms cannot find the abnormal behaviors that are solely reflected on the dropping rate difference. If the threshold is too small,

the false positive probability will be increased, which is shown in Fig. 5.13(b). Based on our simulation, we found out threshold 0.1 can render a high detection rate and low false positive probability.



(a) Detection Rate (b) False Positive

Figure 5.13 Threshold for Identifying Nodes with Different Dropping Rates

5.4.2.8 Impact of Node Collusion

In the collusion attack model, shown as Fig. 5.3, three possible collusion cases can be generalized as follows.

Case 1: Node A, B, C and D are all compromised nodes: every node behaves normally without dropping their own packets and forwarding packets. In this attack model, these compromised nodes collude to protect themselves.

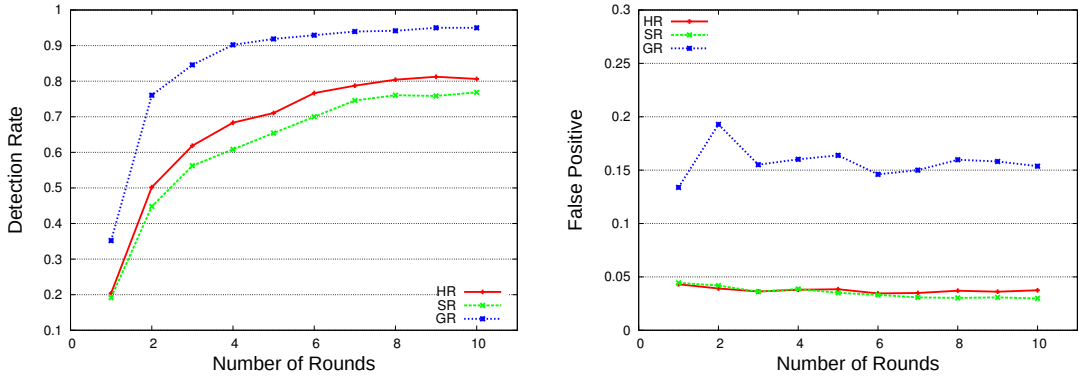
Case 2: Node A is a good node and more than one of its child nodes are compromised: the compromised nodes drop their own packets and/or packets from their children. In this case, these compromised nodes collude to frame the parent node A.

Case 3: Node A is a good node but node B and E are not. Both bad nodes B and E drop packets of their own and/or from their downstream nodes. In this case, compromised node B and node E collude to frame node A.

We randomly generate the above collusion scenarios and conduct a set of simulations to study the impact of the collusion based on *suspicious tuple* and the rule discussed in section III.D.

As shown by Fig. 5.14, under collusion attacks, the global ranking still has the highest detection

rate and the largest false positive. The performance of the stepwise ranking and the hybrid ranking are similar to each other. And after we run our proposed scheme for 10 rounds, the detection rate and the false positive probability tend to be stable. The false positive probability of the global ranking algorithm has a noticeable increase from round 1 to round 2, then it goes down and becomes stable in Fig. 5.14(b). This is because, the information about suspicious nodes got from round 1 and round 2 is very limited, the difference between the suspected times of bad nodes and those of innocent nodes is not big enough, which causes the increase of false positive when the global ranking is adopted. However, after more rounds, the accumulative suspected times of bad nodes become larger and larger, and the accumulative suspected times of innocent nodes increase much slower than those of bad nodes. Note that, each node will randomly choose its parents in a round based on the mutual agreement with the sink, an innocent node may choose a parent which is bad at a round, and choose innocent parents at some other rounds; hence, the accumulative times of being suspected for innocent nodes will be generally less than those for bad nodes. In summary, the trend observed in the collusion scenarios is similar to that in the non-collusion scenarios.

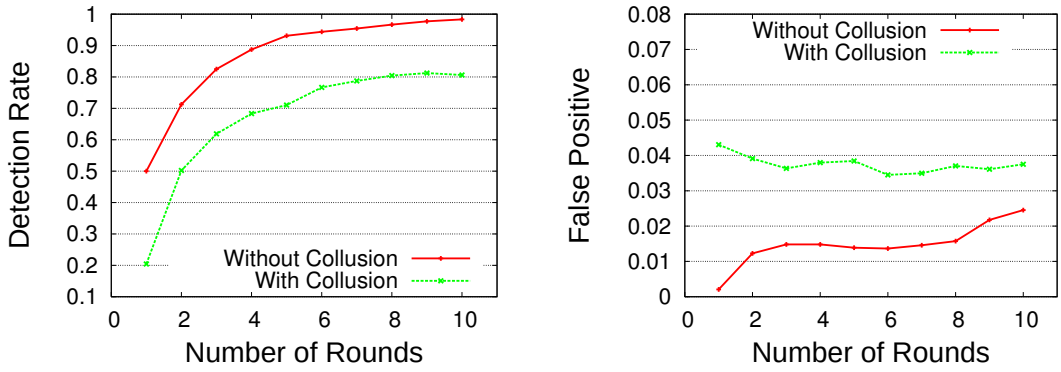


(a) Detection Rate (b) False Positive

Figure 5.14 Comparison of Ranking Strategy with Collusion

We also compare the detection rates under the collusion scenarios and the non-collusion scenarios and show the results in Fig. 5.15. The hybrid ranking algorithm is used. We can see that, the detection performance degrades under the collusion scenarios. But the detection rate is still as high as 80% and a low false positive probability is maintained. The reason for lower detection rate can be explained

as follows: When there are collusions, multiple colluding bad nodes and one or more innocent nodes are put into a single tuple. However, if there is no collusion, generally there is only one bad node in a tuple. Hence, the bad nodes' overall times of being suspected is reduced when there is collusion, which degrades the efficiency of identifying bad nodes. In fact, if a set of bad nodes collude together most of time, only one of these nodes can be identified. To deal with this issue, after one bad node is identified, it should be removed from the system and thus it cannot protect other bad nodes from being identified through continuously colluding with them.



(a) Detection Rate (b) False Positive

Figure 5.15 Comparison between collusion and non-collusion

5.4.3 Performance Comparison

To identify packet modifiers and droppers, it has been proposed to add nested MACs to address this problem [79, 69]. In this section, we compare our proposed scheme with the PNM scheme [79] regarding detection performance and communication overhead.

5.4.3.1 Detection Rate and False Positive

Fig. 5.16 shows the overall detection performance of the PNM scheme [79] based on the same network topology and configuration of compromised nodes as in Fig. 5.7. The highest detection rate it can achieve is about 75%, and the lowest false positive is around 10%. However, as shown by Fig. 5.7, our proposed scheme can achieve 96% detection rate meanwhile maintain the false positive as low as

1%. This is because the nested MAC approach can only identify the problematic links, i.e., it will catch a compromised node as well as its one-hop neighbor node. For a path containing several compromised nodes, it can only identify the first problematic link near the sink.

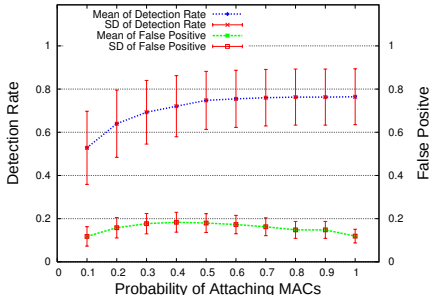
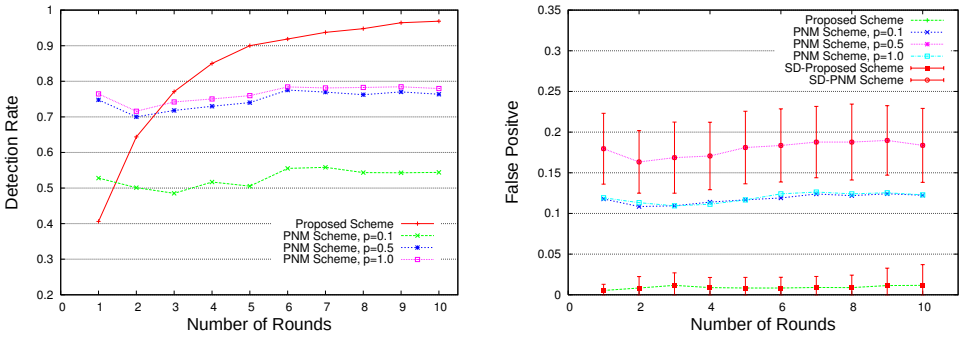


Figure 5.16 Performance of the PNM scheme

Fig. 5.17(a) compares the detection performance between our proposed scheme and the PNM scheme. We can see that our proposed scheme achieves better detection performance after three rounds. Also, the false positive of our proposed scheme is much lower than the PNM scheme, as shown by Fig. 5.17(b). This is because our proposed scheme utilizes the observed wide variety of node behaviors and makes a good heuristic decision on telling which nodes are compromised, while the PNM scheme is only capable to identify the problematic links, and cannot tell which node is the compromised node within a problematic link.



(a) Detection Rate (b) False Positive

Figure 5.17 Comparison between collusion and non-collusion

5.4.3.2 Communication Overhead

We also compare the communication overhead between the PNM scheme and our proposed scheme. In our proposed scheme, only fields R_u and $pad_{u,1}$ are the extra bits, because other fields are necessary even when security issues are not considered (for example, P_u and u are the destination and source of the packet, C_p is the sequence number used by the sink to find out if some packets are lost, etc.). The size of R_u is determined by N_p, which is the maximum number of parents that each sensor node should record at the tree establishment phase. As to be seen later, parameter N_p determines the range that the tree structure can be reshaped; specially, if the total number of nodes in the tree is n, potentially there are N_p^n different tree topologies that can be used to test the behaviors of nodes. Therefore, its value should be reasonably large. It is set to be 8 in our simulations. The size of $pad_{u,1}$ is determined by both N_p and h, which is the height of the routing tree. Though the structure of the routing tree changes dynamically from round to round, the level of each node in the tree remains the same. This is because each node only records nodes which are one hop closer to the sink as its candidate parent nodes, and each node dynamically changes its parent node from the recorded candidates in the tree reshaping phase. Therefore, the communication overhead per node is fixed in our proposed scheme.

In the PNM scheme, the extra communication overhead is the marks added for tracing back the problematic links. In our simulation, we adopt the RC5 primitives to compute the MAC and the block size is configured to 64 bits. Fig. 5.18 shows the comparison of extra communication overhead per node. As we can see, the per node communication overhead of the PNM scheme increases as the probability of attaching marks increases. On the other hand, the per node communication overhead of our proposed scheme increases as the number of rounds increases. The proposed scheme outperforms the PNM scheme regarding communication overhead when the probability of attaching marks is greater than or equal to 0.5. Meanwhile, the detection performance of our proposed scheme also outperforms the PNM scheme, which is shown by Fig. 5.17.

5.4.4 Implementation of The Proposed Scheme

We implemented our proposed scheme on TelosB motes, which are widely used resource-constrained sensor motes produced by CrossBow [44]. Each TelosB mote has a CPU running at 4MHz, a RAM of

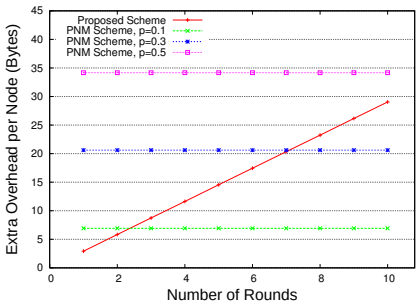


Figure 5.18 Comparison of Communication Overhead

10KB size, and a flash storage of 1MB size. RC5 encryption primitives are used in our implementation. The block size is set to 64 bits. The code running on TelosB consumes 624 bytes of RAM and $15,216$ bytes of ROM. The encryption time on sensor motes depends on the length of encrypted data. In our proposed scheme, the part that needs encryption is $\{R_u, u, C_p \text{ MOD } N_s, D, pad_{u,0}\}_{K_u}$. Its length is decided by several parameters discussed in Sec. 5.3.1.2. We report the computation overhead in Table 5.1 by varying the length of data. Other parameters, namely, N_p, N_s and L_{id} are set to 8, 1024, and 10 bits respectively. From the results shown in Table 5.1, the computational overhead is quite low, and it is affordable by resource-constrained sensor networks.

Table 5.1 Computational cost for sensor to forward a packet (ms)

Data Length (Bytes)	12	20	28	36	44	52
Computational Time	120	178	237	296	354	412

5.5 Conclusion

We propose a simple yet effective scheme to identify misbehaving forwarders that drop or modify packets. Each packet is encrypted and padded so as to hide the source of the packet. The packet mark, a small number of extra bits, is added in each packet such that the sink can recover the source of the packet and then figure out the dropping rate associated with every sensor node. The routing tree structure dynamically changes in each round so that behaviors of sensor nodes can be observed in a large variety of scenarios. Finally, most of the bad nodes can be identified by our heuristic ranking algorithms on the observed node behaviors with small false positive. Extensive analysis and simulations have been conducted and verified the effectiveness of the proposed scheme in various scenarios.

CHAPTER 6. LA^3: a Lightweight Accountable and Anonymous Authentication Scheme

As more and more data has been centralized into the cloud, resource constrained devices, such as smart phones, have been pervasively used to access online data. Many schemes have been proposed to address the security and privacy concerns of data access, however they are not designed for resource constrained devices. In this chapter, we design an accountable and anonymous authentication scheme, which is tailored for resource constrained devices running in low CPU cycles.

6.1 Introduction

The access privilege of online service and data are often required to be verified, however, as more and more sensitive people-centric data has been centralized into the cloud, the privacy of innocent people could be revealed during the course of being verified. For instance, the identity of authorized people may be inferred and correlated when accessing the sensory data. On the other hand, malicious users shall not be able to leverage privacy-preserving mechanisms without being caught. Hence, it is necessary to design an accountable anonymous scheme to authenticate users.

Many accountable anonymous authentication schemes [90, 92, 93, 94, 95, 96] have been proposed for controlled access to online services. These schemes are mainly built upon group signature algorithms [100, 103, 104, 106] which provide provable anonymity and traceability. However, the computational and communication costs introduced by these schemes may be too high for resource constrained devices. As smart phones and/or tablets are pervasively used to access online services, we proposes LA^3, a lightweight accountable and anonymous authentication scheme, for such resource constrained devices. LA^3 scheme assumes three types of entities in the system: a service provider (called verifier) that needs to verify whether a client has the privilege to access its service, a group of clients (called

provers) that need to prove their access privileges, and a trusted authority responsible for choosing system parameters and initializing the verifier and provers. Following the protocol of LA^3, a prover and the verifier can interact with each other in an authentication transaction. The authentication process involves only a few multiplication and exponential operations over a multiplicative cyclic group and over a finite field, in addition to a small number of message exchanges. The prover can keep anonymous to the verifier; but the authority is able to trace out the prover based on the authentication transcript when necessary.

The security of LA^3 relies on the hardness assumptions of the Computational Diffie-Hellman (CDH), Decisional Diffie-Hellman (DDH), q-Strong Diffie-Hellman (q-SDH), q-Decisional Diffie-Hellman Inversion (q-DDHI) and LRSW problems. Intuitively speaking, as long as the above problems are hard to solve, the LA^3 will have the following security properties: (i) *Non-frameability*. It is hard for the verifier and any coalition of provers to impersonate any innocent prover (i.e., prover not belonging to the coalition) in authentication. (ii) *Traceability*. It is hard for any coalition of provers to succeed in an authentication transaction without revealing any of their IDs to the authority. (iii) *Selfless Anonymity*. It is hard for the verifier and any coalition of provers to determine whether two authentication transactions involve the same innocent prover. Formal definitions and rigorous proofs of the above security properties are provided in the paper.

We have implemented the LA^3 scheme, and compared its performance to that of existing group signature schemes. The results show that, as operations needed in the authentication process are simpler, the computational efficiency of LA^3 scheme is much higher than that of the existing group signature schemes; particularly, it is more than 10 times faster than the group signature scheme proposed by Boneh and Shacham [103], which has wide applications.

6.2 Preliminaries

6.2.1 Notations

Let $\mathbb{Z}_p$ denote a prime finite field where p is a prime number. Let $\mathbb{G}$ be a multiplicative cyclic group of p elements, where g is a generator. Hence, the members of $\mathbb{G}$ can be represented as $\{g^0, g, g^2, \cdots, g^{p-1}\}$.

6.2.2 Assumptions: Hard Problems

Our proposed design is based on the assumptions that the following problems are hard:

Computational Diffie-Hellman (CDH) Problem [120] Given g, g^a and g^b in $\mathbb{G}$, find g^{ab}.

Decisional Diffie-Hellman (DDH) Problem [121] Given g, g^a, g^b and g^{bc} in $\mathbb{G}$, determine if $c = a$.

q-Strong Diffie-Hellman (q-SDH) Problem [104] Given g, g^x, g^{x^2}, $\cdots$, g^{x^q} in $\mathbb{G}$ where x, $q \in \mathbb{Z}_p$, find $(c, g^{1/(c+x)})$ where $c \in \mathbb{Z}_p$.

q-Decisional Diffie-Hellman Inversion (q-DDHI) Problem [122] Given g, g^x, g^{x^2}, $\cdots$, g^{x^q} and $g^{1/(x+y)}$ in $\mathbb{G}$ where x, $q \in \mathbb{Z}_p$, determine if $y = 0$.

LRSW Problem [123] Given g, g^x and g^y in $\mathbb{G}$ and oracle O which on input s returns $(g', (g')^{sy}, (g')^{x+sxy})$ where $g' = g^z$ for some $z \in \mathbb{Z}_p$, compute $(b, t, b^{ty}, b^{x+txy})$ where $b \neq g^0 \in \mathbb{G}$ and t is not one of the s that has been queried.

6.2.3 Scheme Overview

We consider a system that is composed of a trusted authority, a verifier, and a set of provers. Our proposed scheme includes the following components.

- *System Initialization*, with which the trusted *authority* initializes system parameters. The primitive is formally denoted as $SystemInit() \rightarrow \mathbb{SS}$, which outputs a set $\mathbb{SS}$ of system parameters.

- *Verifier Initialization*, with which the authority initializes the *verifier* by providing parameters needed by the verifier. The primitive is formally denoted as $VerifierInit\ (\mathbb{SS}) \rightarrow \mathbb{SV}$, which takes as input the set of system parameters and outputs the set $\mathbb{SV}$ of parameters needed by the verifier.

- *Prover Initialization*, with which the authority initializes a prover by providing parameters needed by the prover. The primitive is formally denoted as $ProverInit\ (\mathbb{SS},\ u) \rightarrow (\mathbb{SP}_u, T_u)$, which takes as inputs the set of system parameters and a prover ID u. It outputs the set $\mathbb{SP}_u$ of parameters needed by prover u and the revocation token T_u of prover u. When the authority initializes a prover u (i.e., when the prover joins the system), the authority calls the primitive to obtain $\mathbb{SP}_u$ and T_u, preloads $\mathbb{SP}_u$ to the prover as its private key, and keeps T_u secret for the purpose of tracing and revocation.

- *Authentication Protocol*, with which prover u proves itself to the verifier. The protocol can be formalized as two primitives:
  - $Prove(\mathbb{SP}_u, \tilde{c}_1, \tilde{c}_2) \rightarrow (\tilde{r}_1, \tilde{r}_2)$. This primitive is used by prover u, who holds private key $\mathbb{SP}_u$, to first produce $\tilde{r}_1$ in response to the verifier's challenge $\tilde{c}_1$, and then produce $\tilde{r}_2$ in response to the verifier's challenge $\tilde{c}_2$.
  - $Verify(\mathbb{RT}, \tilde{c}_1, \tilde{r}_1, \tilde{c}_2, \tilde{r}_2) \rightarrow 1/0$. This primitive is used by the verifier. It takes as inputs revocation token set $\mathbb{RT}$ (i.e., the set of tokens of all revoked provers) and the authentication transcript $(\tilde{c}_1, \tilde{r}_1, \tilde{c}_2, \tilde{r}_2)$, and outputs 1 if the authentication succeeds or 0 otherwise.
- *Prover Tracing*, with which the authority traces the identity of a prover based on an authentication transcript. The primitive is formally denoted as $Trace(\mathbb{T}, \tilde{c}_1, \tilde{r}_1, \tilde{c}_2, \tilde{r}_2) \rightarrow u$, which takes as inputs the set $\mathbb{T} = \{T_u | \forall u\}$ of all provers' tokens and an authentication transcript $(\tilde{c}_1, \tilde{r}_1, \tilde{c}_2, \tilde{r}_2)$, and outputs the ID of the prover who generates the responses in the authentication procedure.

6.2.4 Security Definitions

Our design aims to achieve the following security properties:

6.2.4.1 Correctness

Definition The LA^3 scheme is correct if the authentication transcript generated by the verifier and a prover u that has not been revoked must be verified successfully. Formally,

$$(Prove(\mathbb{SP}_u, \tilde{c}_1, \tilde{c}_1) \rightarrow (\tilde{r}_1, \tilde{r}_2)) \Rightarrow [(Verify(\mathbb{RT}, \tilde{c}_1, \tilde{r}_1, \tilde{c}_2, \tilde{r}_2) \rightarrow 1) \vee (T_u \in \mathbb{RT})].$$

6.2.4.2 Non-frameability

Intuitively, non-frameability defines the property that, the verifier and any set of collusive provers cannot impersonate any innocent prover. Its formal definition is as follows.

Definition The LA^3 scheme is (t, q_A, ϵ) non-frameable if no adversary can win the following *Non-frameability Game* with a probability greater than ϵ in time t with less than q_A authentication queries on each prover.

Non-frameability Game The game is between an adversary and a challenger. It is composed of the following phases.

- **Phase I: Initialization.** The challenger initializes one verifier and a set of n provers.
- **Phase II: Queries and responses.** The adversary can issue the following types of queries and the challenger should respond accordingly.
  - *Corruption of the verifier:* The adversary issues a corruption query on the verifier. In response, the challenger returns the secret parameters of the verifier.
  - *Corruption of a prover u:* The adversary issues a corruption query on a prover u. The challenger responds with the secret parameters of the prover.
  - *Authentication for a prover v:* The adversary issues an authentication query on a prover v, and the adversary also provides the first challenge $\tilde{c}_1$. In response, the challenger returns a response $\tilde{r}_1$. Then the adversary provides the second challenge $\tilde{c}_2$, and the challenger returns the second response $\tilde{r}_2$.
  - *Hash:* The adversary issues a hash query, and the challenge responds with an element of $\mathbb{Z}_p$ randomly and consistently.
- **Phase III: Adversary's Response.** The adversary provides an authentication transcript $(\tilde{c}'_1, \tilde{r}'_1, \tilde{c}'_2, \tilde{r}'_2)$.

The adversary wins if the response satisfies the following conditions: (i) The authentication is successful; i.e., $Verify(\emptyset, \tilde{c}'_1, \tilde{r}'_1, \tilde{c}'_2, \tilde{r}'_2) = 1$. (ii) The authentication transaction can trace to a prover. (iii) Assuming the authentication is traced to prover w, no authentication query has been made for w with challenges $\tilde{c}'_1$, and no corruption query has been made on prover w.

