Efficient authentication schemes for routing in mobile ad hoc networks

Shidi Xu
University of Wollongong

Efficient Authentication Schemes for Routing in Mobile Ad Hoc Networks

A thesis submitted in fulfillment of the
requirements for the award of the degree

Master of Computer Science by Research

from

UNIVERSITY OF WOLLONGONG

by

Shidi Xu

School of Information Technology and Computer Science
June 2006

Dedicated to

my parents

Declaration

I, Shidi Xu, declare that this thesis, submitted in partial fulfilment of the requirements for the award of Master of Computer Science by research, in the School of Information Technology and Computer Science, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualification at any other academic institution.

Shidi Xu
June 22, 2006

Publication

Journal Paper

Shidi, Yi Mu, Willy Susilo. "Authenticated AODV Routing Protocol Using One-Time Signature and Transitive Signature Schemes", *Journal of Networks*, pp. 47-53, Volume 1, Issue 1, 2006.

Conference Paper

Shidi Xu, Yi Mu, and Willy Susilo. "Secure AODV Routing Protocol Using One-Time Signature", *In Proceedings of the International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, pp. 288-297, Springer, 2005.

Shidi Xu, Yi Mu, and Willy Susilo. "An Efficient Authentication Scheme for MANET Routing", *In Proceedings of the 1st International Workshop on Security in Ubiquitous Computing Systems*, pp. 854-863, Springer, 2005.

Shidi Xu, Yi Mu, and Willy Susilo. "Online-Offline Signatures and Multisignatures for AODV and DSR Routing Security", *Accepted in the 11th Australasia Conference on Information Privacy and Security (ACISP'06)*, 2006.

Abstract

Mobile ad hoc network (MANET) has been generally regarded as an ideal network model for group communications. However, the security deployment for MANET routing operations is problematic. Firstly, existing secured routing protocols are deficient in achieving both authentication efficiency and full scale of security. In addition, the diversity of routing protocols presents difficulties in the generalisation of the security design. The most possible candidate solution,the digital signature, has far from been properly implemented from an ad hoc point of view.

In this thesis, we are motivated to provide necessary security features to MANET routing operations in an efficient manner. Considering the feasibility of utilising digital signatures in MANET, we incorporate the notion of the online/offline signature scheme in our design, where the computational overhead is shifted to the offline phase. We also make use of the one-time signature scheme, which is efficient in computation, and the multisignature scheme, which is especially suitable for group authentication. Then, we observe the specialities of different routing protocols (AODV-ad hoc on-demand distance vector routing and DSR-dynamic source routing), as well as the similarities between above signature schemes.

In our design, we exploit the efficiency and the adaptability of signature schemes. As our contributions, we propose two authentication schemes to secure AODV and DSR protocol respectively. For AODV protocol, our ID-based online/offline signature schemes enhance the authentication performance by properly balancing the computational overhead, whereas the one-time signature scheme achieves the same objective by making trade-offs between computation power and memory storage. For DSR protocol, we provide a generic construction from ID-based online/offline signature schemes to ID-based multisignature schemes, so that the installation over AODV can be transformed to offer the same level of security for DSR. Our scheme is *unique*, in the sense that a single ID-based online/offline signature scheme can be applied to both AODV and DSR routing protocols.

Acknowledgements

I would like to thank Dr. Yi Mu and Dr. Willy Susilo, my supervisors, for their patient guidance and constant support during my study. I admire their wealth of knowledge in security and cryptography, and appreciate them taking me into the area of cryptography.

I am also grateful to Dr. Joonsang Baek. I appreciate his instruction on security proof methods, which helped me a lot in my later study.

I greatly appreciate Professor John Fulcher for his help in the English expression of this thesis, and the support received from all the staff in the School of IT and CS, University of Wollongong.

Finally, I would like to thank my parents, who support me constantly with their love. Without them, I would never be able to have all my achievements.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The tremendous changes recently in the wireless industry largely accelerate the commercialisation of the mobile ad hoc networks. The idea of supporting mobile users on a peer-to-peer basis in the absence of a centralised controller was reviewed in the mid-nineties. This concept of ad-hoc networking has been successful in the invention of some of the latest technologies, such as Bluetooth and mobile ad-hoc sensors, which are in use on various platforms [24, 63]. People are now expecting efficient group communication in education, entertainment, and industries enabled by mobile ad hoc networks.