6.2.4.3 Traceability

Intuitively, traceability defines the property that, authentication transcripts forged by any coalition of collusive provers must trace to one of these provers. Its formal definition is as follows.

Definition The LA^3 scheme is (t, ϵ) traceable if no adversary can win the following *Traceability Game* with a probability greater than ϵ in time t.

Traceability Game The game is between an adversary and a challenger. It is composed of the following phases.

- **Phase I: Initialization.** The challenger initializes one verifier and a set $\mathbb{P}$ of n provers.
- **Phase II: Queries and responses.** The adversary can issue two types of queries, namely, *corruption of a prover* and *authentication for a prover*, and the challenger responds accordingly as in the *Non-frameability Game*.
- **Phase III: Adversary's Response.** Given $\tilde{c}'_1$ provided by the challenger, the adversary responds with $\tilde{r}'_1$. Further given $\tilde{c}'_2$ by the challenger, the adversary responds with $\tilde{r}'_2$.

The adversary wins if the response satisfies the following conditions: (i) $Verify(\emptyset, \tilde{c}'_1, \tilde{r}'_1, \tilde{c}'_2, \tilde{r}'_2) = 1$. (ii) The authentication transcript cannot trace to any prover in $\mathbb{P}$; i.e., there is no $u \in \mathbb{P}$ such that $Trace(\mathbb{T}, \tilde{c}'_1, \tilde{r}'_1, \tilde{c}'_2, \tilde{r}'_2) \rightarrow u$.

6.2.4.4 Selfless Anonymity

Intuitively, selfless anonymity defines the property that, the verifier and any coalition of collusive provers cannot determine whether two authentication transactions were performed by the same innocent prover or not. Its formal definition is as follows.

Definition The LA^3 scheme is (t, q_A, ϵ) selfless anonymous if no adversary can win the following *Selfless Anonymity Game* with a probability greater than ϵ with less than q_A authentication queries on each prover in time t.

Selfless Anonymity Game The game is between an adversary and a challenger. It is composed of the following phases.

- **Phase I: Initialization.** A verifier and a set $\mathbb{P}$ of n provers are initialized.
- **Phase II: Pre-Challenge Queries and Responses.** The adversary can issue three types of queries, namely, *corruption of the verifier*, *corruption of a prover* and *authentication for a prover*, which are responded by the challenger as in the *Non-frameability Game*.

- **Phase III: Challenge.** The adversary selects two provers u_0 and u_1 from $\mathbb{P}$ that have not been compromised. The challenger randomly picks i from 0 or 1, and presents a response generated by u_i.
- **Phase IV: Post-Challenge Queries and Responses.** The same as Phase II except that compromise queries cannot be made on provers u_0 or u_1.
- **Phase V: Adversary's Response.** The adversary returns $i' \in \{0, 1\}$.

The adversary wins if $i' = i$.

6.3 Our Construction

The design of the proposed scheme is based on the **intuition** as follows: The authority randomly picks numbers k_1, d and l from a finite field, and a polynomial function $C(x)$ over the finite field. Each prover in the system is associated with a unique set of randomly selected numbers and polynomial functions including (i) ID u, (ii) numbers λ_u, s_u and e_u, and (iii) polynomial function $B_u(x)$; based on the numbers, prover u is also associated with a polynomial function $F_u(x)$ such that

$$\lambda_u(k_1 + k_2 s_u + 1)C(x) + B_u(x)d + e_u l + F_u(x) = 0, \text{ where } k_2 = 1 - k_1. \tag{6.1}$$

The authority initializes each prover u by preloading to it a set of parameters (similar to a private key for signing) that embed λ_u, s_u, $B_u(x)$, e_u and $F_u(x)$, and initializes the verifier by preloading to it a set of parameters (similar to a public key for verification) that embed k_1, d, l and $C(x)$. Note that, some of these numbers and functions are not preloaded in plaintext, but in *encrypted* forms in order to achieve the afore-defined security properties. Particularly, each prover u is preloaded with

$$\hat{k}_{u,0} = g^{\lambda_u}, \hat{k}_{u,1} = \hat{k}_{u,0}^{k_1}, s_u, B_u(x), \hat{e}_u = g^{e_u}, \text{ and } \hat{F}_u(x) = g^{F_u(x)},$$

and the verifier is preloaded with

$$k_1, k_2, l, C(x), \text{ and } \hat{d} = g^d,$$

where g is the generator of group $\mathbb{G}$. In every authentication transaction, the verifier provides some challenges that are never reused. The prover generates responses based on the challenges and the aforementioned parameters preloaded by the authority, to prove its knowledge of the parameters without

exposing the knowledge itself. The verifier can determine if the prover has the required knowledge through some test derived from Eq. (6.1). The scheme is presented in detail as follows:

6.3.1 System Initialization

The trusted authority initializes the system by selecting c_0, c_1, d, k_1 and l randomly from $\mathbb{Z}_p$, and let $k_2 = 1 - k_1$. Formally, the procedure of system initialization can be specified as $SystemInit() \rightarrow \mathbb{SS} = \{C(x) = c_0 + c_1 x, d, k_1, k_2, l\}$.

6.3.2 Verifier Initialization

The authority provides the following secrets to the verifier: k_1, k_2, l, $C(x)$, and $\hat{d} = g^d$. The procedure of verifier initialization can be formally specified as

$$VerifierInit(\mathbb{SS}) \rightarrow \mathbb{SV} = \{k_1, k_2, l, C(x), \hat{d}\}.$$

6.3.3 Prover Initialization

Each prover is given a unique ID u and is provided with the following secrets:

- $\hat{k}_{u,0} = g^{\lambda_u}$ and $\hat{k}_{u,1} = \hat{k}_{u,0}^{k_1}$ where λ_u is randomly picked from $\mathbb{Z}_p \setminus \{0\}$;
- s_u which is randomly picked from $\mathbb{Z}_p$;
- $B_u(x) = x + b_u$ where b_u is randomly picked from $\mathbb{Z}_p$;
- $\hat{e}_u = g^{e_u}$ where e_u is randomly picked from $\mathbb{Z}_p$;
- $\hat{F}_u(x) = \hat{f}_{u,1}^x \hat{f}_{u,0}$, where $\hat{f}_{u,1} = g^{f_{u,1}}$, $\hat{f}_{u,0} = g^{f_{u,0}}$ and polynomial $F_u(x) = f_{u,1}x + f_{u,0}$ is constructed to satisfy Eq. (6.1).

Formally, the procedure of initializing prover u can be specified as

$$ProverInit(\mathbb{SS}, u) \rightarrow (\mathbb{SP}_u, T_u),$$

where $\mathbb{SP}_u = \{\hat{k}_{u,0}, \hat{k}_{u,1}, s_u, B_u(x), \hat{e}_u, \hat{F}_u(x)\}$ and $T_u = \{\lambda_u, s_u, b_u\}$. Note that T_u is not provided to prover u but is kept by the authority.

6.3.4 Authentication Protocol

The authentication protocol runs as follows.

1. When a prover u wants to start an authentication transaction, she first proposes a nonce $r'_1 \in \mathbb{Z}_p$ and sends it to the verifier.

2. The verifier randomly picks another nonce $r'_2 \in \mathbb{Z}_p$ and and sends it to the prover.

3. Then, the following sub-steps are conducted:

   (a) Prover u computes $r = h(r'_1, r'_2) \in \mathbb{Z}_p$, where $h()$ is a hash function.

   (b) Prover u randomly selects α, β and ξ from $\mathbb{Z}_p$. She computes the following values and sends them to the verifier:

$$a_{u,1,r} = 2\alpha + \xi - 1, a_{u,2,r} = \alpha(s_u + 1) + \xi - 1, \hat{k}_{u,0,r} = \hat{k}_{u,0}^{\beta}, \hat{k}_{u,1,r} = \hat{k}_{u,1}^{\beta},$$

$$B_{u,r} = \alpha\beta B_u(r), \hat{e}_{u,r} = \hat{e}_u^{\alpha\beta}, \hat{F}_{u,r} = \hat{F}_u(r)^{\alpha\beta}.$$

4. The verifier tests if

$$\hat{k}_{u,1,r} = \hat{k}_{u,0,r}^{k_1}. \tag{6.2}$$

   If so,

$$\hat{k}_{u,2,r} = \hat{k}_{u,0,r} / \hat{k}_{u,1,r}. \tag{6.3}$$

   Then, it randomly picks r'_3 from $\mathbb{Z}_p \setminus \{0\}$, and sends to prover u

$$\hat{k}_{u,4,r} = (\hat{k}_{u,1,r} \hat{k}_{u,2,r})^{1/r'_3}. \tag{6.4}$$

5. Prover u returns to the verifier

$$\hat{k}_{u,5,r} = \hat{k}_{u,4,r}^{\xi}. \tag{6.5}$$

6. The verifier tests if

$$(\hat{k}_{u,0,r} \hat{k}_{u,1,r}^{a_{u,1,r}} \hat{k}_{u,2,r}^{a_{u,2,r}} \hat{k}_{u,5,r}^{-r'_3})^{C(r)} \hat{d}^{B_{u,r}} \hat{e}_{u,r}^{l} \hat{F}_{u,r} = g^0. \tag{6.6}$$

   If the above test has been successful, the verifier will check if the prover has been revoked, the procedure of which is to be detailed in Section 6.3.6. If the prover is not revoked, the verification succeeds.

In the authentication protocol, the behaviors of the prover and the verifier can be formally expressed as

$$Prove(\mathbb{S}_u, \tilde{c}_1, \tilde{c}_2) \rightarrow (\tilde{r}_1, \tilde{r}_2),$$

and

$$Verify(\mathbb{RT}, \tilde{c}_1, \tilde{r}_1, \tilde{c}_2, \tilde{r}_2) \rightarrow 1/0,$$

where

$$\tilde{c}_1 = \{r'_1\}, \tilde{r}_1 = \{r'_2, r, a_{u,1,r}, a_{u,2,r}, \hat{k}_{u,0,r}, \hat{k}_{u,1,r}, B_{u,r}, \hat{e}_{u,r}, \hat{F}_{u,r}\},$$

$$\tilde{c}_2 = \{\hat{k}_{u,4,r}\}, \tilde{r}_2 = \{\hat{k}_{u,5,r}\}.$$

6.3.5 Tracing Algorithm

The authority keeps $T_v = \{s_v, \lambda_v, b_v\}$ for each prover v. Given the transcript $\hat{k}_{u,0,r}$, $a_{u,1,r}$, $a_{u,2,r}$, and $B_{u,r}$, generated during an authentication transaction, the authority traces the prover as follows. For each prover v,

$$\alpha = (a_{u,2,r} - a_{u,1,r})/(s_v - 1). \tag{6.7}$$

Then, if

$$\hat{k}_{u,0,r}^{\alpha B_v(r)/(\lambda_v B_{u,r})} = g, \tag{6.8}$$

the prover involved in the authentication transaction is traced to prover v.

Formally, the tracing procedure can be expressed as $Trace(\mathbb{T}, \tilde{c}_1, \tilde{r}_1, \tilde{c}_2, \tilde{r}_2) \rightarrow v$, where $\mathbb{T} = \{T_v | \forall \text{ prover } v\}$.

6.3.6 Revocation

For each revoked prover v, revocation token T_v is provided to the verifier. After a prover has passed the test expressed in equation (6.6), formula (6.7) is computed and then equation (6.8) is tested to find if the prover in the verification procedure has been revoked.

6.4 Security Proofs

6.4.1 Correctness

Theorem 6.4.1 *The LA^3 scheme is correct.*

Proof Consider the authentication transcript

$$r, \hat{k}_{u,0,r}, \hat{k}_{u,1,r}, \hat{k}_{u,2,r}, a_{u,1,r}, a_{u,2,r}, B_{u,r}, r'_3, \hat{k}_{u,5,r}, \hat{e}_{u,r}, \hat{F}_{u,r}$$

generated by the verifier and a non-revoked prover u. It holds that

$$\begin{aligned} & (\hat{k}_{u,0,r}\hat{k}_{u,1,r}^{a_{u,1,r}}\hat{k}_{u,2,r}^{a_{u,2,r}}\hat{k}_{u,5,r}^{-r'_3})^{C(r)} \qquad (6.9) \\ = \; & (g^{\lambda_u\beta})^{[1+k_1(2\alpha+\xi-1)+k_2(\alpha s_u+\alpha+\xi-1)-(k_1+k_2)\xi]C(r)} \\ = \; & g^{\lambda_u\alpha\beta(1+k_1+k_2s_u)C(r)}, \end{aligned}$$

$$\hat{d}^{B_{u,r}} = g^{d\alpha\beta B_u(r)}, \hat{e}_{u,r}^{l} = g^{\alpha\beta e_u l}, \hat{F}_{u,r} = g^{\alpha\beta F_u(r)}. \qquad (6.10)$$

Hence, the product of the above equations is

$$(g^{\alpha\beta})^{\lambda_u(k_1+k_2s_u+1)C(r)+B_u(r)d+e_u l+F_u(r)} = g^0 \qquad (6.11)$$

according to equation (6.1). Therefore, prover u must be verified successfully.

6.4.2 Non-frameability

Theorem 6.4.2 *If the q-SDH problem is (t, ϵ)-hard, i.e., it cannot be solved with a probability greater than ϵ in time t, then the LA^3 scheme is $(t', q+1, n\epsilon)$ non-frameable where $t' = \Theta(1) \cdot t$ and n is the total number of provers.*

Proof See Appendix A.1

6.4.3 Traceability

Lemma 6.4.1 *If the DDH problem is (t, ϵ)-hard, the probability for the adversary to find out $\hat{d}$ in time $\Theta(1) \cdot t$ through the traceability game is not greater than ϵ.*

Proof See Appendix A.2.

Definition Extended LRSW Problem: given g, g^k, g^{c_1}, $g^{c_0} \in \mathbb{G}$ (k, c_1, $c_0 \in \mathbb{Z}_p$), and oracle O which on input s returns g', $(g')^k$, $(g')^{[k\cdot(s-1)+2]c_1}$ and $(g')^{[k\cdot(s-1)+2]c_0}$, where $g' = g^z$ for some $z \in \mathbb{Z}_p$, to compute g'', t, r, $(g'')^k$ and $(g'')^{[k\cdot(t-1)+2](c_1 r+c_0)}$, where $g'' \neq g^0 \in \mathbb{G}$ and t is different from any s that has been queried.

Lemma 6.4.2 *If the LRSW problem is (t, ϵ)-hard, then the extended LRSW problem is (t', ϵ)-hard where $t' = \Theta(1) \cdot t$.*

Proof See Appendix A.3.

Theorem 6.4.3 *If the CDH problem, the LRSW problem and the DDH problem are (t, ϵ)-hard, the LA^3 scheme is $(t', 2\epsilon)$ traceable where $t' = \Theta(1) \cdot t$.*

Proof See Appendix A.4.

6.4.4 Selfless Anonymity

Theorem 6.4.4 *If the q-DDHI problem is $(t, 0.5 + \epsilon)$-hard, then the LA^3 scheme is $(t', m, 0.5 + 2n^2\epsilon)$ selflessly anonymous where $t' = \Theta(1) \cdot t$, $m = \lfloor q/2 \rfloor + 1$ and n is the total number of provers.*

Proof See Appendix A.5.

6.5 Implementation and Evaluation

6.5.1 Implementation

We have implemented the LA^3 scheme based on an elliptic cyclic group. Specifically, we use the elliptic cryptographic primitives contributed by FlexiProvider [124]. When using the ECC libraries, we adopt the recommended elliptic curve parameters specified by secp160r1 [125].

We compare the performance of LA^3 with that of the group signature scheme proposed by Boneh and Shacham [103], denoted as the BS scheme in this following. We implemented BS based on java paring-based cryptographic (jPBC) library to enable the operations on the bilinear maps [126, 127]. We adopted the type A curve in our implementation, which is the fastest curve to compute the pairing operations based on the benchmark results from jPBC [128].

6.5.2 Performance Comparison with the BS Scheme [103]

We measured the performance of LA^3 and BS on a laptop computer with 1.83 GHz Genuine Intel (R) processor and 3 GB of RAM. The experimental results reported below are the averaged results of over 100 experimental runs.

Computational efficiency BS spends about 1611 milliseconds to generate a group signature and about 1807 milliseconds to verify a group signature. In contrast, LA^3 spends only about 55 milliseconds at the prover side and about 126 milliseconds at the verifier side for each authentication transaction. LA^3 outperforms BS because BS needs pairing operations while LA^3 does not, and also the exponential and pairing operations over groups support bilinear mapping are much more expensive than the exponential operation over elliptic cyclic group. Particularly, for bilinear map $e : G \times G \rightarrow G_T$ that we use on type A curve, an exponentiation computation on group G takes 115 milliseconds and a pairing computation takes 150 milliseconds, while an exponential computation on elliptic curve secp160r1 only takes 12 milliseconds. To check whether a signer has been revoked or not, both BS and LA^3 require only the verifier to check a revocation list. Table 6.1 compares the revocation performance between the two schemes: LA^3 is more efficient than BS; particularly, LA^3 needs only 4.4% of the time needed by BS.

Table 6.1 Checking Time (millisecond) vs. Number of Revoked Provers

Number of Revoked Provers	10	20	30	40	50	60	70	80
Time for Checking Revoked Provers (BS Scheme)	2965	5878	8859	11754	14772	17660	20910	23932
Time for Checking Revoked Provers (LA^3 Scheme)	132	262	401	522	655	785	926	1061

Bandwidth consumption In terms of bandwidth consumption for each verification transaction, BS needs to transmit 2 elements from the bilinear group and 5 elements from $\mathbb{Z}_p$. Based on the type A curve that we use in the experiment, each element from $\mathbb{Z}_p$ takes 20 bytes and each element from the bilinear group takes 128 bytes. Hence, the total signature size is 356 bytes. As the element from the bilinear group can be denoted as compressed version, which is 65 bytes, the total length of a group signature in BS is 230 bytes in the compressed format. Note that the length of group signature in BS scheme can be shorter if other curves such as type D curve is used [128]. However, this benefit is paid by the overhead of degraded computational efficiency. In order to show the computational efficiency of

the proposed scheme, we choose type A curve when implementing BS scheme as type A curve is the fastest curve for pairing operation.

With LA^3, the prover needs to submit 7 points from the elliptic curve and 3 elements from $\mathbb{Z}_p$. As the secp160r1 elliptic curve we used in the experiment is 160-bit elliptic curve, each element from $\mathbb{Z}_p$ takes 20 bytes and each point from elliptic curve can be represented by 41 bytes. Hence, the total bandwidth consumption of LA^3 is 347 bytes. Note that, each point from elliptic curve can also be represented by the compressed version, which takes 21 bytes. Hence, by using the compressed representation, the bandwidth consumption of LA^3 is 207 bytes.

6.5.3 Performance Comparison with Other VLR Group Signature Schemes

We compare the computational costs of LA^3 with that of other VLR group signature schemes in Table 6.2. As we can see, LA^3 is much more efficient than all these schemes because (i) LA^3 needs no pairing operation and smaller number of exponential operations, and (ii) the exponential operation over a regular multiplicative group is much more efficient than that in a group support pairing operations.

Table 6.2 Comparison of Computational Costs of Group Signature Schemes [103, 109, 119, 110]

Schemes	Signing	Verification
LA^3	$5E$	$(7+\|RL\|)E$
BS Scheme [103]	$8E$+$2P$	$6E$+$(3+2\|RL\|)P$
Scheme [109]	$10E$+$1P$	$6E$+$(2+\|RL\|)P$
Scheme [119]	$6E$+$1P$	$3E$+$(2+2\|RL\|)P$
Scheme [110]	$(6+8\|RL\|)E$	$(9+8\|RL\|)E+3\|RL\|P$

E and P: exponential and pairing operations, respectively; $|RL|$: number of revoked users.

6.5.4 Security Property Comparison with the BS Scheme [103]

Comparing to the above group signature schemes, our proposed LA^3 is more computationally efficient than any of them, and LA^3 is a VLR scheme in which revocation is transparent to provers. As BS scheme [103] is the most similar to LA^3 and it has many applications [92, 93, 94, 95], we provide the following more detailed comparison:

LA^3 is similar to the BS scheme in that, they both provide selfless anonymity for provers (signers) and the tracing and revocation capabilities for the trusted authority, and they are both VLR schemes. They are different in that, the BS scheme can publish public group keys to allow everyone knowing the public key to perform verification. LA^3, on the other hand, only allows the so-called verifier, which is

typically the group manager in practical systems, to verify signatures. However, LA^3 has the feature of non-frameability; that is, it is hard for any coalition of group members (including the group manager) to impersonate an innocent group member. Therefore, we expect LA^3 will still be useful in many practical scenarios due to the following reasons: First, in many cases (e.g., authentication for resources access [92]), only the group manager needs to perform verification. Second, with LA^3, only group manager can forge fake signature that cannot trace to any valid group member; hence, if an untraceable signature is found, it can be determined that the group manager must be responsible for it. This feature can deter a group manager from forging signatures.

6.6 Conclusions

The chapter presents LA^3, a lightweight accountable and anonymous authentication scheme. The proposed design is based on the hardness of the CDH, DDH, q-SDH, q-DDHI and LRSW problems. If the above problems are hard, the LA^3 scheme has the security properties of non-frameability, traceability and selfless anonymity. The LA^3 scheme is implemented and compared to existing group signature schemes. The results shows that the LA^3 achieves much higher computational efficiency.

CHAPTER 7. AdHocSign: an Ad Hoc Group Signature Scheme for Accountable and Anonymous Access to Outsourced Data

To preserve the access privacy of the data in the cloud, the existing group signature schemes can be applied such that all the users who are allowed to access this specific data form an ad hoc group. The trusted authority can generate and distribute the keys for each specific data when a new group is formed. To address the system dynamism and scalability issues, this chapter presents a new group signature scheme (named AdHocSign) for dynamically formed groups. The proposed scheme has been implemented and the evaluation results show that its computational cost is comparable to that of a state-of-the-art group signature scheme.

7.1 Introduction

With the increasing adoption of the cloud computing paradigm, storing sensitive user data to untrusted, remote hosts on Internet has been popular. To achieve accountable anonymous access of the outsourced data, the existing group signature schemes [100, 103, 106] can be applied. Specifically, for each piece of outsourced data, an *ad hoc group* can be formed to include all the users who are allowed to access for this specific data. The trusted authority needs to be contacted to compute the public key for this new group, and the authority needs to communicate with each of the users in the new group to distribute the private keys. These operations may incur high communication overhead in terms of bandwidth consumption and delay, especially when the group size is large, the group members are distributed largely, or the group members are intermittently connected to the network. Considering the fact that a user is more likely to be enabled to access many data items in the cloud, the key management for users is cumbersome as they need to manage different keys for different groups. In addition, this approach does not support system dynamism and scalability. For example, when the access privilege

of existing data is changed, the trust authority may need regenerate an ad hoc group, and distribute the corresponding keys to the authorized users; when new data is delivered into cloud, a new ad hoc group may need to be formed if its access privilege is different from existing ones.

To address the above problem, researchers have been proposed attribute based schemes [129, 130, 131, 136, 137]. In these schemes, each user owns a set of attributes and a set of secrets derived from the attributes. When a piece of data is outsourced to a host, the data is labeled with an access structure to specify who are allowed to access the data; the access structure is expressed in attributes and logical operations over the attributes. A user is allowed to access certain data if and only if the attributes owned by the user satisfy the access structure of the data. These schemes provide user anonymity as well as system dynamism and scalability. Specifically, a user can access the data that he/she is authorized to access without exposing his/her identity to the data host; also, when a new access structure is defined for certain data, there is no need to distribute new attributes or secrets to users or change the attributes or secrets owned already by users. These schemes, however, do not provide accountability, which is necessary to trace out misbehaving users and stop them from abusing the privacy preservation features.

To provide user accountability while enforcing access control and protecting user anonymity, we propose a new signature scheme, named *AdHocSign*, for ad hoc groups that are defined dynamically according to access structures. Instead of distributing group private keys to the members of an ad hoc group when the group is created (i.e., when a new access structure is defined), the new scheme pre-loads some key materials to individual users when they join the system and are given attributes. When certain data is posted to a host, the host is given certain auxiliary information that is computed by a trusted authority according to the access structure of the data. When a user needs to access a piece of data, it contacts the host of the data to obtain the access structure of the data and the afore-mentioned auxiliary information pre-loaded to the host by the authority. If the user's attributes satisfy the access structure (i.e., the user is a member of the ad hoc group defined by the access structure), the user can compute his/her own private key for the ad hoc group based on the auxiliary information and pre-loaded key materials, and authenticates himself/herself to the host. The user does not expose his/her identity or ownership of attributes during the authentication.

We first propose a version of AdHocSign scheme that is applicable to the scenarios where all access structures can be expressed as conjunction-only logical expressions of attributes. Then, it is extended

to an advanced version that is applicable to the scenarios where access structures are general (i.e., each access structure can be expressed as a conjunction of disjunctive logical expressions of attributes). The designs are based on the assumptions that the q-Strong Diffie-Hellman (q-SDH) problem and the Decision Linear problem are hard in a multiplicative cyclic group $\mathbb{G}_1$. Rigorous security analysis has been conducted to show that, if the q-SDH and the Decision Linear problem are hard, the scheme is both selflessly-anonymous and traceable. We have also implemented the AdHocSign scheme, and evaluated and analyzed its computational and storage overhead. The results show that, the computational cost of the scheme is comparable to the group signature scheme proposed by Boneh and Shacham [103]. A trade-off between system scalability and user's storage overhead exists in the AdHocSign scheme for general access structures; that is, to support more dynamically-constructed access structures, more key materials should be preloaded to users and thus higher storage overhead will be introduced.

7.2 Preliminaries

7.2.1 System Model

We consider a distributed system composed of one or multiple data storage sites (called data *hosts*), multiple users, and an authority trusted by all the users and hosts. To facilitate access control to the data stored at the hosts, a set of attributes are defined. Each user e has a unique identity number denoted as $x_e \in \mathbb{Z}_p$, where $\mathbb{Z}_p$ is a field of integers modus a large prime number p, and is assigned a set of attributes denoted as $\{a_{e,1}, \cdots, a_{e,n_e}\}$.

When a piece of data is posted to a host, who are allowed to access the data is specified as a logical expression containing attributes and conjunction ($\wedge$) or disjunction ($\vee$) operators on the attributes. We call the logical expression an *access structure*. Generally, an access structure T can be defined as

$$T = DT_1 \wedge \cdots \wedge DT_s, \tag{7.1}$$

where

$$DT_i = a_{T,i,1} \vee \cdots \vee a_{T,i,s_i} \tag{7.2}$$

for each $i = 1, \cdots, s$. For example, $T = a_1 \wedge (a_2 \vee a_3) \wedge (a_4 \vee a_5 \vee a_6)$ is an access structure defined based on attributes $a_1, \cdots, a_6$.

As an access structure can be defined on the fly when a piece of data is posted, the group of users allowed to access the data is not predefined; hence, we call such a group an *ad hoc group* and our proposed scheme is to provide a mechanism for members of such an ad hoc group to sign messages in an anonymous and accountable manner. To protect the confidentiality of the data, we assume a certain encryption scheme, for example, the CP-ABE scheme [131], is used to encrypt the data to prevent the host and unauthorized users from decrypting the data.

To summarize, when a user in our system wants to access a piece of data, he/she first authenticates himself/herself to the host of the data using our proposed ad hoc group signature (*AdHocSign*) scheme. After the authentication succeeds, the data, in the encrypted form, is returned to the user, who can decrypt the data using the ABE schemes. To facilitate the authentication and data decryption, a user is preloaded with some information when she is assigned attributes when he/she joins the system.

7.2.2 Bilinear Pairing

Let $\mathbb{G}_1$ be a multiplicative cyclic group of prime order p. Let g be a generator of $\mathbb{G}_1$ and E be a bilinear map defined as $E : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. The bilinear map E has the following properties:

- Bilinearity: $\forall g_1, g_2 \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p$, it holds that $E(g_1^a, g_2^b) = E(g_1, g_2)^{ab}$.
- Non-degeneracy: $E(g, g) \neq 1$.

We also assume that $\mathbb{G}_1$ is a bilinear group. That is, the group operation in $\mathbb{G}_1$ and the bilinear map $E : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ are both efficiently computable.

Note: in a more general-case scenario, a bilinear map can be defined as $E : \mathbb{G}_1 \times \mathbb{G}_1' \rightarrow \mathbb{G}_2$. To simplify the presentation, this paper assumes $\mathbb{G}_1 = \mathbb{G}_1'$ though our proposed scheme can be extended to the more general-case scenario.

7.2.3 Group Signature Scheme by Boneh and Shacham

Our proposed AdHocSign scheme is designed based on Boneh and Shacham's verifier-local group signature scheme [103], which includes the following primitives:

- *BSG_KeyGen(n).* The primitive outputs a group public key gpk, an n-element vector of user private keys $gsk = (gsk[1], \cdots, gsk[n])$, and an n-element vector of user revocation tokens $grt = (grt[1], \cdots, grt[n])$.

- *BSG_Sign(gpk,$gsk[i]$,M).* The primitive outputs a signature σ of message M for user i which owns private key $gsk[i]$.

- *BSG_Verf(gpk,RL,σ,M).* The primitive verifies whether σ is a valid signature of message M signed by any user whose membership has not been revoked (i.e., the revocation key of the user is not in the revocation list RL).

- *BSG_Trace(M,σ,grt).* Given a message M, a signature σ of the message and the vector of revocation tokens grt, the primitive outputs the index of the user who generated the signature.

Hash functions H_0 and H are used in the scheme. Also, due to the correctness of the scheme, the following property holds:

$$BSG\_Verify(gpk, RL, BSG\_Sign(gpk, gsk[i], M), M) \Leftrightarrow grt[i] \neq RL. \tag{7.3}$$

7.2.4 AdHocSign: Definition and Security Model

7.2.4.1 Primitives

The AdHocSign scheme provides the following primitives:

- *Setup*. The primitive chooses system parameters.

- *AttributeInit(a).* The primitive chooses and outputs the secrets $\mathbb{S}[a]$ for attribute a.

- *UserInit(e,$\mathbb{A}[e]$,$\mathbb{S}$).* This primitive is called when a user joins the system. The primitive takes as inputs an user ID e, the set of attributes $\mathbb{A}[e]$ owned by the user, and the set of secrets $\mathbb{S}$ for all attributes. It outputs the private key gsk_e of the user.

- *AccessStructureInit(T,$\mathbb{S}$).* This primitive is called when a new access structure is defined. It takes as inputs an access structure T and the set of secretes $\mathbb{S}$ for all attributes. It outputs the public key gpk_T regarding the access structure.

- *Sign(gpk_T,gsk_e,M).* This primitive takes as inputs the public key gpk_T regarding a certain access structure T, a private key gsk_e for a certain user e, and a message M. It outputs signature $\sigma_{M,T}$ on M regarding T for user e, or $NULL$ if the attributes owned by the user do not satisfy the access structure.

- *Revoke(gpk_T,gsk_e).* The primitive is called to get the revocation token of user e regarding access structure T. It takes as inputs the public key gpk_T regarding T and the private key gsk_e of e, and outputs the revocation token $grt_T[e]$.

- *Verf(gpk_T,RL,σ,M).* The primitive takes as inputs the public key gpk_T regarding T, list RL of revocation tokens, a signature σ and message M. It outputs *valid* if and only if σ is a valid signature of M regarding T that was generated by a user whose revocation token is not in RL.

- *Trace(M,σ,grt_T).* This primitive takes as inputs message M, signature σ and the set of revocation tokens grt_T regarding T. It outputs the ID of the user who generated the signature.

7.2.4.2 Security Assumptions

The AdHocSign scheme is designed based on the assumptions about the hardness of the following problems.

- q-Strong Diffie-Hellman (q-SDH) Problem in $\mathbb{G}_1$: Given a $(q+1)$-tuple g, g^x, $g^{(x^2)}$, $\cdots$, $g^{(x^q)}$ of group $\mathbb{G}_1$ as input, output a pair $(c, g^{1/(x+c)})$ where $c \in \mathbb{Z}_p$. We say that the (q,t,ϵ)-SDH assumption holds in $\mathbb{G}_1$ if no t-time algorithm has the probability of at least ϵ in solving the q-SDH problem in $\mathbb{G}_1$.

- Decision Linear Problem in $\mathbb{G}_1$: Given u, v, h, u^a, v^b, $h^c \in \mathbb{G}_1$ as input, output yes if $a+b=c$ and no otherwise. We say that the (t,ϵ)-Decision Linear assumption holds in $\mathbb{G}_1$ if no t-time algorithm has probability of at least $\epsilon + 1/2$ in solving the Decision Linear problem in $\mathbb{G}_1$.

7.2.4.3 Security model

The AdHocSign scheme should satisfy three requirements: correctness, traceability, and selfless-anonymity, which are formally defined as follows.

Correctness Given an access structure T, every signature generated by a user whose attributes satisfy the access structure must be verified as valid, except when the user is revoked. That is,

$$[\sigma = Sign(gpk_T, gsk_e, M) \neq NULL] \Longrightarrow$$

$$[Verf(gpk_T, RL, \sigma, M) = valid \iff grt_T[e] \notin RL].$$

Traceability The AdHocSign scheme is traceable if no adversary can win the traceability game defined below. In the traceability game, the adversary attempts to forge a signature that cannot trace to any of the users in his coalition using the *Trace* algorithm. Particularly, the traceability game between a challenger and an adversary $\mathcal{A}$ is defined as follows.

- **Initialization.** The challenger runs algorithm *Setup* to get system parameters, and provides the public parameters to $\mathcal{A}$. He also initializes the coalition of $\mathcal{A}$, denoted as U, to $\emptyset$.
- $\mathcal{A}$ can make following types of query to the challenger.
  - *Corruption*. $\mathcal{A}$ requests the private key gsk_e of some user e who owns a certain set of attributes $\mathbb{A}[e]$. The challenger appends e to U, and responds with gsk_e.
  - *Access Structure Initialization*. $\mathcal{A}$ requests the public key gpk_T for a certain access structure T. The challenger responds with gpk_T, which is computed by using the *AccessStructureInit* algorithm.
  - *Signing*. $\mathcal{A}$ requests a signature on message M regarding access structure T for user e. The challenger computes and replies $\sigma \leftarrow$Sign(gpk_T,gsk_e,M).
- **Response.** Finally, $\mathcal{A}$ outputs M^*, a set of RL^* of revocation tokens, and a signature σ^*.

  $\mathcal{A}$ wins if: (i) σ^* is accepted by the verification algorithm as a valid signature on M^* regarding T; (ii) σ^* traces to a user outside of the coalition $U \setminus RL^*$, or the tracing algorithm fails; and (iii) σ^* is nontrivial, i.e., $\mathcal{A}$ did not obtain σ^* by making a signing query at M^* regarding T', where every set of attributes satisfying T' also satisfies T.

Algorithm $\mathcal{A}$ $(t, q_H, q_S, n, m, \epsilon)$-breaks traceability in an n-user m-attribute AdHocSign scheme if: $\mathcal{A}$ runs in time at most t; $\mathcal{A}$ makes at most q_H hash oracle queries on hash functions H_0 and H, and at most

q_S signing queries; and $\mathcal{A}$ wins the traceability game with a probability of at least ϵ. If no algorithm can $(t, q_H, q_S, n, m, \epsilon)$-break traceability in an AdHocSign scheme, the scheme is $(t, q_H, q_S, n, m, \epsilon)$-traceable.

Selfless-anonymity The AdHocSign scheme is selflessly-anonymous if no adversary can win the selfless-anonymity game defined below. In the game, the adversary attempts to determine which of two users generated a signature when it is not given access to the secrets held by these two users. Particularly, the game is defined as follows.

- **Initialization.** The challenger runs algorithm *Setup* to get system parameters and provides the public parameters to $\mathcal{A}$.
- **Queries.** As in the traceability game, $\mathcal{A}$ can make *corruption*, *access structure initialization* and *signing* queries to the challenger, who responds as in the traceability game.
- **Challenge.** $\mathcal{A}$ outputs a message M, an access structure T, and two user IDs i_0 and i_1. It must have not issued a corruption or revocation query at either user. The challenger chooses a bit b from $\{0, 1\}$ uniformly at random, computes a signature on $\sigma^* \leftarrow Sign(gpk_T, gsk_{i_b}, M)$, and provides σ^* to $\mathcal{A}$.
- **Restricted Queries.** After obtaining the challenge, $\mathcal{A}$ is allowed to make additional *corruption*, *access structure initialization* and *signing* queries with the restriction that the *Corruption* query cannot be made at users i_0 and i_1.
- **Response.** Finally, $\mathcal{A}$ outputs a bit $b^{'}$, his guess of b. The adversary wins if $b^{'} = b$.

Letting $\mathcal{A}$'s advantage in winning the game be $AdvSA_{\mathcal{A}} = |Pr[b = b'] - \frac{1}{2}|$, $\mathcal{A}$ $(t, q_H, q_S, n, m, \epsilon)$-breaks selfless-anonymity in an n-user m-attribute AdHocSign scheme if: $\mathcal{A}$ runs in time at most t; $\mathcal{A}$ makes at most q_H queries to hash functions H_0 and H, and at most q_S signing queries; and $AdvSA_{\mathcal{A}}$ is at least ϵ. If no algorithm can $(t, q_H, q_S, n, m, \epsilon)$-break selfless-anonymity in an AdHocSign scheme, the scheme is $(t, q_H, q_S, n, m, \epsilon)$-selflessly-anonymous.

7.3 Construction for Conjunction-only Access Structures

In this section, we present the design of AdHocSign when the access structure is a conjunction-only logical expression of attributes. Specifically, the general format of a conjunction-only access structure is as follows:

$$T = a_{T,1} \wedge \cdots \wedge a_{T,s}, \tag{7.4}$$

where each $a_{T,i}$ $(i = 1, \cdots, s)$ is an attribute.

7.3.1 Algorithms

The following algorithms implement the primitives defined in Section 7.2.4.1.

7.3.1.1 CO_Setup.

The setup algorithm chooses the following system parameters:

- a finite field of integers $\mathbb{Z}_p$ where p is a large prime number,
- a multiplicative cyclic and bilinear group $\mathbb{G}_1$ of prime order p,
- bilinear map E: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$,
- a generator g of group $\mathbb{G}_1$, and
- a secret element $\gamma \in \mathbb{Z}_p$.

7.3.1.2 CO_AttributeInit(a).

The attribute initialization algorithm takes as input an attribute a, and outputs a secret number denoted as α_a, where α_a is picked from $\mathbb{Z}_p$ uniformly at random. α_a is also stored at vector $\mathbb{S}$ at index a; i.e., $\mathbb{S}[a] \leftarrow \alpha_a$.

7.3.1.3 CO_UserInit($e, \mathbb{A}[e], \mathbb{S}$).

The user initialization algorithm takes the following inputs:

- e: the user ID;

- $\mathbb{A}[e] = \{a_{e,i} | i = 1, \cdots, n_e\}$: the set of attributes assigned to user e; and
- $\mathbb{S}$: the set of secret numbers for attributes.

The output of the algorithm is a private key of the user, i.e.,

$$gsk_e = \langle x_e, \{(a_{e,i}, A_{e,a_{e,i}}) | i = 1, \cdots, n_e\}\rangle, \tag{7.5}$$

where x_e is picked from $\mathbb{Z}_p$ uniformly at random and

$$A_{e,a_{e,i}} = g^{\frac{\alpha_{a_{e,i}}}{\gamma + x_e}}. \tag{7.6}$$

7.3.1.4 CO_AccessStructureInit(T, $\mathbb{S}$).

The algorithm takes as inputs the access structure T and the set of secrets $\mathbb{S}$ for all attributes. It picks $r_{a_{T,i}}$ ($i = 1, \cdots, s$) from $\mathbb{Z}_p$ uniformly at random, for each attribute $a_{T,i}$ present in T. Then, it computes and outputs the public key, denoted as gpk_T, regarding T. In particular,

$$gpk_T = \langle T, \{r_{a_{T,1}}, \cdots, r_{a_{T,s}}\}, g_T, w_T\rangle, \tag{7.7}$$

where

$$g_T = g^{\sum_{k=1}^{s}(r_{a_{T,k}} \cdot \alpha_{a_{T,k}})} \text{ and } w_T = g_T^{\gamma}. \tag{7.8}$$

7.3.1.5 CO_Sign(gpk_T,gsk_e,M).

The inputs of the signature generation algorithm include

- gpk_T: the public key regarding access structure T for which a signature is to be generated;
- gsk_e: the private key held by user e for whom the signature is to be generated; and
- M: the message to be signed.

The algorithm outputs $\sigma_{M,T}$, signature of message M regarding T. The signature is generated in the following steps:

1. If user e does not have all attributes present in T, $NULL$ is returned. Otherwise, the following steps are executed.

2. $\hat{A}_{e,T} \leftarrow \prod_{i=1}^{s} A_{e,a_{T,i}}^{r_{a_{T,i}}}$;

3. $\sigma_{M,T} \leftarrow BSG\_Sign(\{g_T, w_T\}, \{\hat{A}_{e,T}, x_e\}, M)$. Recall that BSG_Sign is the signing primitive in the group signature scheme proposed by Boneh and Shacham. When calling the primitive, $\{g_T, w_T\}$ is the public key and $\{\hat{A}_{e,T}, x_e\}$ is the user private key.

7.3.1.6 CO_Revoke(gpk_T,gsk_e).

This algorithm takes as inputs the public key gpk_T for a certain access structure T and the private key gsk_e for a certain user e. It computes the revocation token of e regarding T as

$$grt_T[e] \leftarrow \prod_{i=1}^{s} A_{e,a_{T,i}}^{r_{a_{T,i}}}.$$

Then, the algorithm returns $grt_T[e]$.

7.3.1.7 CO_Verf(gpk_T,RL_T,σ,M).

To verify if σ is a signature of message M regarding access structure T, the verification algorithm takes the public key gpk_T and the list RL_T of revocation keys as inputs. The algorithm can be implemented by calling

$$BSG\_Verf(\{g_T, w_T\},\ RL_T,\ \sigma,\ M).$$

Recall that BSG_Verf is the revocation primitive in the group signature scheme proposed by Boneh and Shacham.

7.3.1.8 CO_Trace(M, σ, grt_T).

This algorithm can be implemented by calling $BSG\_Trace(M, \sigma, grt_T)$.

7.3.2 Security Analysis

7.3.2.1 Correctness

Theorem 7.3.1 *The AdHocSign scheme for conjunction-only access structures is correct. Formally, if a user e has all attributes present in an access structure as defined in Equation (7.4), then*

$$\{CO\_Verf(gpk_T, RL_T, CO\_Sign(gpk_T, gsk_e, M), M) = valid\} \Leftrightarrow \{grt_T[e] \notin RL_T\}. \quad (7.9)$$

Proof Suppose user e has all attributes appearing in a conjunction-only access structure T. According to algorithms *CO_Sign* and *CO_Verify*, it holds that

$$CO\_Sign(gpk_T, gsk_e, M) = BSG\_Sign(\{g_T, w_T\}, \{\hat{A}_{e,T}, x_e\}, M)$$

and

$$CO\_Verf(gpk_T, RL_T, \sigma, M) = BSG\_Verf(\{g_T, w_T\}, RL_T, \sigma, M),$$

where

$$w_T = g_T^{\gamma}, \quad \hat{A}_{e,T} = g_T^{\frac{1}{\gamma + x_e}}, \quad \text{and } \sigma = CO\_Sign(gpk_T, gsk_e, M).$$

As the group signature scheme proposed by Boneh and Shacham is correct, it holds that

$$BSG\_Verf(\{g_T, w_T\}, RL_T, \sigma, M) \Leftrightarrow \hat{A}_{e,T} \notin RL_T.$$

Also, $grt_T[e] = \hat{A}_{e,T}$ according to algorithm *CO_Revoke*. Hence, Equation (7.9) holds.

7.3.2.2 Traceability

Based on the definitions of traceability game and traceability in Section 7.2.4.3, the traceability of the AdHocSign scheme for conjunction-only access structure is stated in Theorem 7.3.2.

Theorem 7.3.2 *If the (q, t', ϵ')-SDH assumption holds in $\mathbb{G}_1$, the AdHocSign scheme for conjunction-only access structures is $(t, q_H, q_S, n, m, \epsilon)$-traceable, where $n = q - 1$, $\epsilon = 8n\sqrt{\epsilon' q_H} + 2n/p$, and $t' = \Theta(1) \cdot (t + m \cdot q)$.*

Proof In Appendix B.1.

7.3.2.3 Selfless-anonymity

The selfless-anonymity of our proposed scheme is stated in Theorem 7.3.3.

Theorem 7.3.3 *The AdHocSign scheme for conjunction-only access structures is (t, q_H, q_S, n, m, ϵ)-selflessly-anonymous assuming the (t', ϵ') Decision Linear assumption holds in group $\mathbb{G}_1$ for $\epsilon' = \frac{\epsilon}{2}(\frac{1}{n^2} - \frac{q_S q_H}{p})$ and $t' = \Theta(1) \cdot (t + mn)$.*

Proof In Appendix B.2.

7.3.3 Overhead Analysis

With the AdHocSign scheme for conjunction-only access structures, a user needs to perform the following computations to sign a message regarding an access structure of form $T = a_1 \wedge \cdots \wedge a_s$: (i) s exponential operations in $\mathbb{G}_1$, (ii) s multiplication operations in $\mathbb{G}_1$, and (iii) one invocation of BSG_Sign primitive. As shown in the results of system implementation and evaluation (Fig. 7.2 in Section 5), the computational cost of computing a private key (i.e., steps (i) and (ii)) is lower than that of BSG_Sign when s is not large.

Note that the computational cost of verifying a signature is the same as that of BSG_Verify primitive. Also a user in the AdHocSign scheme only needs to keep one piece of secret for each attribute owned by itself, which is the same as in Boneh and Shacham's group signature scheme [103].