Communication in networks implies transmitting data packets along some certain paths, *routes*. How to find the path (*routing*) thus enable data transmission is the fundamental step of network communication. Routing [94] is conducted using *routing protocols* which use metrics to evaluate what path will be the best for a packet to travel. A *metric* is a standard of measurement, such as path bandwidth, which is used by routing algorithms to determine the optimal path to a destination. To enable the process of path determination, routing algorithms initialise and maintain routing tables, which contain route information.

On decision of the availability and optimality of a path, the information acquired during routing is justified using metrics. However, if the routing information is maliciously modified during routing , the routing protocol will then not be able to draw the correct conclusion. Attacks against routing protocols include denial-of-service (DoS), forming routing loops by modifying routing packets, IP spoofing, etc, have been extensively studied. However, prevention approaches offered by routing algorithms are far from sufficient.

For a networks without pre-defined infrastructure, routing is even more important

than in fixed networks, where pre-set routes are available. In a mobile ad hoc network (MANET), without the correct performance of routing protocols, the network is futile. Moreover, routing operations are even more likely to be disrupted in mobile ad hoc scenarios where the open and dynamic infrastructure offers advantages to unauthorise nodes to get access into the network and move around. In order to prevent mobile ad hoc routing operations from being sabotaged thereby protecting the availability of the network, security features are supposed to be deployed.

Cryptographic primitives which provide authentication, integrity and non-repudiation are especially suitable for the mobile ad hoc scenario. Digital signatures, which have been long used as an authentication method, offer the above three properties. However, the deployment of a digital signature enabled authentication scheme in mobile ad hoc networks is not straightforward. The way leading to secure mobile ad hoc environments is full of disturbances.

1.2 The Challenges

In this thesis, we are motivated to offer complete but efficient secure features for two significant mobile ad hoc routing protocols, AODV and DSR. We intend to provide authentication, integrity and non-repudiation for AODV and DSR routing operations.

It is generally recognised that the security deployment for MANET routing protocol is difficult because of the following reasons.

1. No central control exists in MANET. In a pure ad hoc environment, there is no trusted third party in the network. All the nodes are equally likely. The absence of trusted third party causes the major difficulty in our security deployment. The public key cryptography we use to provide authentication has to be constructed with the help of a certification centre or certification authority (CA), which is a trusted third party. Without a CA, there is no way to authenticate the linkage between the public key and the key holder. In this sense, authentication, integrity and non-repudiation are compromised.

2. Nodes are *resource constraint*. In MANET, nodes are mobile devices. Compared with devices such as laptops and desktops, mobile devices are limited by their computational power, memory size and battery life. The security deployment, such as signature generation and verification, will somehow consume the limited resources, which in turn affects the performance of the node.

3. Routing protocols are *distinct*. According to Misra's survey [65] and Royer *et al.*'s survey [86] in 1999, there exists more than a dozen different routing protocols in the literature. These protocols are designed based on different routing algorithms and almost share no common attributes. Accordingly, an authentication scheme designed for certain types of routing protocols will not be applicable to others. On the other hand, design a general authentication scheme without considering the nature of protocols will result in huge waste in routing operation overhead.

4. There exist various attacks against routing processing. Besides the general attacks which can be prevented using authentication and integrity protection, some other more complicated attacks such as worm hole [38] and sybil [23] are not detectable using a normal authentication scheme.

1.3 The Solutions

In contrast to the challenges we are facing in the secure deployment of MANET, we offer some solutions, which will be further discussed later in this thesis.

1. We argue the insignificance of the pure ad hoc scenario. To establish the security foundation for our scheme, in our construction, we assume the existence of an offline trusted third party to handle all the trust issues. Before entering the network, nodes must submit their identities to this trusted third party in order to obtain necessary information, such as keys and system parameters, for authentication. In addition, the behaviour of the trusted third party is in accordance with the secure deployment of the network, i.e. the trusted third party will not behave maliciously by disclosing subscribers' secret information or launching attacks.

2. The resource constraint forces us to make tradeoffs between the security and the efficiency, or possibly between the computational cost and bandwidth. For example, pairing based cryptography provides shorter signature size, whereas computing pairing is comparatively inefficient. One-time signature schemes have the benefit in efficient signature generation, however the resulting key size and signature size are tremendous. Hence, in design of our scheme, we need to take into consideration both the computational cost and bandwidth capacity. It is generally desirable if a scheme can provide a full scale of security features but have lower computational and bandwidth overhead.