7.4 Construction for General Access Structures

To lay the foundation for our construction for general access structures, our construction for disjunction-only access structures is presented first, which is then followed by the construction for general access structures and the security and overhead analysis of the construction.

7.4.1 Construction for Disjunction-only Access Structures: The Algorithms

We now describe the AdHocSign scheme for disjunction-only access structures as $T = a_{T,1} \vee \cdots \vee a_{T,s}$, where each $a_{T,i}$ $(i = 1, \cdots, s)$ is an attribute.

7.4.1.1 DO_Setup.

The algorithm is the same as CO_Setup, except that (i) a secret ξ is also randomly chosen from $\mathbb{Z}_p$, and (ii) hash functions H_1 and H_2 are chosen where H_1 hashes four elements of $\mathbb{Z}_p$ to an element of $\mathbb{G}_1$ and H_2 hashes three elements of $\mathbb{Z}_p$ to an element of $\mathbb{Z}_p$.

7.4.1.2 DO_AttributeInit(a, N).

The algorithm takes two inputs: a which is ID of the attribute, and N which is an integer. The algorithm outputs N secret numbers to be associated with attribute a, denoted as $\alpha_{a,i}$ for $i = 1, \cdots, N$,

where each $\alpha_{a,i}$ is picked from $\mathbb{Z}_p$ uniformly at random. All these secret numbers are recorded in vector $\mathbb{S}[a]$.

7.4.1.3 DO_UserInit(e, $\mathbb{A}[e]$, $\mathbb{S}$).

The inputs to the user initialization algorithm include the user ID e, the set of attributes

$$\mathbb{A}[e] = \{a_{e,i} | i = 1, \cdots, n_e\}$$

owned by the user, and the set of secret keys

$$\mathbb{S} = \{\alpha_{a_{e,i},j} | i = 1, \cdots, n_e; j = 1, \cdots, N\}$$

for the above attributes.

The algorithm outputs the private key for user e, i.e.,

$$gsk_e = \langle x_e, \{(a_{e,i}, A'_{e,a_{e,i},1}, \cdots, A'_{e,a_{e,i},N}) | i = 1, \cdots, n_e\}\rangle,$$

where x_e is picked from $\mathbb{Z}_p$ uniformly at random,

$$A'_{e,a_{e,i},j} = A_{e,a_{e,i},j} \cdot H_1(e, a_{e,i}, j, H_2(\xi, a_{e,i}, j))^{-1} \text{ for } j = 1, \cdots, N,$$

and

$$A_{e,a_{e,i},j} = g^{\frac{\alpha_{a_{e,i},j}}{\gamma + x_e}}.$$

7.4.1.4 DO_AccessStructureInit(T, $\mathbb{S}$).

The inputs to the access structure initialization algorithm include the access structure T and the set of secret keys $\mathbb{S}$ for all attributes.

The algorithm picks r_T from $\mathbb{Z}_p$ uniformly at random. For each $a_{T,i}$, the algorithm also picks $\delta_{T,i}$ from $\{1, \cdots, N\}$ such that $\alpha_{a_{T,i},\delta_{T,i}}$ is a secret associated with $a_{T,i}$ that has not been used in access structure initialization before. If such $\delta_{T,i}$ cannot be found successfully, the primitive fails. Otherwise, the algorithm computes and outputs the following public key regarding T:

$$gpk_T = \langle T, \tilde{R}_T, g_T = g^{r_T}, w_T = g_T^{\gamma}\rangle,$$

where

$$\tilde{R}_T = \{(r_{T,i} = \frac{r_T}{\alpha_{a_{T,i},\delta_{T,i}}}, \delta_{T,i}, h_{T,i} = H_2(\xi, a_{T,i}, \delta_{T,i})) | i = 1, \cdots, s\}.$$

7.4.1.5 DO_Sign(gpk_T, gsk_e, M).

The signature generation algorithm takes as inputs public key gpk_T regarding T, private key gsk_e of user e, and message M. It outputs signature $\sigma_{M,T}$ on message M regarding T for user e as follows:

1. If user e does not have any attribute appearing in T, $NULL$ is returned.

2. Otherwise, assuming the user owns attribute $a_{T,i}$, the user computes

$$A_{e,a_{T,i},\delta_{T,i}} = (A'_{e,a_{T,i},\delta_{T,i}}) \cdot H_1(e, a_{T,i}, \delta_{T,i}, h_{T,i}) \text{ and } \hat{A}_{e,T} = A^{r_{T,i}}_{e,a_{T,i},\delta_{T,i}},$$

and returns

$$\sigma_{M,T} = BSG\_Sign(\{g_T, w_T\}, \{\hat{A}_{e,T}, x_e\}, M).$$

7.4.1.6 DO_Verify(gpk_T,RL_T,σ,M).

The algorithm is the same as CO_Verify.

Discussion: According to the above AdHocSign scheme, for every dynamically-constructed conjunction-only access structure, one unique secret associated with every attribute occurring in the access structure is consumed (i.e., it cannot be used for other access structure any more) at every user owning the attribute.

The reason that the secret of an attribute cannot be reused for two or more access structures can be explained intuitively with the following example. Consider two access structures as follows:

$$T_1 = a \vee b,$$

and

$$T_2 = a \vee c,$$

and a user e who owns attribute b but not a and c. The user is thus preloaded with x_e and $A'_{e,b}$. Suppose that, in the public key gpk_{T_1} for T_1, parameters $r_{T_1,a}$ and $r_{T_1,b}$ are associated with attributes a and b; in the public key gpk_{T_2} for T_2, parameters $r_{T_2,a}$ and $r_{T_2,c}$ are associated with attributes a and c. Assume user e is only preloaded with one secret (denoted as α_b) associated with b. Also, it is assumed that, for each user owning attribute a, the same secret of attribute a is used for deriving the private keys for T_1 and T_2.

Hence, the private key of user e for T_1 is

$$(x_e, \hat{A}_{e,T_1} = A_{e,b}^{r_{T_1,b}}),$$

where $A_{e,b} = A'_{e,b} \cdot H_1(e, b, 1, h_{T_1,b})$. Furthermore, however, user e is also able to derive the private key for T_2

$$(x_e, \hat{A}_{e,T_2} = A_{e,b}^{r_{T_1,b}/r_{T_1,a} \cdot r_{T_2,a}})$$

though e does not own attribute a or c. This is because of the following:

- If user e had owned attribute b and thus been preloaded with $A'_{e,b}$, it can derive $A_{e,a}$ as

$$A_{e,a} = g^{\frac{\alpha_a}{\gamma + x_e}} = \left(g^{\frac{\alpha_b}{\gamma + x_e}}\right)^{\alpha_a/\alpha_b} = A_{e,b}^{\alpha_a/\alpha_b} = A_{e,b}^{r_{T_1,b}/r_{T_1,a}},$$

  where $A_{e,b} = A'_{e,b} \cdot H_1(e, b, 1, h_{T_1,b})$.

- Hence, it holds that

$$\hat{A}_{e,T_2} = A_{e,a}^{r_{T_2,a}} = A_{e,b}^{r_{T_1,b}/r_{T_1,a} \cdot r_{T_2,a}}.$$

7.4.2 Construction for General Access Structures: The Algorithms

Integrating the AdHocSign schemes for disjunction-only access structures and for conjunction-only access structures, the AdHocSign scheme for general access structures is designed as follows.

7.4.2.1 G_Setup, G_AttributeInit(i,N), G_UserInit(e, $\mathbb{A}[e]$, $\mathbb{S}$) and G_Verify(gpk_T, RL_T, σ, M).

The algorithms are the same as DO_Setup, DO_AttributeInit, DO_UserInit and CO_Verify, respectively.

7.4.2.2 G_AccessStructureInit(T, $\mathbb{S}$).

The inputs to the access structure initialization algorithm include the access structure T as defined in Equations (7.1) and (7.2), and the set of private keys $\mathbb{S}$.

For each $DT_i = a_{T,i,1} \vee \cdots \vee a_{T,i,s_i}$, the algorithm picks $r_{T,i}$ from $\mathbb{Z}_p$ uniformly at random. Then, for each $a_{T,i,j}$ that is a part of DT_i, the algorithm finds $\delta_{T,i,j}$ from $\{1, \cdots, N\}$ such that $\alpha_{a_{T,i,j},\delta_{T,i,j}}$ is

a secret associated with attribute $a_{T,i,j}$ and has not been used in access structure initialization before, and computes $r_{T,i,j} = \frac{r_{T,i}}{\alpha_{a_{T,i,j},\delta_{T,i,j}}}$. If such $\delta_{T,i,j}$ for $j = 1, \cdots, s_i$ cannot be found in $\{1, \cdots, N\}$ successfully, the primitive fails. Finally, the algorithm computes and outputs the following public key:

$$gpk_T = \langle T, \{(r_{T,i,j}, \delta_{T,i,j}, h_{T,i,j}) | i = 1, \cdots, s; \text{ for each } i, j = 1, \cdots, s_i\}, \; g_T = g^{\sum_{i=1}^{s} r_{T,i}}, \; w_T = g_T^{\gamma} \rangle, \tag{7.10}$$

where

$$h_{T,i,j} = H_2(\xi, a_{T,i,j}, \delta_{T,i,j}).$$

7.4.2.3 G_Sign(gpk_T, gsk_e, M).

The signature generation algorithm takes as inputs public key gpk_T regarding T, private key gsk_e of user e, and message M. It outputs signature $\sigma_{M,T}$ on message M regarding T for user e as follows:

1. If the attributes owned by user e do not satisfy T, $NULL$ is returned.

2. Otherwise, assuming the user owns attribute a_{T,i,k_i} for $i = 1, \cdots, s$, the user computes

$$\hat{A}_{e,T} = \prod_{i=1}^{s} [A'_{e,a_{T,i,k_i},\delta_{T,i,k_i}} \cdot H_1(x_e, a_{T,i,k_i}, \delta_{T,i,k_i}, h_{T,i,k_i})]^{r_{T,i,k_i}},$$

   and returns

$$\sigma_{M,T} = BSG\_Sign(\{g_T, w_T\}, \{\hat{A}_{e,T}, x_e\}, M).$$

7.4.2.4 G_Revoke(gpk_T,gsk_e)

This algorithm takes as inputs the public key gpk_T for a certain access structure T and the private key gsk_e for a certain user e. It computes the revocation token of e regarding T as

$$grt_T[e] \leftarrow \hat{A}_{e,T},$$

where the computation of $\hat{A}_{e,T}$ is the same as step 2 in primitive G_Sign. Then, the algorithm returns $grt_T[e]$.

7.4.2.5 G_Trace(M, σ, grt_T).

This algorithm can be implemented by calling $BSG\_Trace(M, \sigma, grt_T)$.

7.4.3 Security Analysis

7.4.3.1 Correctness

Theorem 7.4.1 *The AdHocSign scheme for general access structures is correct. Formally, if the attributes owned by a user e satisfy an access structure T as defined in Equation (7.1), then*

$$\{G\_Verify(gpk_T, RL_T, G\_Sign(gpk_T, gsk_e, M), M) = valid\} \Leftrightarrow \{grt_T[e] \notin RL_T\}. \quad (7.11)$$

Proof Assume $T = DT_1 \wedge \cdots \wedge DT_s$, and user e owns attribute a_{i,k_i} that appears in DT_i for $i = 1, \cdots, s$. Hence, private key gsk_e includes the following information:

$$x_e, \{(a_{i,k_i}, A'_{e,a_{i,k_i},1}, \cdots, A'_{e,a_{i,k_i},N}) | i = 1, \cdots, s\},$$

where

$$A'_{e,a_{i,k_i},j} = g^{\frac{\alpha_{a_{i,k_i},j}}{\gamma + x_e}} \cdot H_1(e, a_{i,k_i}, j, H_2(\xi, a_{i,k_i}, j))^{-1} \text{ for } j = 1, \cdots, N.$$

Also, from gpk_T user e can obtain the following information:

$$\{(r_{T,i,k_i} = \frac{r_{T,i}}{\alpha_{a_{T,i,k_i},\delta_{T,i,k_i}}}, \delta_{T,i,k_i}, h_{T,i,k_i} = H_2(\xi, a_{T,i,k_i}, \delta_{T,i,k_i})) | i = 1, \cdots, s\}, g_T = g^{\sum_{i=1}^{s} r_{T,i}}, w_T = g_T^{\gamma}.$$

Based on the above information in gsk_e and gpk_T, user e can obtain

$$\{A_{e,a_{T,i,k_i},\delta_{T,i,k_i}} = A'_{e,a_{T,i,k_i},\delta_{T,i,k_i}} \cdot H_1(e, a_{T,i,k_i}, \delta_{T,i,k_i}, h_{T,i,k_i}) = g^{\frac{\alpha_{a_{T,i,k_i},\delta_{T,i,k_i}}}{\gamma + x_e}} | i = 1, \cdots, s\}.$$

Then, it can compute

$$\hat{A}_{e,T} = \prod_{i=1}^{s} (A_{e,a_{T,i,k_i},\delta_{T,i,k_i}})^{r_{T,i,k_i}} = g^{\frac{\sum_{i=1}^{s} r_{T,i}}{\gamma + x_e}}.$$

It is satisfied that

$$E(w_T g_T^{x_e}, \hat{A}_{e,T}) = E(g_T, g_T);$$

that is, $(\hat{A}_{e,T}, x_e)$ and (g_T, w_T) form a pair of private key and public key for the group signature scheme proposed by Boneh and Shacham.

Further,

$$\begin{aligned} & G\_Verify(gpk_T, RL_T, G\_Sign(gpk_T, gsk_e, M), M) \quad (7.12) \\ = \; & BSG\_Verify(\{g_T, w_T\}, RL_T, BSG\_Sign(\{g_T, w_T\}, \{\hat{A}_{e,T}, x_e\}, M), M). \end{aligned}$$

Due to the correctness of the BSG scheme (i.e., property (7.3)), it holds that

$$G\_Verify(gpk_T, RL_T, G\_Sign(gpk_T, gsk_e, M), M) \Leftrightarrow grt_T[e] \notin RL_T.$$

Also due to $grt_T[e] = \hat{A}_{e,T}$, property (7.11) holds.

Therefore, the correctness of the scheme is proved.

7.4.3.2 Traceability

Theorem 7.4.2 *If the AdHocSign scheme for conjunction-only access structures is $(t', q_H, q_S, n, m, \epsilon)$-traceable, then the AdHocSign scheme for general access structures is $(t, q_H, q_S, n, m, \epsilon)$-traceable, where $t' = O(t \cdot m \cdot N)$ and N is the maximum number of secrets associated with each attribute.*

Proof In Appendix B.3.

Theorem 7.4.3 *If the (q, t', ϵ')-SDH assumption holds in $\mathbb{G}_1$, then the AdHocSign scheme for general access structures is $(t, q_H, q_S, n, m, \epsilon)$-traceable, where $n = q - 1$, $\epsilon = 8n\sqrt{\epsilon' q_H} + 2n/p$, $t' = O(t \cdot m \cdot N)$, and N is the maximum number of secrets associated with each attribute.*

Proof The theorem can be inferred from Theorems 7.3.2 and 7.4.2.

7.4.3.3 Selfless-anonymity

Theorem 7.4.4 *The AdHocSign scheme for general access structures is (t, q_H, q_S, n, m, ϵ)-selflessly-anonymous assuming the (t', ϵ') Decision Linear assumption holds in group $\mathbb{G}_1$ for $\epsilon' = \frac{\epsilon}{2}(\frac{1}{n^2} - \frac{q_S q_H}{p})$, where $t' = \Theta(1) \cdot (t + m \cdot n \cdot N)$, and N is the maximum number of secrets associated with each attribute.*

Proof Similar to the proof of Theorem 7.3.3.

7.4.4 Overhead Analysis

7.4.4.1 Computational Costs

The analysis is similar to that in Section 7.3.3. With the AdHocSign scheme for general access structures, a user needs to perform the following computations to sign a message regarding an access

structure of form $T = DT_1 \wedge \cdots \wedge DT_s$ where each DT_i is an attribute or a disjunction of attributes: (i) s invocations of hash functions, (ii) s exponential operations in $\mathbb{G}_1$, (iii) $2s$ multiplication operations in $\mathbb{G}_1$, and (iv) one invocation of BSG_Sign primitive. As shown in the results of system implementation and evaluation (Fig. 7.2 in Section 5), the cost of computing a private key (i.e., (i)-(iii)) is lower than that of BSG_Sign when s is not large. Besides, the computational cost of verifying a signature is the same as that of BSG_Verify primitive.

7.4.4.2 Storage Costs *vs.* Number of Dynamically-defined Access Structures

A user needs to store N secrets, each of which is an element of $\mathbb{G}_1$, for every attribute owned by it. As we can see from the AccessStructureInit primitive, *one secret of an attribute is consumed for each occurrence of the attribute in access structures*. The larger is N, the more access structures can be constructed. Hence, there is a trade-off between the storage cost at a user and the number of access structures that can be defined dynamically. In practice, as the size of a secret is small, a user is able to store a large number of secrets and thus the number of access structures defined dynamically can be large.

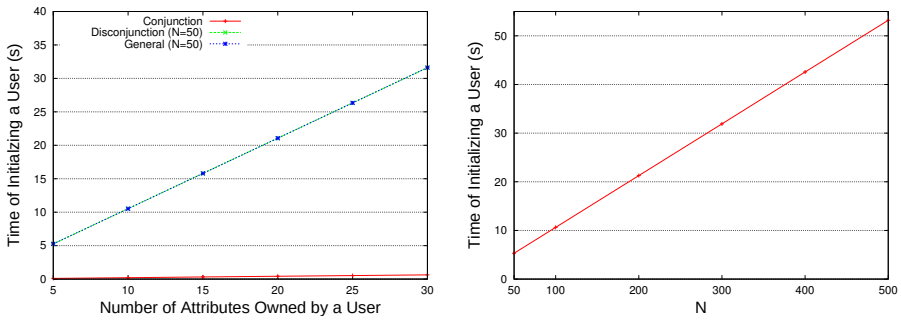
7.5 Implementation and Evaluation

7.5.0.3 System Implementation

We have implemented a proof-of-concept system composed of a client computer, a data host server, and a trusted authority server. The client computer is a desktop computer with 1.83 GHz Genuine Intel (R) processor and 3 GB of RAM. The data host server and the trusted authority server are workstation computers with two 2.13 GHz Intel Xeon (R) processors and 24 GB of RAM. The java paring-based cryptographic (jPBC) library [127] is used to implement the operations on bilinear groups. When using the jPBC libraries, we adopt the type A curves defined at [127].

7.5.0.4 Computational Cost of BSG Primitives

As the group signature scheme [103] provides primitive operations for the proposed scheme, we first measure its computational overhead at the user side and the server side respectively. The signing



(a) (b)

Figure 7.1 Computational cost of UserInit primitives.

primitive (BSG_Sign) is conducted at user side, which takes about 1.65 seconds by average. The signature verification primitive (BSG_Verify) is conducted at the server side. The measurement indicates that the server spends 0.28 seconds by average to check the validity of a signature and about $0.035L$ seconds by average to check if the signature signer is one of L revoked users.

7.5.0.5 Computational Cost of UserInit Primitive

The trusted authority server is responsible for executing the UserInit primitive to initialize a user. The time of initializing a user is linearly increased with the number of attributes that the user has. For each attribute a owned by a user e, the authority needs to compute $A_{e,a}$ if the AdHocSign scheme is for conjunction-only access structures. When the AdHocSign scheme is for general access structures, each attribute is associated with N secrets $A_{e,a,1}, \cdots, A_{e,a,N}$.

Fig. 7.1 (a) shows the average computational cost for UserInit in the AdHocSign schemes for conjunction-only access structures, disjunction-only access structures, and general access structures, respectively. Here, N is fixed at 50 while the number of attributes owned by the user changes varies from 5 to 30. Fig. 7.1 (b) shows the average computational cost for UserInit in the AdHocSign scheme for general access structures, where the number of attributes owned by a user is fixed at 5 while N varies from 50 to 500.

7.5.0.6 Computational Cost of AccessStructureInit Primitive

In the AdHocSign schemes for conjunction-only access structures, disjunction-only access structures and general access structures, the time for executing the AccessStructureInit primitive is almost

the same, i.e., 40ms by average. This is because the execution of the primitive in these three cases all requires 2 exponential operations in $\mathbb{G}_1$ to compute g_T and w_T for structure T, which are the major time-consuming operations.

7.5.0.7 Computational Cost for Deriving a Private Key in Signing Primitive

To sign a message, a client needs to derive the private key and then invoke the BSG_Sign primitive. Fig. 7.2 shows the time for deriving a private key in the AdHocSign schemes for conjunction-only access structures, disjunction-only access structures and general access structures, respectively. As we can see, the computational time spent by the scheme for disjunction-only access structure does not change, while the time spend by the schemes for other two types of access structure increases linearly as the number of attributes in the conjunction-only access structure or the number of disjunction components in the general access structure increases.

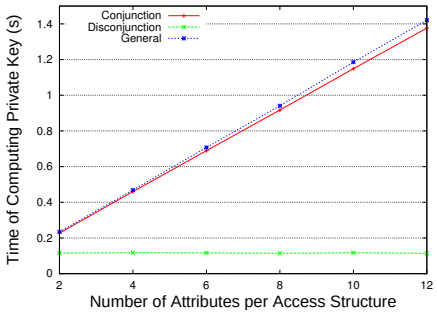


Figure 7.2 Time for deriving a private key.

7.6 Conclusion

We have presented a new group signature scheme (named AdHocSign) for dynamically formed groups, to support accountable and anonymous access to outsourced Data. Rigorous security analysis of the scheme has been conducted to prove its selfless-anonymity and traceability based on the hardness assumption of q-SDH and Decisional Linear problems. The scheme has also been implemented and the evaluation results show that its computational cost is comparable to that of a state-of-the-art group signature scheme.

CHAPTER 8. Concluding Remarks and Future Work

Wireless sensor networks have been widely deployed for data acquisition and collection in many people centric surroundings. Their constrained resources prevent the data processing from being performed within wireless sensor networks. To bridge the gap between data acquisition from wireless sensor networks and data processes within the cloud, the framework of cloud-integrated wireless sensor networks has been proposed. By leveraging the cloud computing capabilities, vast amount of heterogeneous sensory data can be processed, analyzed, and stored in the cloud.

To address the security and privacy vulnerabilities in this framework, we first propose privacy-preserving schemes for collecting and transmitting sensory data into the cloud. We then propose privacy-preserving schemes for accessing and sharing sensory data after being processed within the cloud. Extensive security analysis, simulations and implementations have been conducted to evaluate the effectiveness and efficiency of our proposed schemes. The results show that the proposed schemes address the limitations in existing work and achieve better performance than that of the state-of-the-art schemes in terms of resource efficiency, security strength and privacy protection.

Envisioning the future pervasive computing platform, data acquisition and collection will be even more diverse with the integration of sensing capabilities into mobile phones, vehicles and other daily utility devices. People will be discouraged to contribute and take part in performing tasks and reporting data of their surroundings if they do not feel confident in protecting their privacy. However, the apparent conflicts between data integrity and privacy will be even more challenging in the structure-free mobile environment. We are interested in extending our security and privacy aware schemes to collect people centric sensory data in a dynamic network without stable network topology. After being centralized into the distrust cloud, people centric data remain exposed to serious security and privacy threats. Sensitive private data may be encrypted before being transmitted into the cloud; however the cloud cannot perform data processing if the data is encrypted via classic encryption technique. On the other hand,

the correctness of cloud computing results is questionable. The fully homomorphic encryption (FHE) and secure multi-party computation (SMC) provide generic theoretical solutions to address these issues. However, the complexity of applying FHE and SMC are still prohibitively expensive. Instead, efficient schemes designed for certain specific problems may be developed to address the security and privacy issues in distrust cloud computing. Furthermore, as the continuant proliferating of social networks, it becomes more and more important for people to share their information among their families and friends meanwhile to satisfy the security and privacy requirements. We are also motivated to design more efficient schemes to address this issue based on AdHocSign by leveraging the result of LA^3 to drive down the computational overhead.

APPENDIX A. Security Proof for LA^3 Scheme

A.1 Proof of Theorem 6.4.2 (Non-frameability)

Proof We are to show that, if there is a t-time adversary $\mathcal{A}$ winning the non-frameability game, an algorithm $\mathcal{B}$ can be constructed to solve the q-SDH problem in time $\Theta(1) \cdot t$. Suppose $\mathcal{B}$ is given the following (q+1)-tuple: $(\tilde{g},\ \tilde{g}^{\gamma},\ \cdots,\ \tilde{g}^{\gamma^q}) \in \mathbb{G}^{q+1}$. Using the algorithm adopted in the proof of Lemma 3.2 in [139], a certain $g \in \mathbb{G}$ and the following q SDH pairs can be obtained:

$$(x_1, A_1), \cdots, (x_q, A_q), \tag{A.1}$$

where each x_i, $i \in \{1, \cdots, q\}$, is randomly picked from $\mathbb{Z}_p$ and $A_i = g^{1/(\gamma + x_i)}$. Then, $\mathcal{B}$ is constructed to act as the challenger and play with $\mathcal{A}$ in the non-frameability game as follows.

Phase I: Initialization. $\mathcal{B}$ initializes the system and the verifier, and thus obtains the set of system-wide parameters, i.e., $\mathbb{SS} = \{k_1, k_2, d, l, C(x)\}$, and the set of parameters for the verifier, i.e., $\mathbb{SV} = \{k_1, k_2, l, C(x), \hat{d}\}$. Let n be the total number of provers. $w \in \{1, \cdots, n\}$ is randomly determined. For each prover u ($u \in \{1, \cdots, n\} \setminus \{w\}$), $\mathcal{B}$ initializes prover u and obtains the set of parameters for it, i.e., $\mathbb{SP}_u = \{\hat{k}_{u,0}, \hat{k}_{u,1}, s_u, B_u(x), \hat{e}_u, \hat{F}_u(x)\}$. To initialize prover w, $\mathcal{B}$ picks s_w and q distinct numbers randomly from $\mathbb{Z}_p$. Let the q numbers be denoted as $r''_1, \cdots, r''_q$. A set Ω is created to contain 3-tuples $(*, r''_i, x_i)$ ($i = 1, \cdots, q$) where $*$ represents an empty placeholder and each x_i is a part of the SDH pair (x_i, A_i).

Phase II: Queries and responses. $\mathcal{B}$ responds to queries issued by $\mathcal{A}$ as follows.

- *Corruption of the verifier:* When $\mathcal{A}$ issues a corruption query on the verifier, $\mathcal{B}$ returns $\mathbb{SV}$.
- *Corruption of a prover u:* $\mathcal{A}$ can issue a corruption query on a prover u. If $u = w$, failure is declared and the game exits. Otherwise, $\mathcal{B}$ responds with $\mathbb{SP}_u$.

- *Authentication for prover v:* $\mathcal{A}$ can issue an authentication query on a prover v. If q authentication queries have been served on prover v, failure is declared and the game exits. If $v \neq w$, $\mathcal{B}$ responds according to the authentication protocol in Section 6.3.4. If $v = w$ and the pending query is the i-th ($i \leq q$) authentication query on w, $\mathcal{B}$ responds as follows. The first challenge provided by the adversary is denoted as $\tilde{c}_1 = \{r'_i\}$. In Ω, the 3-tuple $(*, r''_i, x_i)$ is replaced with (r'_i, r''_i, x_i). $b_{w,i}$, α and ξ are picked from $\mathbb{Z}_p$ randomly, and $b_{w,i} = (\gamma + x_i)\alpha\beta$ is assumed. Hence, $g^{1/(\gamma+x_i)} = g^{\frac{\alpha\beta}{b_{w,i}}}$, and therefore $g^{\beta} = (g^{1/(\gamma+x_i)})^{b_{w,i}/\alpha}$. Then, the first response generated by $\mathcal{B}$ is $\tilde{r}_1$ as $\{r''_i, r_i, a_{w,1,i}, a_{w,2,i}, b_{w,i}, \hat{k}_{w,0,i}, \hat{k}_{w,1,i}, \hat{e}_{w,i}, \hat{F}_{w,i}\}$, where $\hat{e}_{w,i}$ is randomly picked from $\mathbb{G}$, $a_{w,1,i} = 2\alpha + \xi - 1$, $a_{w,2,i} = \alpha(s_w + 1) + \xi - 1$, $\hat{k}_{w,0,i} = g^{\beta}$, $\hat{k}_{w,1,i} = \hat{k}_{w,0,i}^{k_1}$, and $\hat{F}_{w,i} = [\hat{k}_{w,0,i}^{\alpha(1+k_1+k_2 s_w)C(x_i)} \hat{d}^{b_{w,r}} \hat{e}_{w,i}^{l}]^{-1}$. In response to the second challenge $\tilde{c}_2 = \{\hat{k}_{w,4,i}\}$ from $\mathcal{A}$, $\mathcal{B}$ returns $\tilde{r}_2 = \{\hat{k}_{w,5,i} = \hat{k}_{w,4,i}^{\xi}\}$.

- *Hash function $h()$:* When $\mathcal{A}$ queries $h(r_1, r_2)$, $\mathcal{B}$ searches Ω to find if there is a 3-tuple (r'_i, r''_i, x_i) such that $r'_i = r_1$ and $r''_i = r_2$. If a match is found, x_i is returned. Otherwise, a number is randomly picked from $\mathbb{Z}_p \setminus \{x_1, \cdots, x_q\}$ and returned. Meanwhile, consistency is maintained; that is, hashing on the same pair of (r_1, r_2) always results in the same value.

Phase III: Adversary's Output. Finally, $\mathcal{A}$ outputs the following transaction transcript: $\tilde{c}_1 = \{r'_1\}$, $\tilde{r}_1 = \{r'_2, a'_1, a'_2, b', \hat{k}'_0, \hat{k}'_1, \hat{e}', \hat{F}'\}$, $\tilde{c}_2 = \{\hat{k}'_4\}$, $\tilde{r}_2 = \{\hat{k}'_5\}$.

If $\mathcal{A}$ wins the game, the transcript can trace to a prover w'. As w is randomly placed among the n provers, $w' = w$ with probability $1/n$. Note that, the game will not fail if $w' = w$. In this case, a SDH pair $(x', g^{1/(\gamma+x')})$ that is different from any pair in Eq. (A.1) can be derived, where $x' = h(r'_1, r'_2)$ and $(\hat{k}'_0)^{(a'_2 - a'_1)/[(s_w - 1)b']} = g^{1/(\gamma+x')}$. Hence, the q-SDH problem is solved.

Therefore, if the q-SDH problem cannot be solved with probability greater than ϵ in time t, the probability for $\mathcal{A}$ to win the Non-frameability Game cannot be greater than $n\epsilon$ in time $\Theta(1) \cdot t$, where the total number of authentication queries towards each prover no more than q.

A.2 Proof of Lemma 6.4.1

Proof This proof is to show that, if there is an adversary $\mathcal{A}$ that can find out $\hat{d}$ in time t through the traceability game, then an algorithm $\mathcal{B}$ can be constructed to solve the DDH problem in time $\Theta(1) \cdot t$.

Assume $\mathcal{B}$ is provided with g, g^a, g^b and g^c, and is asked to find out if $c = ab$. $\mathcal{B}$ as the challenger plays with $\mathcal{A}$ in the traceability game as follows.

- **Phase I: Initialization Phase.** $\mathcal{B}$ randomly picks r_0, l' and k_1 from $\mathbb{Z}_p \setminus \{0\}$ and let $k_2 = 1 - k_1$. Then, it initializes each prover i ($i = 1, \cdots, n$ where n is the number of provers) as in the following. It randomly picks r_1, b_u and s_u from $\mathbb{Z}_p \setminus \{0\}$. Let

$$r_2 = 1 - r_1, B_u(x) = x + b_u, \hat{k}_{u,0} = [g^{r_1}(g^b)^{r_2}]^{1/(k_1+k_2 s_u+1)},$$

$$\hat{f}_{u,1} = [(g^a)^{r_1}(g^c)^{r_2}]^{-1}, \hat{f}_{u,0} = \hat{f}_{u,1}^{r_0}, \hat{e}_u = g^{l'(b_u - r_0)}.$$

  With the above assignments, if

$$c_1 = (c-a)/(b-1), c_0 = r_0 c_1, \hat{d} = g^{(ba-c)/(b-1)}, l = (ba-c)/[(1-b)l'], \tag{A.2}$$

  then it holds that $\hat{k}_{u,0}^{(k_1+k_2 s_u+1)c_1}\hat{d}\hat{f}_{u,1} = g^0$ and $\hat{k}_{u,0}^{(k_1+k_2 s_u+1)c_0}\hat{d}^{b_u}\hat{e}_u^l\hat{f}_{u,0} = g^0$.

- **Phase II: Queries and Response**. For a corruption query on prover u, $\mathbb{SP}_u$ is returned. For an authentication query on prover v, the prover and verifier exchange messages as specified in Section 6.3.4.

- **Phase III: Adversary's output**. The adversary outputs $\hat{d}'$ which is a guess of $\hat{d}$.

  In response, $\mathcal{B}$ returns $c = ab$ if $\hat{d}' = g^0$, and it returns $c \neq ab$ otherwise. This is because, $c = ba$ if $\hat{d} = \hat{d}' = g^0$, according to Eq. (A.2).

Obviously, if $\hat{d}' = \hat{d}$ with a probability greater than ϵ, then $\mathcal{B}$ can solve the DDH problem also with a probability greater than ϵ. Hence, if the DDH problem is (t, ϵ)-hard, then the probability for $\mathcal{A}$ to find out $\hat{d}'$ cannot be greater than ϵ.

A.3 Proof of Lemma 6.4.2

Proof This proof is to show that, if there is t-time algorithm $\mathcal{A}$ solving the extended LRSW problem, an algorithm $\mathcal{B}$ can be constructed to solve the LRSW problem in time $\Theta(1)\cdot t$. Suppose $\mathcal{B}$ is provided with g, g^x, $g^y \in \mathbb{G}$ (where x,$y \in \mathbb{Z}_p$) and an oracle O that can answer queries as specified in the definition of the LRSW problem. $\mathcal{B}$ provides to $\mathcal{A}$ with

$$g, g^k = g^y, g^{c_1} = g^x, g^{c_0} = (g^x)^\alpha,$$

where α is randomly picked from $\mathbb{Z}_p$. Then, $\mathcal{B}$ answers oracle queries from $\mathcal{A}$ as follows. When $\mathcal{A}$ queries oracle with input s, $\mathcal{B}$ queries oracle O with input $s' = (s-1)/2$ and obtains

$$g', (g')^y, (g')^{s'xy+x} = (g')^{(s-1)xy/2+x}.$$

Then $\mathcal{B}$ computes and returns to $\mathcal{A}$ the following:

$$g', (g')^k = (g')^y, (g')^{[k(s-1)+2]c_1} = (g')^{x[y(s-1)+2]} = [(g')^{s'xy+x}]^2,$$

$$(g')^{[k(s-1)+2]c_0} = [(g')^{[k(s-1)+2]c_1}]^\alpha.$$

At the end of queries, suppose $\mathcal{A}$ solves the extended LRSW problem by returning

$$g'', t, r, (g'')^k, A = (g'')^{[k(t-1)+2](c_1 r + c_0)}.$$

Then, algorithm $\mathcal{B}$ solves the LRSW problem as follows. It lets $t' = (t-1)/2$ and returns

$$g'',\ t',\ (g'')^y = (g'')^k, (g'')^{t'xy+x} = A^{1/[2(r+\alpha)]}.$$

A.4 Proof of Theorem 6.4.3 (Traceability)

Proof This proof is to show that, if there is t-time adversary $\mathcal{A}$ winning the traceability game and the CDH problem is (t, ϵ)-hard, then with probability $1-\epsilon$ either the DDH problem can be solved or an algorithm $\mathcal{B}$ can be constructed to solve the extended LRSW problem in $\Theta(1) \cdot t$ time.

Suppose algorithm $\mathcal{B}$ is given the extended LRSW problem: g, g^{k_2}, g^{c_1}, $g^{c_0} \in \mathbb{G}$ (k_2, c_1, $c_0 \in \mathbb{Z}_p$), and Oracle O that can answer queries specified in the extended LRSW problem, it acts as the challenger in the traceability game.

- **Phase I: Initialization.** $\mathcal{B}$ randomly selects $\hat{d}$ from $\mathbb{G}$ and l from $\mathbb{Z}_p$.
- **Phase II: Queries and Responses.** Specified as the traceability game, adversary $\mathcal{A}$ may issue corruption query and authentication query on prover u. When adversary $\mathcal{A}$ issues a corruption query on prover u, $\mathcal{B}$ responds as follows.
  - $\mathcal{B}$ randomly selects s_u from $\mathbb{Z}_p$. It calls the Oracle O provided by the extended LRSW problem on input s_u, let $k_1 = 1 - k_2$, $\mathcal{B}$ obtains:

$$\hat{k}_{u,0} = g', \hat{k}_{u,2} = (g')^{k_2}, \hat{k}_{u,1} = g'/(g')^{k_2} = (g')^{k_1},$$

$$g'_1 = (g')^{[k_2(s_u-1)+2]c_1} = (g')^{(k_1+k_2s_u+1)c_1},$$

$$g'_0 = (g')^{[k_2(s_u-1)+2]c_0} = (g')^{(k_1+k_2s_u+1)c_0}.$$

- $\mathcal{B}$ randomly selects $B_u(x) = x + b_u$, where $b_u \in \mathbb{Z}_p$.
- $\hat{e}_u$ is randomly picked from $\mathbb{G}$.
- $\hat{F}_u(x) = \hat{f}_{u,1}^x \hat{f}_{u,0}$, where $\hat{f}_{u,1} = [g'_1\hat{d}]^{-1}$ and $\hat{f}_{u,0} = [g'_0\hat{d}^{b_u}]^{-1}/\hat{e}_u^l$.
- Finally, $\mathbb{SP}_u = \{\hat{k}_{u,0}, \hat{k}_{u,1}, s_u, B_u(x), \hat{e}_u, \hat{F}_u(x)\}$ is returned to $\mathcal{A}$.

When adversary $\mathcal{A}$ issues an authentication query on prover u, $\mathcal{A}$ and $\mathcal{B}$ interacts based on the authentication protocol specified in Section 6.3.4.

- **Phase III: Adversary's Response.** Adversary $\mathcal{A}$ outputs the following transaction transcript: $\tilde{c}_1 = \{r'_1\}$, $\tilde{r}_1 = \{r'_2, r, a_{w,1,r}, a_{w,2,r}, \hat{k}_{w,0,r}, \hat{k}_{w,1,r}, B_{w,r}, \hat{e}_{w,r}, \hat{F}_{w,r}\}$, $\tilde{c}_2 = \{\hat{k}_{w,4,r}\}$, $\tilde{r}_2 = \{\hat{k}_{w,5,r}\}$.

If the adversary $\mathcal{A}$ wins, with the probability of at least $(1-\epsilon)$ it should be able to produce r'_2, r, $a_{w,1,r}$, $a_{w,2,r}$, $\hat{k}_{w,0,r}$, $\hat{k}_{w,1,r}$, $\hat{k}_{w,2,r}$, $B_{w,r}$, r'_3, $\hat{k}_{w,5,r}$, $\hat{e}_{w,r}$, and $\hat{F}_{w,r}$ for a certain r'_1 proposed by $\mathcal{B}$ and $r = h(r'_1, r'_2)$, such that

$$(\hat{k}_{w,0,r}\hat{k}_{w,1,r}^{a_{w,1,r}}\hat{k}_{w,2,r}^{a_{w,2,r}}\hat{k}_{w,5,r}^{-r'_3})^{C(r)}\hat{d}^{B_{w,r}}\hat{e}_{w,r}^l\hat{F}_{w,r} = g^0.$$

Note: due to the assumption that the CDH problem is (t,ϵ)-hard, $\mathcal{A}$ should not know ξ with probability at least $(1-\epsilon)$ when issuing authentication query on prover u. Otherwise, in the proposed authentication protocol, $\hat{k}_{u,4,r} = (\hat{k}_{u,1,r}\hat{k}_{u,2,r})^{1/r'_3}$, where r'_3 could be any arbitrary element picked from $\mathbb{Z}_p$, and $\mathcal{A}$ should return a correct $\hat{k}_{u,4,r}^{\xi}$ with a probability at least ϵ for a randomly chosen r'_3 with less than t time, which contradicts the hardness assumption of the CDH problem.

Upon receiving the information, $\mathcal{B}$ performs the following operations. It computes $\alpha = (a_{w,1,r} - \xi + 1)/2$ and $s' = (a_{w,2,r} - \xi + 1)/\alpha - 1$. Then, we consider two cases:

- Case I. If s' is different from any s_u that has been queried before, $\mathcal{B}$ computes

$$g'' = \hat{k}_{w,0,r}, (g'')^{k_1} = \hat{k}_{w,1,r}, (g'')^{k_2} = \hat{k}_{w,2,r},$$

$$(g'')^{(k_2(s'-1)+2)(c_1r'+c_0)} = (g'')^{(k_1+k_2s'+1)(c_1r'+c_0)} = \frac{1}{\hat{d}^{B_{w,r}}\hat{e}_{w,r}^l\hat{F}_{w,r}}.$$

Then, it outputs g'', s', r, $(g'')^{k_1}$, $(g'')^{k_2}$, and $(g'')^{(k_2(s'-1)+2)(c_1r'+c_0)}$. That is, $\mathcal{B}$ solves the extended LRSW problem. Due to Lemma 6.4.2, the LRSW problem can thus be solved.

- Case II. If s' is the same as a certain s_u that has been queried before, as $\mathcal{A}$ wins (i.e., the transcript cannot trace to prover u), it should hold that $B_{w,r} \neq B_{u,r}$. Meanwhile, as $s' = s_u$, it holds that $\hat{d}^{B_{w,r}}\hat{e}^l_{w,r}\hat{F}_{w,r} = \hat{d}^{B_{u,r}}\hat{e}^l_{u,r}\hat{F}_{u,r}$. Hence, $\hat{d}$ can be computed as $\hat{d} = [\hat{e}_{u,r}\hat{F}_{u,r}/(\hat{e}_{w,r}\hat{F}_{w,r})]^{B_{w,r}-B_{u,r}}$. That is, $\mathcal{A}$ should have known $\hat{d}$. However, according to Lemma 6.4.1, $\mathcal{A}$ cannot find $\hat{d}$ unless the DDH problem can be solved. Hence, the DDH problem can be solved.

The above two cases show that, if the CDH problem is (t, ϵ)-hard and A wins the traceability game, then either the DDH or the LRSW problem can be solved with probability at least $1 - \epsilon$. Now, if the CDH, DDH problem and LRSW problems are all (t, ϵ)-hard, and $\mathcal{A}$ is assumed to win the traceability game with probability ϵ' in time $\Theta(1) \cdot t$, then $(1 - \epsilon)\epsilon' \leq \epsilon$. That is, $\epsilon' \leq \frac{\epsilon}{1-\epsilon}$. If $\epsilon \leq 0.5$, it holds that $\epsilon' \leq \frac{\epsilon}{1-\epsilon} \leq \frac{\epsilon}{0.5} = 2\epsilon$; otherwise (i.e., $\epsilon > 0.5$), $\epsilon' \leq 1 < 2\epsilon$. Therefore, $\epsilon' \leq 2\epsilon$. That is, the LA^3 scheme is $(t', 2\epsilon)$-hard where $t' = \Theta(1) \cdot t$.

A.5 Proof of Theorem 6.4.4 (Selfless Anonymity)

Proof The proof is to show that, if there is adversary algorithm $\mathcal{A}$ winning the selfless anonymity game, an algorithm $\mathcal{B}$ can be constructed to solve the q-DDHI problem.

Suppose $\mathcal{B}$ is provided with $(g', (g')^{\gamma}, (g')^{\gamma^2}, \cdots, (g')^{\gamma^q})$ and $(g')^{1/(\gamma+y)}$ for some unknown $\gamma \in \mathbb{Z}_p$, and is requested to determine if $y = 0$. Using the algorithm presented in the proof of Lemma 3.2 in [139], $\mathcal{B}$ can obtain a certain $g \in \mathbb{G}$, $w = g^{\gamma}$ and q valid SDH pairs $(x_i, g^{1/(\gamma+x_i)})$ $(i = 1, \cdots, q)$ from $(g', (g')^{\gamma}, (g')^{\gamma^2}, \cdots, (g')^{\gamma^q})$ such that $e(g^{1/(\gamma+x_i)}, wg^{x_i}) = e(g, g)$. A SDH pair (x, A), where $x \in \mathbb{Z}_p, A \in \mathbb{G}$ is valid if $e(A, wg^x) = e(g, g)$. Similarly, $\mathcal{B}$ can also obtain g'' from $(g')^{1/(\gamma+y)}$ such that $g'' = g^{1/\gamma}$ if and only if $y = 0$. More specifically, the latter is accomplished as follows. Let f be the univariate polynomial defined as $f(\Gamma) = \prod_{i=1}^{q}(\Gamma + x_i)$.

Expand f and write $f(\Gamma)$ as $\sum_{i=0}^{q} f_i\Gamma^i$, where $f_i \in \mathbb{Z}_p$ are the coefficients of the polynomial f. Suppose $g = (g')^{\theta f(\gamma)}$, where $\theta \in \mathbb{Z}_p$ is randomly selected in the aforementioned course of generating

g and the q SDH pairs. Then, $\mathcal{B}$ computes

$$g'' = \left(\frac{\left((g')^{1/(\gamma+y)}\right)^{f_0}}{\prod_{i=0}^{q-1} \left((g')^{\gamma^i}\right)^{-f_{i+1}}} \right)^{\theta}.$$

Note that, if $y = 0$, g'' can be rewrite as:

$$\begin{aligned} g'' &= \left(\frac{\left((g')^{1/(\gamma)}\right)^{f_0}}{\prod_{i=0}^{q-1} \left((g')^{\gamma^i}\right)^{-f_{i+1}}} \right)^{\theta} = \left(\frac{\left((g')^{1/(\gamma)}\right)^{f_0}}{(g')^{-\sum_{i=0}^{q-1} f_{i+1}\gamma^i}} \right)^{\theta} \\ &= \left((g')^{\sum_{i=0}^{q} f_i\gamma^i}\right)^{\theta/\gamma} = \left((g')^{\theta f(\gamma)}\right)^{1/\gamma} = g^{1/\gamma} \end{aligned}$$

In the rest of the proof, we are to show that if $\mathcal{A}$ can win the selfless anonymity game with probability $0.5+\epsilon$, then whether $g'' = g^{1/\gamma}$ (equivalent to whether $y = 0$) can be determined by $\mathcal{B}$ with probability $0.5+\epsilon/n^2$ where n is the number of provers. $\mathcal{B}$ plays with $\mathcal{A}$ in the selfless anonymity game as follows.