3. To mitigate the difficulties caused by the diversity of routing protocols, we suggest the design of authentication schemes with adaptability. By doing this, one single scheme can be shared by several routing protocols, where the scheme works with each of the routing protocol in a harmonious sense. To achieve this, it is necessary to start with observing the similarities between different schemes and enable their transformations in-between.

4. The detection of sophisticated attacks requires the collaboration between normal authentication scheme and some other intrusion detection mechanisms such as neighbourhood detection.

In this thesis, we concentrate on providing efficient authentication schemes for mobile ad hoc network routing operations. We present signature constructions which are computationally efficient, small in size, adaptable, and offer authentication, integrity and non-repudiation. The installation is targeting two most popular routing protocols for MANET: AODV and DSR. We are trying to provide a concrete solution to the problems identified in the previous section.

1.4 Thesis Structure

The rest of the thesis is organised as followed:

- In Chapter 2, we introduce the basics of mobile ad hoc networks and their routing paradigms. We begin with analysing our application domain in order to extract the specialities of MANET. Then we examine the routing procedures conducted by two most famous routing protocols - AODV and DSR. Consequently, we are able to identify the attacks and review current countermeasures. Finally we can come up with the necessary security features which will provide the guidance in our design. In addition, we also review the secure routing protocols which exist in the literature and examine the security features they provided.

- We introduce all the cryptographic primitives to be used in our design in Chapter 3. We provide formal definitions for the cryptographic techniques covered in this thesis. Furthermore, we focus on the digital signature schemes which satisfy our security goals. In order to help bring up formal discussions in Chapter 4 and Chapter 5, we present the generic scheme and security arguments for each category of signature schemes covered. We also review the algorithms for some significant schemes to acquire a clearer understanding.

- We present our first cryptographic construction in Chapter 4. We extend the definition of the online/offline signature schemes introduced in Chapter 3 to an identity based scenario. Similar to Chapter 3, we give the generic scheme and analyse its security requirements. Then we provide two ID-based online/offline signature schemes (IOS) of our own construction and prove their security.

- In Chapter 5, we provide the construction of our identity based multisignature scheme (IBMS). We firstly formalise the generic scheme of ID-based multisignature and analyse its security requirements. Then we observe the similarity between ID-based online/offline signature schemes and ID-based multisignature scheme in their construction. Hence we present a generic construction from IOS to IBMS, which takes an IOS as input and transform it to an IBMS. To show the correctness of the generic scheme, we make use of the second IOS scheme introduced in Chapter 4 to show the actual transformation. The security of the generic scheme as well as the concrete scheme are both examined. Finally, we conduct a comparison over current IBMS schemes and their prototype schemes and are able to show that our scheme is efficient in signature generation.

- The implementation of our schemes over MANET routing protocols are described in Chapter 6. We focus on offering efficient authentication schemes for AODV and DSR protocol. We begin with identifying the security requirements for AODV and DSR respectively. We are motivated to provide sender authentication and hop-by-hop authentication in routing operations. We also observe the specialities of AODV and DSR protocol in our design. Finally, we are able to present two constructions for AODV by using IOS and HORS one-time signature, and one construction for DSR by using IBMS. We give the detailed construction and argue the tradeoffs.

- Chapter 7 is the conclusion, where we summarise the contribution of this thesis, and propose future research directions.

Chapter 2

MANET Preliminaries

In this chapter, we introduce the application domain used in our construction: Mobile Ad Hoc Networks (MANET). We start by identifying the main properties of mobile ad hoc networks which affect our design. The routing procedure of the mobile ad hoc network, which is the target of this thesis, is discussed. We further describe two famous mobile ad hoc network routing protocols: *ad hoc on-demand distance vector routing protocol* (AODV) and *dynamic source routing protocol* (DSR) in detail. Based on the detailed routing operations, we are able to recognise the major threats presented to mobile ad hoc routing security, as well as the performance of current countermeasures. We come up with a list of security requirements to be followed in designing secure routing protocols for mobile ad hoc networks. Finally, we briefly review previous work on secure routing protocols and justify their performance.