Phase I: Initialization. $\mathcal{B}$ initializes the system and the verifier, and thus obtains the set of system-wide parameters, i.e., $\mathbb{SS} = \{k_1, k_2, d, l, C(x)\}$, and the set of parameters for the verifier, i.e., $\mathbb{SV} = \{k_1, k_2, l, C(x), \hat{d}\}$. Let the number of provers be n. $\mathcal{B}$ randomly picks i_0 and i_1 from $\{1, \cdots, n\}$. For each prover $u \notin \{i_0, i_1\}$, $\mathcal{B}$ initializes prover u and obtains the set of parameters for it, i.e., $\mathbb{SP}_u = \{\hat{k}_{u,0}, \hat{k}_{u,1}, s_u, B_u(x), \hat{e}_u, \hat{F}_u(x)\}$. For provers i_0 and i_1, $\mathcal{B}$ randomly pick s_{i_0}, s_{i_1}, z_{i_0} and z_{i_1} from $\mathbb{Z}_p$. It also creates a set Ω which initially contains 3-tuples: $(*, *, x_{2j-1} + z_{i_0})$ and $(*, *, x_{2j} + z_{i_1})$ for every $j = 1, \cdots, \lfloor q/2 \rfloor$ and each $*$ represents an empty placeholder.

Phase II: Pre-Challenge Queries and Responses. The adversary can issue the following types of queries which are responded by the challenger.

- *Corruption of the verifier.* When $\mathcal{A}$ issues a corruption query on the verifier, $\mathcal{B}$ returns $\mathbb{SV}$.
- *Corruption of prover u.* If $\mathcal{A}$ requests to corrupt prover $u \in \{1, \cdots, n\} \setminus \{i_0, i_1\}$, $\mathcal{B}$ returns $\mathbb{SP}_u$. Otherwise, failure is declared and the game exits.
- *Authentication for prover v.* If the number of authentication queries on v has exceeded $\lfloor q/2 \rfloor$, failure is declared and the game exits. Otherwise, if $v \notin \{i_0, i_1\}$, $\mathcal{B}$ responds according to the authentication protocol in Section 6.3.4. If $v \in \{i_0, i_1\}$, letting the query be the j'-th query on v and the first challenge proposed by the adversary be $\tilde{c}_1 = \{r'_v\}$, $\mathcal{B}$ responds as follows. Let $j = 2j' - 1$ if $v = i_0$; or, $j = 2j'$ otherwise. Then $\mathcal{B}$ responds based on $(x_j + z_v, g^{1/(\gamma+x_j)})$,

which is a valid SDH pair regarding g and $\gamma - z_v$. Specifically, $\mathcal{B}$ randomly picks r''_v, $B_{v,r}$, α and ξ from $\mathbb{Z}_p$, $\hat{e}_{v,r}$ from $\mathbb{G}$, and sets $r = x_j + z_v$. Meanwhile, in Ω, the 3-tuple $(*, *, r)$ is replaced with (r'_v, r''_v, r). Then, it computes

$$\hat{k}_{v,0,r} = (g^{1/(\gamma+x_j)})^{B_{v,r}/\alpha}, \;\; \hat{k}_{v,1,r} = \hat{k}_{v,0,r}^{k_1}, \;\; a_{v,1,r} = 2\alpha + \xi - 1,$$
$$a_{v,2,r} = \alpha(s_v + 1) + \xi - 1, \;\; \hat{F}_{v,r} = [\hat{k}_{v,0,r}^{\alpha(k_1+k_2 s_v+1)C(r)} \hat{d}^{B_{v,r}} \hat{e}_{v,r}^{l}]^{-1}.$$

The above is returned as $\tilde{r}_1$. In response to $\tilde{c}_2 = \{\hat{k}_{v,4,r}\}$ provided by the adversary, $\mathcal{B}$ returns $\tilde{r}_2 = \{\hat{k}_{v,5,r} = \hat{k}_{v,4,r}^{\xi}\}$, where $\tilde{c}_2$ and $\tilde{r}_2$ are computed according to step 4 and step 5 in the authentication protocol described in Section 6.3.4 respectively.

- *Hash function $h()$:* When $\mathcal{A}$ queries $h(r_1, r_2)$, $\mathcal{B}$ searches Ω to find if there is a 3-tuple (r'_i, r''_i, x'_i) such that $r'_i = r_1$ and $r''_i = r_2$. If a match is found, x'_i is returned. Otherwise, a number is randomly picked from $\mathbb{Z}_p \setminus \{z_{i_0}, z_{i_1}, x_1 + z_{i_0}, x_2 + z_{i_1}, \cdots, x_{\lfloor q/2 \rfloor * 2 - 1} + z_{i_0}, x_{\lfloor q/2 \rfloor * 2} + z_{i_1}\}$ and returned. Meanwhile, consistency is maintained; hashing on the same pair of (r_1, r_2) always results in the same value.

Phase III: Challenge. The adversary selects two provers u_0 and u_1 that have not been queried. If $\{u_0, u_1\} \neq \{i_0, i_1\}$, failure is declared and the game exits. Otherwise, the challenger randomly picks x from 0 or 1, and presents an authentication transcript $\tilde{r}_1$ and $\tilde{r}_2$ in response to challenges $\tilde{c}_1$ and $\tilde{c}_2$ presented by the adversary, respectively, based on (z_{u_x}, g''). Specifically, in response to $\tilde{c}_1 = \{r'_{u_x}\}$, $\mathcal{B}$ picks r''_{u_x}, $B_{u_x,r}$, α and ξ from $\mathbb{Z}_p$, $\hat{e}_{u_x,r}$ from $\mathbb{G}$, and sets $r = z_{u_x}$. Meanwhile, 3-tuple (r'_{u_x}, r''_{u_x}, r) is injected into Ω. Next, it computes

$$\hat{k}_{u_x,0,r} = (g'')^{B_{u_x,r}/\alpha}, \;\; \hat{k}_{u_x,1,r} = \hat{k}_{u_x,0,r}^{k_1}, \;\; a_{u_x,1,r} = 2\alpha + \xi - 1,$$
$$a_{u_x,2,r} = \alpha(s_{u_x} + 1) + \xi - 1, \;\; \hat{F}_{u_x,r} = [\hat{k}_{u_x,0,r}^{\alpha(k_1+k_2 s_{u_x}+1)C(r)} \hat{d}^{B_{u_x,r}} \hat{e}_{u_x,r}^{l}]^{-1}.$$

The above is presented as $\tilde{r}_1$. In response to $\tilde{c}_2 = \{\hat{k}_{u_x,4,r}\}$, $\mathcal{B}$ presents $\tilde{r}_2 = \{\hat{k}_{u_x,4,r}^{\xi}\}$, where $\tilde{c}_2$ and $\tilde{r}_2$ are computed according to step 4 and step 5 in the authentication protocol described in Section 6.3.4 respectively.

Phase IV: Post-Challenge Queries and Responses. The same as Phase II except that corruption queries cannot be made on the selected provers.

Phase V: Adversary's Response. $\mathcal{A}$ outputs $x' \in \{0, 1\}$ which is a guess of x chosen by $\mathcal{B}$. If $x' = x$, $\mathcal{B}$ returns $y = 0$ with probability $1 - \epsilon/2$; otherwise, it returns $y \neq 0$.

The game may abort if the adversary makes a corruption query on prover i_0 or i_1 in Phase II, or it does not choose $\{u_0, u_1\} = \{i_0, i_1\}$ in Phase III. Suppose the adversary has corrupted k provers in Phase II, the probability that it does not corrupted i_0 or i_1 is C^k_{n-2}/C^k_n, and the probability that it chooses $\{u_0, u_1\} = \{i_0, i_1\}$ in Phase III is $1/C^2_{n-k}$. Hence, the probability for the game to succeed is

$$\frac{C^k_{n-2}}{C^k_n C^2_{n-k}} = \frac{2}{n(n-1)} > 2/n^2.$$

For each successful game, there are two cases: (i) When $y = 0$ is true, $g'' = g^{1/\gamma}$; hence, (z_{u_x}, g'') is a valid SDH pair regarding g and $\gamma - z_{u_x}$. If $\mathcal{A}$ wins with advantage ϵ, the probability for $x' = x$ is $0.5 + \epsilon$ and hence the probability for $\mathcal{B}$ to return $y = 0$ is $(0.5 + \epsilon)(1 - \epsilon/2) = 0.5 + \epsilon - 0.25\epsilon - 0.5\epsilon^2 > 0.5 + 0.25\epsilon$. That is, the advantage gained by $\mathcal{B}$ is at least 0.25ϵ. (ii) When $y = 0$ is false, (z_{u_x}, g'') is not a SDH pair regarding g and $\gamma - z_{u_x}$; that is, the response based on (z_{u_x}, g'') cannot trace to either i_0 or i_1. Hence $\mathcal{A}$ returns x' as 0 or 1 randomly. So, the probability for $x' = x$ is 0.5 and therefore the probability for $\mathcal{B}$ to return $y = 0$ is $0.5(1 - \epsilon/2) = 0.5 - 0.25\epsilon$. That is, the probability for $\mathcal{B}$ to return $y \neq 0$ is $0.5 + 0.25\epsilon$; the advantage gained by $\mathcal{B}$ is 0.25ϵ.

To summarize, if $\mathcal{A}$ has probability greater than $0.5 + \epsilon$ to win the selfless anonymity game in time t, then $\mathcal{B}$ has probability greater than $0.5 + 0.25\epsilon \cdot (2/n^2) = 0.5 + 0.5\epsilon/n^2$ to solve the q-DDHI problem in time $\Theta(1) \cdot t$. This is equivalent to that, if the q-DDHI problem cannot be solved with probability greater than $0.5 + \epsilon$ in time t, then the proposed scheme cannot win the selfless anonymity game with probability greater than $0.5 + \epsilon'$ in time t', where $\epsilon' = 2n^2\epsilon$ and $t' = \Theta(1) \cdot t$.

APPENDIX B. Security Proof for AdHocSign Scheme

B.1 Proof of Theorem 7.3.2

Proof This proof is to show: if there is a t-time algorithm $\mathcal{A}$ that wins the traceability game in the AdHocSign scheme for conjunction-only access structures (called *Game 1* hereafter), a t'-time algorithm $\mathcal{B}$ can be constructed to solve the q-SDH problem, where $t' = \Theta(1) \cdot (t + mq)$. Similar to the proof of traceability (i.e., Theorem 6.2) in Boneh and Shacham's group signature scheme [103], this proof also proceeds in three parts: (1) a framework through which $\mathcal{B}$ interacts with $\mathcal{A}$; (2) instantiation of the framework for different types of algorithm ($\mathcal{A}$); and (3) derivation the conclusion.

Interaction Framework $\mathcal{A}$ and $\mathcal{B}$ are allowed to interact with each other as follows.

- *Initialization*. $\mathcal{B}$ is given $\mathbb{Z}_p$, $\mathbb{G}_1$, $g \in \mathbb{G}_1$, bilinear mapping $E : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ and a q-SDH tuple. It creates n users and m attributes. For each user e, $\mathcal{B}$ also creates a tuple $(x_e, A_{e,1}, \cdots, A_{e,m})$ where x_e could be a known element of $\mathbb{Z}_p$ or placeholder (denoted as $*$), and each $A_{e,a}$ could be a known element of $\mathbb{G}_1$ or placeholder (denoted as $*$). As to be elaborated later, users and attributes are initialized in different ways for different types of breaker algorithm.

- *Hash Queries*. $\mathcal{A}$ can query hash functions. $\mathcal{B}$ responds with random values while maintaining consistency.

- *Corruptions (i.e., Private Key Queries)*. $\mathcal{A}$ can determine the set of attributes $\{ a_{e,k} \mid k = 1, \cdots, n_e \}$, which is a subset of the attributes created by $\mathcal{B}$, for a certain user e, and then request $\mathcal{B}$ for the private key gsk_e of e. If $\mathcal{B}$ cannot provide the requested private key (i.e., $x_e = *$ or $A_{e,a_{e,k}} = *$ for some $k \in \{1, \cdots, n_e\}$), failure is declared and algorithm $\mathcal{B}$ exits. Otherwise, $\mathcal{B}$ computes and responds with the requested gsk_e.

- *Access Structure Initializations*. $\mathcal{A}$ can define an access structure T and then request $\mathcal{B}$ for the public key gpk_T regarding T. $\mathcal{B}$ computes and responds with the requested public key according to the *CO_AccessStructureInit* algorithm.

- *Signature queries*. $\mathcal{A}$ can request $\mathcal{B}$ for a user e's signature $\sigma_{M,T}$ on message M regarding an access structure T. Here, initialization of T should have been requested by $\mathcal{A}$ before. $\mathcal{B}$ responds as follows: In the first step, if $A_{e,a} \neq *$ for each attribute a present in T, it computes $\hat{A}_T$ as in the CO_Sign algorithm and then continues with the second step; otherwise, $NULL$ is returned. In the second step, if $x_e \neq *$, $\sigma_{M,T} = BSG\_Sign(\{ g_T, w_T\}, \{\hat{A}_{e,T}, x_e\}, M)$ is computed and returned; otherwise (i.e., $x_e = *$), the following is executed to compute the signature.

  1. A nonce r is picked from $\mathbb{Z}_p$ uniformly at random, and $(u, v) = H_0(gpk, M, r)$ is queried, where H_0 is a hash function.
  2. α is picked from $\mathbb{Z}_p$ uniformly at random, $T_1 \leftarrow u^\alpha$ and $T_2 \leftarrow \hat{A}_{e,T} g_T^\alpha$ are set, and the Protocol 1 simulator (in [103]) is run with values (u, v, T_1, T_2).
  3. The simulator returns a transcript (u, v, T_1, T_2, R_1, R_2, R_3, c, s_α, s_x, s_δ), from which a group signature $\sigma_{M,T} = (r, T_1, T_2, c, s_\alpha, s_x, s_\delta)$ is derived.
  4. $\mathcal{B}$ patches the hash oracle at (M, r, T_1, T_2, R_1, R_2, R_3) to equal c. If this causes a collision, i.e., if it has previously set the oracle at this point to some other c', failure is declared. Otherwise, $\sigma_{M,T}$ is returned to $\mathcal{A}$.

- *Output*. Finally, if $\mathcal{A}$ succeeds, it outputs a forged signature $\sigma_{M,T} = (r, T_1, T_2, c, s_\alpha, s_x, s_\delta)$ on message M regarding access structure T. $\mathcal{B}$ applies the revocation algorithm, with revocation tokens $\bar{A}_T = \{\hat{A}_{e,T}|$ for each user e that owns attributes satisfying $T\}$ to determine which $\hat{A}^*$ is encoded in (T_1, T_2). For the forgery to be nontrivial, $\hat{A}^*$ cannot be on the revocation list RL^*; also, if $\hat{A}^* \in \bar{A}_T$, letting $\hat{A}^* = \hat{A}_{e^*,T}$, $\mathcal{A}$ should have not issued a corruption query for user e^* or issued a signature query on M regarding T for e^*.

  If $\hat{A}^*$ does not belong to $\bar{A}_T$, we output $\sigma_{M,T}$. If $\hat{A}^* \in \bar{A}_T$, letting $\hat{A}^* = \hat{A}_{e^*,T}$, $\mathcal{B}$ outputs $\sigma_{M,T}$ when $x_{e^*} = *$, or declares failure and exits otherwise.

As implied by the output phase, there are three types of successful forger $\mathcal{A}$.

- Type I forgers output a forgery $\sigma_{M,T}$ that encodes some $\hat{A}^* \notin \bar{A}_T$, but $\hat{A}^* \in \tilde{A}_T$ where $\tilde{A}_T = \{\hat{A}_{e,T} |$ for every e that has been corrupted by $\mathcal{A}\}$. That is, $\hat{A}^*$ is the same as $\hat{A}_{e,T}$ of some user e who does not own all the attributes appearing in T and whose private key has been queried by $\mathcal{A}$.

- Type II forgers output a forgery $\sigma_{M,T}$ that encodes an $\hat{A}^*$ that does not belong to either $\bar{A}_T$ or $\tilde{A}_T$. That is, the forger outputs a forgery signature produced by a user not created by $\mathcal{B}$.

- Type III forgers output a forgery $\sigma_{M,T}$ that encodes an identity $\hat{A}^*$ such that $\hat{A}^* = \hat{A}_{e^*}$ for some e^* that the forger has not issued corruption query.

Next, we instantiate the above framework for each of the three types of forger.

Instantiating the Framework for Type I Forger Against a (t, q_H, q_S, n, m, ϵ)-Type I forger $\mathcal{A}$, the initialization phase of the framework should be customized as follows.

Given q-SDH tuple (g_0, g_0^{γ}, $g_0^{\gamma^2}$, $\cdots$, $g_0^{\gamma^q}$), we can construct m q-SDH tuples denoted as

$$T_1; T_2; \cdots; T_m, \tag{B.1}$$

where each T_i for $i = 1, \cdots, m$ is

$$(g_0^{\lambda_i}), (g_0^{\lambda_i})^{\gamma}, (g_0^{\lambda_i})^{\gamma^2}, \cdots, (g_0^{\lambda_i})^{\gamma^q}, \tag{B.2}$$

$\lambda_1 = 1$ and each λ_i ($i = 2, \cdots, m$) is randomly picked from $\mathbb{Z}_p$.

$\mathcal{B}$ creates m attributes (denoted as $1, 2, \cdots, m$) and $q-1$ users (associated with numbers $x_1, x_2, \cdots, x_{q-1}$ respectively), where each x_i is picked from $\mathbb{Z}_p$ uniformly at random. Also, we pick x_q from $\mathbb{Z}_p$ uniformly at random.

Based on T_1 defined in Eq. (B.1), using the method adopted in the proof of Lemma 3.2 in [139], we can obtain the following $q-1$ SDH pairs:

$$(x_1, A_{1,1}), (x_2, A_{2,1}), \cdots, (x_{q-2}, A_{q-2,1}), (x_q, A_{q,1}),$$

which are associated with a certain $g_1 \in \mathbb{G}_1$. Based on each T_j for $j = 2, \cdots, m$, similarly, we can obtain the following $q-1$ SDH pairs:

$$(x_1, A_{1,j}), (x_2, A_{2,j}), \cdots, (x_{q-2}, A_{q-2,j}), (x_{q-1}, A_{q-1,j}),$$

each associated with certain $g_j \in \mathbb{G}_1$. In the later phases of Game, each $(x_i, A_{i,j})$ for $i = 1, \cdots, q-1$ and $j = 1, \cdots, m$ is used as the private key for user i regarding attribute j. Note that, the initialization phase has the computational complexity of $\Theta(1) \cdot (mq)$.

A (t, q_H, q_s, n, m, ϵ)-Type I forger has the probability of $\epsilon'_I = \frac{\epsilon}{2(q-1)}$ to successfully output a forgery signature $\sigma_{M,T}$ that traces to the user $q-1$ regarding a certain access structure T where attribute 1 appears. This is because x_i for $i = 1, \cdots, q-1$ are randomly picked and all users appear the same to $\mathcal{A}$. The chances for $\mathcal{A}$ to win by forging a signature regarding a T with or without attribute 1 are equal, and the chances to trace the signature to every user are equal as well. Without loss of generality, let $gpk_T = \langle T, \{r_1, \cdots, r_m\}, g_T, w_T \rangle$, where $g_T = \prod_{j=1}^{m} g_j^{r_j}$, and $w_T = g_T^{\gamma}$.

As demonstrated by the *Application of Forger* part of the proof of traceability in [103], with probability $(\epsilon'_I - 1/p)^2/(16q_H) = (\epsilon/2(q-1) - 1/p)^2/(16q_H)$, a SDH pair $(x_{q-1}, \hat{A}_{q-1,T})$ can be derived from $\sigma_{M,T}$. Since $\hat{A}_{q-1,T} = \prod_{j=1}^{m} A_{q-1,j}^{r_j}$, we can compute $A_{q-1,1} = (\hat{A}_{q-1,T} / \prod_{j=2}^{m} A_{q-1,j}^{r_j})^{1/r_1}$. Hence, we get a new SDH pair $(x_{q-1}, A_{q-1,1})$.

Therefore, using a $(t, q_H, q_s, n, m\epsilon)$-Type I forger, the probability to solve the q-SDH problem is $(\epsilon/2(q-1) - 1/p)^2/(16q_H)$ in time t' where $t' = \Theta(1) \cdot (t + mq)$.

Instantiating and Applying the Framework for Type II forger Against a (t, q_H, q_s, n, m, ϵ)-Type II forger $\mathcal{A}$, the initialization phase of the framework is instantiated as follows. $\mathcal{B}$ creates $q-1$ users and m attributes. $q-1$ numbers $x_1, \cdots, x_{q-1}$ are picked from $\mathbb{Z}_p$ uniformly at random to be associated with the $q-1$ users, respectively. Based on the given T_1 in Eq. (B.1), a certain g_1 can be obtained together with $q-1$ SDH pairs

$$(x_1, A_{1,1}), (x_2, A_{2,1}), \cdots, (x_{q-1}, A_{q-1,1}),$$

where $A_{i,1} = g_1^{\frac{1}{\gamma + x_i}}$ for each $i = 1, \cdots, q-1$. Furthermore, we pick $m-1$ numbers $\alpha_2, \cdots, \alpha_m$ from $\mathbb{Z}_p$ uniformly at random. Then, for each $j = 2, \cdots, m$, we can obtain $g_j = g_1^{\alpha_i}$ and $q-1$ SDH pairs

$$(x_1, A_{1,j}), (x_2, A_{2,j}), \cdots, (x_{q-1}, A_{q-1,j}),$$

where for each $i = 1, \cdots, q-1$, $A_{i,j} = A_{i,1}^{\alpha_i}$, and thus $A_{i,j} = g_j^{\frac{1}{\gamma + x_i}}$. In the later phases of the framework, each pair $(x_i, A_{i,j})$ is used as the private key of user i for attribute j.

A (t, q_H, q_s, n, m, ϵ)-Type II forger has the probability of $\epsilon'_{II} = \epsilon$ to successfully output a forgery signature $\sigma_{M,T}$ that traces to a user outside of the $q-1$ created users regarding a certain access structure T. Without loss of generality, let $gpk_T = \langle T, \{r_1, \cdots, r_m\}, g_T, w_T \rangle$, where $g_T = \prod_{j=1}^{m} g_j^{r_j}$ and $w_T = g_T^{\gamma}$.

As demonstrated by the *Application of Forger* part of the proof of traceability in [103], with probability $(\epsilon'_{II} - 1/p)^2/(16q_H) = (\epsilon - 1/p)^2/(16q_H)$, a SDH pair $(x_q, \hat{A}_{q,T})$ can be derived from $\sigma_{M,T}$, where $x_q \neq x_i$ for $i = 1, \cdots, q-1$. Since $\hat{A}_{q,T} = \prod_{j=1}^{m} A_{q,j}^{r_j} = \prod_{j=1}^{m} A_{q,1}^{r_j \cdot \alpha_j}$, where $\alpha_1 = 1$, we can compute $A_{q,1} = \hat{A}_{q,T}^{-\sum_{j=1}^{m} r_j \cdot \alpha_i}$. Hence, we get a new SDH pair $(x_q, A_{q,1})$.

Therefore, using a (t, q_H, q_s, n, m, ϵ)-Type II forger, the probability to solve the q-SDH problem is $(\epsilon - 1/p)^2/(16q_H)$ in time t' where $t' = \Theta(1) \cdot (t + mq)$.

Instantiating and applying the Framework for Type III forger Against a (t, q_H, q_s, n, m, ϵ)-Type III forger $\mathcal{A}$, the initialization phase of the framework is instantiated as follows. $\mathcal{B}$ first creates $q-1$ users and m attributes as in the case for Type II forger. Particularly, for each user $i \in \{1, \cdots, q-1\}$ and each attribute $j \in \{1, \cdots, m\}$, g_j can be obtained together with $(x_i, A_{i,j})$ as above. In addition, $A_{q,1}$ is picked from $\mathbb{G}_1$ uniformly at random, and $A_{q,j} = A_{q,1}^{\alpha_j}$ is constructed for each $j = 2, \cdots, m$. Then, another user q can be created, where each pair $(*, A_{q,j})$ is used as the private key of user q for attribute j.

A (t, q_H, q_s, n, m, ϵ)-Type III forger has the probability of $\epsilon'_{III} = \epsilon/q$ to successfully output a forgery signature $\sigma_{M,T}$ that traces to user q regarding a certain access structure T, because the user q has the same appearance as other $q-1$ users to $\mathcal{A}$. Without loss of generality, let $gpk_T = \langle T, \{r_1, \cdots, r_m\}, g_T, w_T \rangle$, where $g_T = \prod_{j=1}^{m} g_j^{r_j}$ and $w_T = g_T^{\gamma}$.

As demonstrated by the *Application of Forger* part of the proof of traceability in [103], with probability $(\epsilon'_{III} - 1/p)^2/(16q_H) = (\epsilon/q - 1/p)^2/(16q_H)$, pair $(x_q, \hat{A}_{q,T})$ can be derived from $\sigma_{M,T}$. Since $\hat{A}_{q,T} = \prod_{j=1}^{m} A_{q,j}^{r_j} = \prod_{j=1}^{m} A_{q,1}^{r_j \cdot \alpha_j}$, where $\alpha_1 = 1$, we can compute $A_{q,1} = \hat{A}_{q,T}^{-\sum_{j=1}^{m} r_j \cdot \alpha_j}$. Hence, we get a new SDH pair $(x_q, A_{q,1})$.

Therefore, using a $(t - mq, q_H, q_s, n, m, \epsilon)$-Type III forger, the probability to solve the mq-SDH problem is $(\epsilon/q - 1/p)^2/(16q_H)$ in time t' where $t' = \Theta(1) \cdot (t + mq)$.

Summary Consider the above three cases together, using a (t, q_H, q_s, n, m, ϵ) breaker algorithm of Game 1, the probability to solve a q-SDH problem is at least $(\epsilon/2(q-1) - 1/p)^2/(16q_H)$ within time $\Theta(1) \cdot (t + mq)$. Therefore, if the SDH is (q, t', ϵ')-hard on $\mathbb{G}_1$, the proposed signature scheme is $(t, q_H, q_s, n, m, \epsilon)$-traceable, where $n = q - 1$, $\epsilon = 8n\sqrt{\epsilon' q_H} + 2n/p$, and $t' = \Theta(1) \cdot (t + m \cdot q)$.

B.2 Proof of Theorem 7.3.3

Proof The proof is to show: if there is algorithm $\mathcal{A}$ (t, q_H, q_S, n, m, ϵ)-breaking the selfless anonymity of the AdHocSign scheme, algorithm $\mathcal{B}$ can be constructed to break the Decision Linear assumption in $\mathbb{G}_1$.

Specifically, $\mathcal{B}$ is given as input a 6-tuple (u_0, u_1, v, $h_0 = u_0^a$, $h_1 = u_1^b$, Z)$\in \mathbb{G}_1^6$ where u_0, u_1 and v are picked from $\mathbb{G}_1$ and a and b are picked from $\mathbb{Z}_p$, uniformly at random. $\mathcal{B}$ decides whether $Z = v^{a+b} \in \mathbb{G}_1$ or Z is randomly picked from $\mathbb{G}_1$, through the following interactions with $\mathcal{A}$:

- **Initialization**. $\mathcal{B}$ runs algorithm *CO_Setup* to get system parameters $\mathbb{Z}_p$, $\mathbb{G}_1$, $E : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, $g \in \mathbb{G}_1$ and $\gamma \in \mathbb{Z}_p$. It creates n users denoted as $1, \cdots, n$, from which two users e_0 and e_1 are randomly picked. For each user $e \in \{1, \cdots, n\} \setminus \{e_0, e_1\}$, x_e is picked from $\mathbb{Z}_p$ uniformly at random. It defines m attributes denoted as $1, \cdots, m$, and runs algorithm *CO_AttributeInit* to get secrets $\alpha_a \in \mathbb{Z}_p$ for each attribute $a \in \{1, \cdots, m\}$. For each user $e \in \{1, \cdots, n\} \setminus \{e_0, e_1\}$, its private key $gsk_e = \langle x_e, A_{e,1} = g^{\alpha_1/(\gamma + x_e)}, \cdots, A_{e,m} = g^{\alpha_m/(\gamma+x_e)} \rangle$. For user $\{e_0, e_1\}$, $\mathcal{B}$ randomly picks W from $\mathbb{G}_1$. The private keys gsk_{e_0} and gsk_{e_1} are defined as follows:

  $$gsk_{e_0} = \langle *, A_{e_0,1} = (ZW/v^a)^{\alpha_1}, \cdots, A_{e_0,m} = (ZW/v^a)^{\alpha_m} \rangle,$$

  and

  $$gsk_{e_1} = \langle *, A_{e_1,1} = (Wv^b)^{\alpha_1}, \cdots, A_{e_0,m} = (Wv^b)^{\alpha_m} \rangle.$$

  We emphasize that $\mathcal{B}$ does not know either $A_{e_0,i}$ or $A_{e_1,i}$, where $i = 1 \cdots, m$, since it does not know a or b. Note that, if $Z = v^{a+b}$ in the given input of Decision Linear problem, it follows that

  $$A_{e_0,i} = (ZW/v^a)^{\alpha_i} = (v^{a+b}W/v^a)^{\alpha_i} = (Wv^b)^{\alpha_i} = A_{e_1,i}.$$

  Hence if $Z = v^{a+b}$, user e_0 and e_1 have the same private key. But if Z is random in $\mathbb{G}_1$, then e_0 and e_1 have independent private keys.

- **Hash Queries**. Whenever $\mathcal{A}$ queries the hash functions, $\mathcal{B}$ responds with random values while ensuring consistency.

- **Pre-challenge Queries**. $\mathcal{A}$ can issue corruption, access structure initialization and signing queries. $\mathcal{B}$ responses as follows.

  - *Corruption*. If $\mathcal{A}$ issues a corruption query on user $e \notin \{e_0, e_1\}$, gsk_e is returned. Otherwise, failure is declared and $\mathcal{B}$ exits.

  - *Access Structure Initialization*. If $\mathcal{A}$ issues an access structure initialization query on access structure T, $\mathcal{B}$ calls *CO_AccessStructureInit* to get gpk_T and returns it.

  - *Signing*. If $\mathcal{A}$ issues a signing query on a message M regarding access structure T for user e, $\mathcal{B}$ first calls *CO_AccessStructureInit(T)* to get gpk_T. If $e \notin \{e_0, e_1\}$, $\mathcal{B}$ calls *CO_Sign(gpk_T, gsk_e, T)* to get signature σ and then returns it. Otherwise, suppose $gpk_T = \langle T, \{r_{a_T,1}, \cdots, r_{a_T,s}\}, g_T = g^{\sum_{i=1}^{s}(r_{a_T,i} \cdot \alpha_{a_T,i})}, w_T \rangle$. Let

$$\beta_T = \sum_{i=1}^{s}(r_{a_T,i} \cdot \alpha_{a_T,i}),$$

$$u'_0 = u_0^{\beta_T}, u'_1 = u_1^{\beta_T}, v' = v^{\beta_T}, W' = W^{\beta_T},$$

$$h'_0 = h_0^{\beta_T}, h'_1 = h_1^{\beta_T}, Z' = Z^{\beta_T}.$$

    $\mathcal{B}$ picks randomly s, t, l from $\mathbb{Z}_p$, and makes the following assignment:

    * If $e = e_0$,

$$T_1 \leftarrow h'_0(u'_0)^s, T_2 \leftarrow Z'W'(v')^s(h'_0)^t(u'_0)^{st}, \hat{u} \leftarrow (u'_0)^l, \hat{v} \leftarrow [v'(u'_0)^t]^l.$$

      Let $\alpha = (a+s)/l \in \mathbb{Z}_p$, then $T_1 = \hat{u}^\alpha$, $T_2 = \hat{A}_{e_0,T} \cdot \hat{v}^\alpha$, where $\hat{A}_{e_0,T} = \prod_{i=1}^{s} A_{e_0,a_{T,i}}^{r_{a_{T,i}}}$ because,

$$\begin{aligned}
T_1 &= h'_0(u'_0)^s = (u'_0)^{a+s} = [(u'_0)^l]^\alpha = \hat{u}^\alpha \\
T_2 &= Z'W'(v')^s(h'_0)^t(u'_0)^{st} = (Z'W'/(v')^a)(v')^{(a+s)}(u'_0)^{t(a+s)} \\
&= (Z'W'/(v')^a)(v'(u'_0)^t)^{(a+s)} = (ZW/v^a)^{\beta_T}\hat{v}^\alpha \\
&= \prod_{i=1}^{s} A_{e_0,a_{T,i}}^{r_{a_{T,i}}}\hat{v}^\alpha = \hat{A}_{e_0,T}\hat{v}^\alpha
\end{aligned}$$

$*$ If $e = e_1$,

$$T_1 \leftarrow h'_1(u'_1)^s, T_2 \leftarrow W'(h'_1)^t(u'_1)^{st}/(v')^s, \hat{u} \leftarrow (u'_1)^l, \hat{v} \leftarrow [(u'_1)^t/v']^l.$$

Let $\alpha = (b+s)/l \in \mathbb{Z}_p$, then $T_1 = \hat{u}^\alpha$ and $T_2 = \hat{A}_{e_1,T} \cdot \hat{v}^\alpha$, where $\hat{A}_{e_1,T} = \prod_{i=1}^{s} A_{e_1,a_{T,i}}^{r_{a_{T,i}}}$ because,

$$\begin{aligned} T_1 &= h'_1(u'_1)^s = (u'_1)^{b+s} = [(u'_1)^l]^\alpha = \hat{u}^\alpha \\ T_2 &= W'(h'_1)^t(u'_1)^{st}/(v')^s = W'(v')^b[(u'_1)^t/(v')]^b[(u'_1)^t/(v')]^s \\ &= W'(v')^b[(u'_1)^t/(v')]^{b+s} = (Wv^b)^{\beta_T}\hat{v}^\alpha \\ &= \prod_{i=1}^{s} A_{e_1,a_{T,i}}^{r_{a_{T,i}}}\hat{v}^\alpha = \hat{A}_{e_1,T}\hat{v}^\alpha \end{aligned}$$

Either way, $T_1 = \hat{u}^\alpha$ and $T_2 = \hat{A}_{e,T} \cdot \hat{v}^\alpha$ for some unknown random $\alpha \in \mathbb{Z}_p$. Using the same approach in the proof of selfless anonymity in [103], a signature σ on message M regarding T can be generated.

- **Challenge**. $\mathcal{A}$ outputs a message M, an access structure T, and two users e_0^* and e_1^* where it wishes to be challenged. If $\{e_0^*, e_1^*\} \neq \{e_0, e_1\}$, $\mathcal{B}$ declares failure and exits. Otherwise, $\mathcal{B}$ picks a random bit b from $\{0,1\}$ uniformly at random and generates a signature σ^* under user e_b's key for M regarding T using the same method used to respond to signing queries in the pre-challenge phase. It gives σ^* as the challenge to $\mathcal{A}$.

- **Restricted Queries**. $\mathcal{A}$ issues restricted queries. $\mathcal{B}$ responds as in the pre-challenged phase.

- **Output**. $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ for b. If $b = b'$ then $\mathcal{B}$ outputs 0 (indicating that Z is random in $\mathbb{G}_1$); otherwise $\mathcal{B}$ outputs 1 (indicating that $Z = v^{a+b}$).

According to the same reasoning as in the proof of selfless anonymity for the VLR group signature scheme [103], it follows that $\mathcal{B}$ can solve the Decision Linear problem with advantage at least $\frac{\epsilon}{2}(\frac{1}{n^2} - \frac{q_S q_H}{p})$.

Assuming the (t', ϵ') Decision Linear assumption holds in group $\mathbb{G}_1$ (i.e., no t'-time algorithm can solve the Decision Linear problem with advantage at least ϵ'), the ad hoc group signature scheme in $\mathbb{G}_1$ is (t, q_H, q_S, n, m, ϵ)-selflessly-anonymous, where $\epsilon' = \frac{\epsilon}{2}(\frac{1}{n^2} - \frac{q_S q_H}{p})$ and $t' = \Theta(1) \cdot (t + m \cdot n)$, considering the initialization phase may take $\Theta(mn)$ time.

B.3 Proof of Theorem 7.4.2

Proof This proof is to show: if there is a t-time algorithm $\mathcal{A}$ winning the traceability game in a system where there are n users and m attributes, each attribute is associated with N secret numbers, and access structures are general (i.e., each can be represented as conjunction-of-disjunction logical expressions of attributes), then a t'-time algorithm $\mathcal{B}$ can be constructed to win the traceability game in a system where there are n users and up to $m \cdot N + 1$ attributes, and access structures are conjunction-only logical expressions of attributes. Here, $t' = \Theta(1) \cdot (t + m \cdot N \cdot q)$. Hereafter, the traceability games mentioned above are called CD-Game (with conjunction-of-disjunction access structures) and CO-Game (with conjunction-only access structures), respectively.

Let $\mathcal{B}$ play with a challenger (denoted as $\mathcal{C}$) in the CO-Game, and meanwhile play with $\mathcal{A}$ in the CD-Game in order to leverage $\mathcal{A}$ to win the CO-Game. In the games, $\mathcal{B}$ interacts with $\mathcal{C}$ and $\mathcal{A}$ as follows:

- **Initialization**. At the end of the initialization phase of the CO-Game, $\mathcal{C}$ provides to $\mathcal{B}$ system parameters: $\mathbb{G}_1$, $g \in \mathbb{G}_1$, E, the number of users n (users are named as 1, 2, $\cdots$, n) and the number of attributes $m \cdot N + 1$ (attributes are named as 0, 1, 2, $\cdots$, $m \cdot N$). Based on the received parameters, $\mathcal{B}$ initializes the CD-Game as follows. It also uses $\mathbb{G}_1$, g and E as system parameters. Secret number $\xi \in \mathbb{Z}_p$ is chosen randomly. n users (called users 1, 2, $\cdots$, n) are created and the number of attributes is set to m (attributes are named as 1, $\cdots$, m). For each attribute $a \in \{1, \cdots, m\}$, two arrays (denoted as $A_a[1..N]$ and $R_a[1..N]$) each with N elements are created to be associated to the attribute, and each entry of the arrays is initialized to empty. Finally, U, the set of corrupted users, is initialized to $\emptyset$.

- **Queries**. In the CD-Game, $\mathcal{A}$ can make queries to $\mathcal{B}$. As reactions, $\mathcal{B}$ make queries to $\mathcal{C}$ and responds to $\mathcal{A}$ based on the responses from $\mathcal{C}$.

  - *Corruption*. When $\mathcal{A}$ in the CD-Game determines a set of attributes $\{a_{e,i} | i = 1, \cdots, n_e\}$ for user e and requests for the private key of e, $\mathcal{B}$ reacts by (1) making a corruption query for e at attribute 0 to $\mathcal{C}$ and thus obtaining x_e, and (2) returning to $\mathcal{A}$ the following private

key:

$$gsk_e = \langle x_e, \{A'_{e,a_{e,i},j} = g^{H_3(\xi,x_e,a_{e,i},j)} | i = 1, \cdots, n_e; j = 1, \cdots, N\}\rangle,$$

where $H_3(\cdot,\cdot,\cdot,\cdot)$ is a random oracle that outputs an element of $\mathbb{Z}_p$ on any input of four elements of $\mathbb{Z}_p$. (3) e is added to U.

– *Access Structure Initialization*. When $\mathcal{A}$ in the CD-Game requests for the public key gpk_T for a certain access structure $T = DT_1 \wedge \cdots \wedge DT_s$, where each $DT_i = a_{T,i,1} \vee \cdots \vee a_{T,i,s_i}$ ($a_{T,i,1} < \cdots < a_{T,i,s_i}$) is a disjunction-only access structure, $\mathcal{B}$ reacts as follows.

  * In the context of the CD-Game, for each DT_i appearing in T: Let $v_i = \sum_{j=1}^{s_i} 2^{a_{T,i,j}-1}$. For each attribute $a_{T,i,j}$ appearing in DT_i, its associated array $A_{a_{T,i,j}}$ is searched to locate v_i. If v_i is not found, v_i is put into an empty entry of the array; if no empty entry can be found, the games end and failure is declared. Now, let $a'_{T,i,j}$ be an index such that $A_{a_{T,i,j}}[a'_{T,i,j}] = v_i$. After the above steps have been performed for each $a_{T,i,j}$ of DT_i, let $a''_i = [(a_{T,i,1}) - 1] \cdot N + a'_{T,i,1}$.
  * In the CO-Game, $\mathcal{B}$ constructs a conjunction-only access structure $T' = a''_1 \wedge \cdots \wedge a''_s$. $\mathcal{B}$ requests $\mathcal{C}$ for public key $gpk'_{T'}$ regarding T', and suppose that the public key received is

$$gpk'_{T'} = \langle T', r_{T',1}, \cdots, r_{T',s}, g_{T'}, w_{T'}\rangle.$$

  * In the CD-Game, for every attribute $a_{T,i,j}$ appearing in each DT_i, $\mathcal{B}$ chooses a value for $R_{a_{T,i,j}}[a'_{T,i,j}]$ from $\mathbb{Z}_p \setminus \{0\}$ uniformly at random and computes $r_{T,i,j} = r_{T',i} \cdot R_{a_{T,i,j}}[a'_{T,i,j}]$. It also computes $h_{T,i,j} = H_2(\xi, a_{T,i,j}, a'_{T,i,j})$, where $H_2(\cdot,\cdot,\cdot)$ is a random oracle which outputs an element of $\mathbb{Z}_p$ on any input of three elements of $\mathbb{Z}_p$.

  Finally, the following public key is returned to $\mathcal{A}$:

$$gpk_T = \langle T, \{(r_{T,i,j}, h_{T,i,j}) | i = 1, \cdots, s; \text{ for each } i, j = 1, \cdots, s_i\}, g_{T'}, w_{T'}\rangle.$$

– *Signing*. Whenever $\mathcal{A}$ in CD-Game requests for the signature of user e on message M regarding a certain access structure T, $\mathcal{B}$ reacts in the following steps:

  * As in *Access Structure Initialization*, access structure T' in the CO-Game is obtained from the access structure T in the CD-Game.

  - ∗ In the CO-Game, $\mathcal{B}$ requests $\mathcal{C}$ for the signature of e on message M regarding T'. Upon receiving the signature σ, $\mathcal{B}$ returns the signature to $\mathcal{A}$ in the CD-Game.

- *Hash Queries*. In response to $H_1(e, a, a', h)$: If entry $A_a[a']$ is empty (i.e., the a'-th secret associated with attribute a has not been used in host preparation yet), an arbitrary value is returned. Otherwise:
  - ∗ Let $DT = a_1 \vee \cdots \vee a_s$ (where $a_1 < \cdots < a_s$) such that $A_a[a'] = \sum_{j=1}^{s} 2^{a_j - 1}$. Array A_{a_1} is searched to find out an index a'_1 such that $A_{a_1}[a'_1] = A_a[a']$. A compromise query for e with attribute a'_1 is made to $\mathcal{C}$.
  - ∗ Upon receiving the private key (x_e, A'), the actual a'-th private key for e on attribute a is computed as
$$A_{e,a,a'} = (A')^{(1/R_a[a'])}.$$
  - ∗ The hash value is returned as
$$A_{e,a,a'} \cdot (A'_{e,a,a'})^{-1} = A_{e,a,a'} \cdot g^{-H_3(\xi, x_e, a, a')}.$$

  In response to queries on other hash functions (i.e., H_0, H, H_1 and H_2), algorithm $\mathcal{B}$ returns random values while ensuring consistency.

- *Response*. If $\mathcal{A}$ in CD-Game outputs M^*, a certain access structure T, a set RL^* of revocation tokens and a signature σ^*, $\mathcal{B}$ in CO-Game also outputs M^*, access structure T' which is converted from T as in the handling of *Access Structure Initialization* queries, RL^* and σ^*.

According to the above strategy, if $\mathcal{A}$ wins the CD-Game, $\mathcal{B}$ must also win the CO-Game, because

- If σ^* is accepted as a valid signature on M^* regarding T in the system where CD-Game is played, the signature is also accepted as a valid signature on M^* regarding T' in the system where CO-Game is played, where T' is converted from T as in the handling of *access structure initialization* queries.

- If σ^* traces to some user e outside of $U \setminus RL^*$ in the CD-Game, it also traces to the same user e in the CO-Game.

- If the tracing fails in the CD-Game system (i.e., it cannot trace to any user), it cannot trace to any user either in the CO-Game because users in the two games are one-to-one correspondent.

- If $\mathcal{A}$ never makes a signing query at M^* for user e on access structure T in the CD-Game, $\mathcal{B}$ never makes a signing query at M^* for user e on access structure T' in the CO-Game, where T' is converted from T as in the handling of *access structure initialization* queries.

Also according to algorithm $\mathcal{B}$, algorithm $\mathcal{B}$ needs computational complexity of $O(m \cdot N)$ to handle each type of query issued by $\mathcal{A}$. Hence, if $\mathcal{A}$ needs time t, q_S signing queries and q_H queries on hash functions H_0 and H to win the CD-Game, then algorithm $\mathcal{B}$ needs time t', q_S signing queries and q_H queries on H_0 and H to win the CO-Game, where $t' = O(t \cdot m \cdot N)$. Therefore, if the AdHocSign scheme for conjunction-only access structures is $(t', q_S, q_H, n, m, \epsilon)$-traceable, then the AdHocSign scheme for general access structures is $(t, q_S, q_H, n, m, \epsilon)$-traceable.

BIBLIOGRAPHY

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E.Cayirci, "Wireless Sensor Networks: A Survey," in *Computer Networks*, vol. 38, no. 4, 2002.

[2] D. Ganesan, A. Cerpa, W. Ye, Y. Yu, J. Zhao, and D. Estrin, "Networking issues in wireless sensor networks," in *J. Parallel Distrib. Comput.*, vol. 64, no. 7, pp. 799–814, 2004.

[3] A. Parker, S. Reddy, T. Schmid, K. Chang, G. Saurabh, M. Srivastava, M. Hansen, J. Burke, D. Estrin, M. Allman, and V. Paxson, "Network system challenges in selective sharing and verification for personal social, and urban-scale sensing applications," in *HotNets*, 2006.

[4] S. Reddy, A. Parker, J. Hyman, J. Burke, D. Estrin, and M. Hansen, "Image browsing, processing, and clustering for participatory sensing: lessons from a dietsense prototype," in *EmNets*, 2007.

[5] K. Mayer and W. Fritsche, "IP-enabled wireless sensor networks and their integration into the internet," in *InterSense*, 2006.

[6] A. Sleman and R. Moeller, "Integration of wireless sensor network services into other home and industrial networks; using device profile for web services (dpws)," in *ICTTA*, 2008.

[7] M. H. Kido, C. W. Mundt, K. N. Montgomery, A. Asquith, D. W. Goodale, and K. Y. Kaneshiro, "Integration of wireless sensor networks into cyberinfrastructure for monitoring hawaiian mountain-to-sea environments," in *Environmental Management*, vol. 42, no. 4, 2008.

[8] V. Casola, A. Gaglione, and A. Mazzeo, "A reference architecture for sensor networks integration and management," in *Proceedings of the 3rd International Conference on GeoSensor Networks*, pp. 158–168, 2009.

[9] L. H. Beng, "Sensor cloud : Towards sensor-enabled cloud services." [Online]. Available: http://files.ngp.org.sg/GridAsia2009/presentations_slides/1-monday/rnd/1400-SensorCloud-NTU(LimHockBeng).pdf.

[10] Hassan, M. M., Song, B., and Huh, E.-N. A framework of sensor-cloud integration opportunities and challenges. in *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, pages 618–626, 2009.

[11] Melchor, J. and Fukuda, M. A design of flexible data channels for sensor-cloud integration. in *Systems Engineering (ICSEng), 21st International Conference on*, pages 251–256, 2011.

[12] M. Yuriyama, and T. Kushida, "Sensor-Cloud Infrastructure - Physical Sensor Management with Virtualized Sensors on Cloud Computing," in *Network-Based Information Systems (NBiS), 13th International Conference on*, 2010.

[13] R. Campbell, J. Al-Muhtadi, P. Naldurg, G. Sampemane, and M. D. Mickunas, Towards security and privacy for pervasive computing. in *Proceedings of the 2002 Mext-NSF-JSPS international conference on Software security: theories and systems*, 2002.

[14] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn, The rise of people-centric sensing. in *IEEE Internet Computing*, vol. 12, no. 4, 2008.

[15] Johnson, P., Kapadia, A., Kotz, D., and Triandopoulos, N. People-centric urban sensing: Security challenges for the new paradigm. in *Technical Report*, [Online]. Available: http://www.cs.dartmouth.edu/cms_file/SYS_techReport/439/TR2007-586.pdf, 2007.

[16] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for ad-hoc sensor networks," in *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, 2002.

[17] X. Tang and J. Xu, "Extending network lifetime for precision-constrained data aggregation in wireless sensor networks," in *IEEE INFOCOM*, 2006.

[18] K.-W. Fan, S. Liu, and P. Sinha, "On the potential of structure-free data aggregation in sensor networks," in *IEEE INFOCOM*, 2006.

[19] C. Intanagonwiwat, D. Estrin, R. Govindan and J. Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks," in *IEEE ICDCS*, 2002.

[20] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The impact of data aggregation in wireless sensor networks," in *IEEE ICDCSW*, 2002.

[21] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson, "Synopsis Diffusion for Robust Aggregation in Sensor Networks," in *ACM SenSys*, 2004.

[22] P. Jadia and A. Mathuria, "Efficient secure aggregation in sensor networks," in *High Performance Computing*, vol. 3296, 2004.

[23] C. Castelluccia, E. Mykletun, and G. Tsudik, "Efficient aggregation of encrypted data in wireless sensor networks," in *MobiQuitous*, 2005.

[24] J. Girao, D. Westhoff, and M. Schneider, "CDA: concealed data aggregation for reverse multicast traffic in wireless sensor networks," in *IEEE ICC*, 2005.

[25] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "PDA: Privacy-preserving data aggregation in wireless sensor networks," in *IEEE INFOCOM*, 2007.

[26] T. Feng, C. Wang, W. Zhang, and L. Ruan, "Confidentiality protection for distributed sensor data aggregation," in *IEEE INFOCOM*, 2008.

[27] W. Zhang, C. Wang, and T. Feng, "GP^{2}S: Generic privacy-preservation solutions for approximate aggregation of sensor data (concise contribution)," in *IEEE PerCom*, 2008.

[28] S. Roy, M. Conti, S. Setia, and S. Jajodia, "Securely computing an approximate median in wireless sensor networks," in *SecureComm*, 2008.

[29] E. Mykletun, J. Girao, and D. Westhoff, "Public key based cryptoschemes for data concealment in wireless sensor networks," in *IEEE ICC*, 2006.

[30] J. Horey, M. M. Groat, S. Forrest, and F. Esponda, "Anonymous data collection in sensor networks," in *MobiQuitous*, 2007.

[31] R. K. Ganti, N. Pham, Y.-E. Tsai, and T. F. Abdelzaher, "PoolView: stream privacy for grassroots participatory sensing," in *ACM SenSys*, 2008.

[32] H. Chan, A. Perrig, and D. Song, "Secure hierarchical in-network aggregation in sensor networks," in *ACM CCS*, 2006.

[33] Y. Yang, X. Wang, S. Zhu, and G. Cao, "SDAP:: a secure hop-by-hop data aggregation protocol for sensor networks," in *ACM MobiHoc*, 2006.

[34] K. B. Frikken and J. A. Dougherty, IV, "An efficient integrity-preserving scheme for hierarchical sensor aggregation," in *ACM WiSec*, 2008.

[35] E. De Cristofaro, J.-M. Bohli, and D. Westhoff, "FAIR: fuzzy-based aggregation providing in-network resilience for real-time wireless sensor networks," in *ACM WiSec*, 2009.

[36] W. Zhang, Y. Liu, S. Das, and P. De, "Secure data aggregation in wireless sensor networks: A watermark based authentication supportive approach," in *Pervasive and Mobile Computing*, vol. 4, no. 5, 2008.

[37] L. Hu and D. Evans, "Secure aggregation for wireless networks," in *SAINT*, 2003.

[38] P. Haghani, P. Papadimitratos, M. Poturalski, K. Aberer, and J. P. Hubaux, "Efficient and robust secure aggregation for sensor networks," in *NPSec*, 2007.

[39] G. Taban and V. D. Gligor, "Efficient handling of adversary attacks in aggregation applications," in *ESORICS*, 2008.

[40] W. He, X. Liu, H. Nguyen, and K. Nahrstedt, "A cluster-based protocol to enforce integrity and preserve privacy in data aggregation," in *IEEE ICDCSW*, 2009.

[41] W. He, H. Nguyen, X. Liu, K. Nahrstedt, and T. Abdelzaher, "*i*PDA: An integrity-protecting private data aggregation scheme for wireless sensor networks," in *IEEE MILCOM*, 2008.

[42] C. Wang, G. Wang, W. Zhang, and T. Feng, "Reconciling privacy preservation and intrusion detection in sensory data aggregation," in *IEEE INFOCOM*, 2011.

[43] P. Bartosz, D. Song, and A. Perrig, "SIA: Secure Information Aggregation in Sensor Networks," in *ACM SenSys*, 2003.

[44] Crossbow, "Telosb mote plateform." [Online]. Available: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf.

[45] J. Polastre, R. Szewczyk, C. Sharp, and D. Culler, "The mote revolution: Low power wireless sensor network devices," in *in Hot Chips 16: A Symposium on High Performance Chips*, 2004.

[46] A. Perrig, J. Stankvic, and D. Wagner, "Security in wireless sensor networks," in *Journal of Communications of the ACM*, vol. 47, 2004.

[47] R. Mavropodi, P. Kotzanikolaou, and C. Douligeris, "SecMR - a secure multipath routing protocol for ad hoc networks," in *Ad Hoc Networks*, vol. 5, no. 1, 2007.

[48] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," in *the First IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[49] V. Bhuse, A. Gupta, and L. Lilien, "DPDSN: Detection of packet-dropping attacks for wireless sensor networks," in *the Fourth Trusted Internet Workshop*, 2005.

[50] M. Kefayati, H. R. Rabiee, S. G. Miremadi, and A. Khonsari, "Misbehavior resilient multi-path data transmission in mobile ad-hoc networks," in *ACM SASN*, 2006.

[51] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *ACM MobiCom*, 2000.

[52] S. Lee and Y. Choi, "A resilient packet-forwarding scheme against maliciously packet-dropping nodes in sensor networks," in *ACM SASN*, 2006.

[53] T. H. Hai and E. N. Huh, "Detecting selective forwarding attacks in wireless sensor networks using two-hops neighbor knowledge," in *IEEE NCA*, 2008.

[54] F. Liu, X. Cheng, and D. Chen, "Insider Attacker Detection in Wireless Sensor Networks," in *IEEE INFOCOM*, 2007.

[55] S. Ganeriwal, L. K. Balzano, and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," in *ACM Transactions on Sensor Networks (TOSN)*, vol. 4, no. 3, 2008.

[56] W. Li, A. Joshi, and T. Finin, "Coping with node misbehaviors in ad hoc networks: A multi-dimensional trust management approach," in *IEEE Mobile Data Management*, 2010.

[57] P. Michiardi and R. Molva, "CORE: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *CMS*, 2002.

[58] S. Buchegger and J. Le Boudec, "Performance analysis of the confidant protocol," in *ACM MobiHoc*, 2002.

[59] I. Khalil and S. Bagchi, "Mispar: mitigating stealthy packet dropping in locally-monitored multihop wireless ad hoc networks," in *SecureComm*, 2008.

[60] K. Ioannis, T. Dimitriou, and F. C. Freiling, "Towards intrusion detection in wireless sensor networks," in *European Wireless Conference*, 2007.

[61] I. Krontiris, T. Giannetsos, and T. Dimitriou, "LIDeA: a distributed lightweight intrusion detection architecture for sensor networks," in *SecureComm*, 2008.

[62] A. Srinivasan, J. Teitelbaum, H. Liang, J. Wu, and M. Cardei, "Reputation and trust-based systems for ad hoc and sensor networks," in *Algorithms and Protocols for Wireless Ad Hoc and Sensor Networks,*. 2008.

[63] M. Just, E. Kranakis, and T. Wan, "Resisting malicious packet dropping in wireless ad hoc networks," in *ADHOCNOW*, vol. 2856, 2003.

[64] R. Roman, J. Zhou, and J. Lopez, "Applying intrusion detection systems to wireless sensor networks," in *IEEE CCNC*, 2006.

[65] J. M. Mccune, E. Shi, A. Perrig, and M. K. Reiter, "Detection of denial-of-message attacks on sensor network broadcasts," in *IEEE S&P*, 2005.

[66] B. Yu and B. Xiao, "Detecting selective forwarding attacks in wireless sensor networks," in *IPDPS*, 2006.

[67] S. Kaplantzis, A. Shilton, N. Mani, and Y. Sekercioglu, "Detecting selective forwarding attacks in wireless sensor networks using support vector machines," in *ISSNIP*, 2007.

[68] B. Barak, S. Goldberg, and D. Xiao, "Protocols and lower bounds for failure localization in the internet," in *Eurocrypt*, 2008.

[69] X. Zhang, A. Jain, and A. Perrig, "Packet-dropping adversary identification for data plane security," in *ACM CONEXT*, 2008.

[70] B. Xiao, B. Yu, and C. Gao, "CHEMAS: Identify suspect nodes in selective forwarding attacks," in *Journal of Parallel and Distributed Computing*, vol. 67, no. 11, 2007.

[71] J. R. Hughes, T. Aura, and M. Bishop, "Using conservation of flow as a security mechanism in network protocols," in *IEEE S&P*, 2000.

[72] K. Zhang, X. Zhao, and S. Wu, "An analysis on selective dropping attack in bgp," in *IEEE IPCCC*, 2004.

[73] A. T. Mizrak, Y. C. Cheng, K. Marzullo, and S. Savage, "Fatih: detecting and isolating malicious routers," in *DSN*, 2005.

[74] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford, "Path-quality monitoring in the presence of adversaries," in *ACM SIGMETRICS*, 2008.

[75] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," in *IEEE INFOCOM*, 2004.

[76] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks," in *IEEE S&P*, 2004.

[77] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh, "Toward resilient security in wireless sensor networks," in *ACM MobiHoc*, 2005.

[78] Z. Yu and Y. Guan, "A dynamic en-route scheme for filtering false data injection in wireless sensor networks," in *IEEE INFOCOM*, 2006.

[79] F. Ye, H. Yang, and Z. Liu, "Catching Moles in Sensor Networks," in *IEEE ICDCS*, 2007.

[80] N. Subramanian, C. Yang, and W. Zhang, "Securing distributed data storage and retrieval in sensor networks," in *IEEE PERCOM*, 2007.

[81] W. Zhang, M. Tran, S. Zhu, and G. Cao, "A random perturbation-based scheme for pairwise key establishment in sensor networks," in *ACM MobiHoc*, 2007.

[82] W. Zhang, N. Subramanian, and G. Wang, "Lightweight and compromise-resilient message authentication in sensor networks," in *IEEE INFOCOM*, 2008.

[83] C. Wang, T. Feng, J. Kim, G. Wang, and W. Zhang, "Catching packet droppers and modifiers in wireless sensor networks," in *IEEE SECON*, 2009.

[84] C. Wang, T. Feng, J. Kim, G. Wang, and W. Zhang, "Catching packet droppers and modifiers in wireless sensor networks," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 5, pp.835–843, 2012.

[85] H. Chan and A. Perrig, "Security and privacy in sensor networks," in *Computer*, vol. 36, no. 10, 2003.

[86] Q. Li and D. Rus, "Global clock synchronization in sensor networks," in *IEEE INFOCOM*, 2004.

[87] K. Liu, J. Deng, P. K. Varshney, and K. Balakrishnan, "An acknowledgment-based approach for the detection of routing misbehavior in manets," in *Mobile Computing, IEEE Transactions on*, vol. 6, no. 5, 2007.

[88] K. Sun, P. Ning, C. Wang, A. Liu, and Y. Zhou, "TinySeRSync: Secure and resilient time synchronization in wireless sensor networks," in *ACM CCS*, 2006.

[89] H. Song, S. Zhu, and G. Cao, "Attack-resilient time synchronization for wireless sensor networks," in *Ad Hoc Networks*, vol. 5, no. 1, 2007.

[90] E. Brickell and J. Li, "Enhanced privacy ID from bilinear pairing for hardware authentication and attestation, " in *SOCIALCOM* , 2010.

[91] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, "AnonySense: privacy-aware people-centric sensing, " in *ACM MobiSys* 2008.

[92] K. Ren and W. Lou, "A sophisticated privacy-enhanced yet accountable security framework for metropolitan wireless mesh networks," in *IEEE ICDCS*, 2008.

[93] J. Zhang, L. Ma, W. Su, and Y. Wang, "Privacy-preserving authentication based on short group signature in vehicular networks, " in *Proceedings of the The First International Symposium on Data, Privacy, and E-Commerce*, 2007.

[94] X. Lin, X. Sun, P.-H. Ho, and X. Shen, "GSIS: A secure and privacy-preserving protocol for vehicular communications," in *Vehicular Technology, IEEE Transactions on*, vol. 56, no. 6, pp. 3442–3456, 2007.

[95] R. Lu, X. Lin, H. Zhu, P.-H. Ho, and X. Shen, "ECPP: Efficient conditional privacy preservation protocol for secure vehicular communications," in *IEEE INFOCOM*, 2008.

[96] D. He, J. Bu, S. Zhu, M. Yin, Y. Gao, H. Wang, S. Chan and C. Chen, "Distributed privacy-preserving access control in a single-owner multi-user sensor network, " in *IEEE INFOCOM*, 2011.

[97] W.Zhang and C. Wang, "LA^3: a Lightweight Accountable and Anonymous Authentication Scheme, " submitted to *ASIACCS*.

[98] D. Chaum and E. Van Heyst, "Group signatures, " in *EUROCRYPT*, pp. 257–265, 1991.

[99] J. Camenisch and M. Michels, "A group signature scheme based on an rsa-variant, "in *Tech. rep*, 1998.

[100] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme, " in *CRYPTO*, pp. 255–270, 2000.

[101] G. Ateniese, D.X. Song, and G. Tsudik, "Quasi-efficient revocation in group signatures, " in *Financial Cryptography*, pp. 183–197, 2002.

[102] J. Camenisch and A. Lysyanskaya, “Dynamic accumulators and application to efficient revocation of anonymous credentials, ” in *CRYPTO*, pp. 61–76, 2002.

[103] D. Boneh and H. Shacham, “Group signatures with verifier-local revocation,” in *ACM CCS*, 2004.

[104] D. Boneh, X. Boyen, and H. Shacham, “Short group signatures, ” in *CRYPTO*, pp. 41–55, 2004.

[105] J. Furukawa and H. Imai, “An efficient group signature scheme from bilinear maps, ” in *Information Security and Privacy, Lecture Notes in Computer Science*, vol.3574, pp.92-128, 2005.

[106] J. Camenisch and A. Lysyanskaya, “Signature schemes and anonymous credentials from bilinear maps, ” in *CRYPTO*, pp. 56–72, 2004.

[107] J. Camenisch, and J. Groth, “Group signatures: Better efficiency and new theoretical aspects, ” in *SCN*, pp. 120–133, 2004.

[108] D.X. Song, “Practical forward secure group signature schemes, ” in *ACM CCS*, 2001.

[109] T. Nakanishi, and N. Funabiki, “Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps, ” in *ASIACRYPT*, pp. 533–548, 2005.

[110] T. Nakanishi, and N. Funabiki, “A short anonymously revocable group signature scheme from decision linear assumption, ” in *ASIACCS*, pp. 337–340, 2008.

[111] B. Libert, and M. Yung, “Dynamic fully forward-secure group signatures, ” in *ASIACCS*, pp. 70–81, 2010.

[112] S.D. Gordon, J. Katz, and V. Vaikuntanathan, “A group signature scheme from lattice assumptions, ” in *SIACRYPT*, pp. 395–412, 2010.

[113] T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki, “Revocable group signature schemes with constant costs for signing and verifying, ” in *Public Key Cryptography, Lecture Notes in Computer Science*, vol. 5443, pp. 463–480, 2009.

[114] C.I. Fan, R.H. Hsu, and M. Manulis, "Group signature with constant revocation costs for signers and verifiers, " in *Cryptology and Network Security, Lecture Notes in Computer Science*, vol. 7092, pp. 214–233, 2011.

[115] G. Ateniese, J. Camenisch, S. Hohenberger, and B. de Medeiros, "Practical group signatures without random oracles, " in *Cryptology ePrint Archive*, [Online]. Available:http://eprint.iacr.org/2005/385.pdf, 2005.

[116] X. Boyen, and B. Waters,"Compact group signatures without random oracles, " in *EUROCRYPT*, pp. 427–444, 2006.

[117] X. Boyen, and B. Waters, "Full-domain subgroup hiding and constant-size group signatures, " in *Public Key Cryptography*, pp. 1–15, 2007.

[118] J.Groth, "Fully anonymous group signatures without random oracles, " in *ASIACRYPT*, 2007.

[119] T. Nakanishi, and N. Funabiki, "A short verifier-local revocation group signature scheme with backward unlinkability, " in *Lecture Notes in Computer Science*, vol. 4266, pp. 17–32, 2006.

[120] W. Diffie, and M. Hellman,"New directions in cryptography, " in *IEEE Transactions on Information Theory*, 1976.

[121] D. Boneh, "The decision-diffie-hellman problem," in *Proceedings of the Third Algorithmic Number Theory Symposium*, vol. 1423, pp. 48–63, 1998.

[122] F. Vercautern, "Main computational assumptions in cryptography, "[Online]. Available: http://www.ecrypt.eu.org/documents/D.MAYA.3.pdf, 2010.

[123] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf, "Pseudonym systems, " in *SAC*, 1999.

[124] "Flexiprovider, " [Online]. Available: http://www.flexiprovider.de/.

[125] Certicom Research, "Sec 2: Recommended elliptic curve domain parameters, " in *Standards for Efficient Cryptography*,[Online]. Available: http://www.secg.org/collateral/sec2_final.pdf, 2000.

[126] A. De Caro, V. Iovino, "jPBC: Java pairing based cryptography, " in *IEEE Symposium on Computers and Communications*, 2011.

[127] "Java pairing-based cryptography library," [Online]. Available: http://gas.dia.unisa.it/projects/jpbc/.

[128] "Benchmark, " [Online]. Available: http://libeccio.dia.unisa.it/projects/jpbc/benchmark.html.

[129] A. Sahai and B.R. Waters, "Fuzzy identity-based encryption, " in *EUROCRYPT*, 2005

[130] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data", in *ACM CCS*, 2006

[131] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption, " in *IEEE S&P*, 2007.

[132] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: an online social network with user-defined privacy, " in *ACM SIGCOMM*, 2009.

[133] J. Camenisch, M. Dubovitskaya, and G. Neven, "Oblivious transfer with access control, " in *ACM CCS*, 2009.

[134] D. Khader, "Attribute based group signature with revocation, " in *Cryptology ePrint Archive*, [Online]. Available: http://eprint.iacr.org/2007/241, 2007.

[135] D. Khader, "Attribute based group signatures," in *Cryptology ePrint Archive*, [Online]. Available: http://eprint.iacr.org/2007/159.pdf, 2007.

[136] H.K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures, " in *RSA*, 2011.

[137] J. Li, M.H. Au, W. Susilo, D. Xie, and K. Ren, "Attribute-based signature and its applications, " in *ACM ASIACCS*, 2010.

[138] W. Zhang and C. Wang, "An Ad Hoc Group Signature Scheme for Accountable and Anonymous Accessed to Outsourced Data, " in *ACNS*, 2012.

[139] D. Boneh, X. Boyen, "Short signatures without random oracles, " in *Eurocrypt*, 2004.