2.1 Mobile Ad Hoc Networks

The Mobile Ad hoc Network (MANET) is a collection of mobile nodes that are dynamically and arbitrarily located in such a manner that the interconnection between nodes are capable of changing on a continual basis [86]. We extend this definition as follows so as to give a clearer glance of MANET.

- It is an *infrastructureless* network. There is no pre-image that can be made on how the network will be formed. Even after the formation of the network, the topology is still unpredictable. Thus, nodes need to be constantly informed about the current status of the network. Otherwise, the network will fall int pieces.
- Nodes in the network communicate with each other through radio signals, which are broadcasted to the whole network and can be received by everyone. Since

the radio signals have a certain transmission range, the activity of nodes is consequently limited within some areas. Namely, mobile nodes can only communicate directly with others within their transmission ranges, which are referred as "neighbours". To reach out of their neighbourhoods, nodes must count on their neighbours to forward the data. Thus, each node must be able to act as a router to discovery paths and forward data packets.

- All the nodes in MANET are equally likely. No node is superior or inferior to others. There is no central control over MANET. Every node takes exactly the same responsibility in the network. Besides, no node is more credible than other nodes in nature. Online trusted third parties such as public key infrastructure (PKI) or time stamping services [44] are not available.

- Network topology continues changing. Nodes are free to join and leave the network whenever they want. They are also able to move around while still maintaining their connections. The move or leave of one node is simply regarded as a broken of linkage to that node and will be broadcasted to other nodes using this link. In a word, the network is highly dynamic.

- The mobile nodes in MANET are usually resource constrained. The joint nodes are usually laptops, PDAs and even network-enabled mobile phones. These mobile devices usually have low computational power and limited battery life. Although some of them have become more and more powerful recently, they are generally designed to compromise their performance (e.g. constantly go into power-saving mode) in order to live longer.

The dynamic nature makes MANET become one of the most powerful communication network infrastructure so far. On the other hand, the popularisation of MANET has been stumbled because of some major designing difficulties such as computational and bandwidth efficiency and security.

2.2 MANET Routing Protocols

Routing is the foundation of facilitating communication within networks. The distinct nature of MANET results in the existence of a group of specific routing protocols [31, 43, 65, 72, 73, 86]. Generally, MANET routing protocols are categorised into two groups: *table-driven* routing protocols and *on-demand* routing protocols.

In table-driven routing protocols, each node maintains one or more tables which contains routing information to every other node in the network. All nodes update their tables in order to maintain a consistent and up-to-date view of the network. When a topology change occurs, nodes will sense this change and will propagate update messages throughout the network. Then other nodes will be able to update their tables according to the message. Besides, nodes also inform other nodes about their status information by periodically propagating status messages. Through active information exchanging, all the nodes will be able to finally obtain the up-to-date topology information. When there is data to be sent, nodes can simply search their tables and extract the route. It is an active approach to conduct routing.

On the other hand, on-demand routing protocols take a lazy approach. Nodes do not propagate the topology status to each other. Instead of maintaining the topology information for the whole network, nodes maintain the information for active routes only. Besides, nodes do not actively look for available routes. Routes are created when they are required. When a node wants to send data to a destination, it invokes a route discovery mechanism to find the suitable path. This route will remain valid until a failure on this route is detected.

In general, on-demand approach is more preferable than table driven approach because of the nature of MANET. Nodes in MANET are mostly mobile devices. Like PDAs, they are usually constrained by their memory size and battery life. Besides, the wireless connection bandwidth is not as much as in fixed networks. Unlike table-driven approach, on-demand routing protocols do not require large memory spaces for routing tables. Since there is no periodical propagated messages, the bandwidth usage is reduced. Consequently, since no network-wide propagations is needed, the battery life is saved.

2.2.1 AODV Routing Protocol

Ad hoc On-demand Distance Vector routing Protocol (AODV) [73] is one of the most popular MANET on-demand routing protocol. It emerged as an on-demand version of distance vector routing protocol [55], which is based on the classical Distributed Bellman-Ford (DBF) algorithm [9]. It enables mobile nodes to obtain routes quickly for new destinations, and does not require nodes to maintain inactive routes to destinations. The routing messages do not contain information about the whole route path, but only about the source and destination. Therefore, routing messages are not increasing in size. In addition, AODV uses the destination sequence number to ensure

loop freedom, which is easy to implement. All these features enable AODV to be a suitable routing protocol for MANET.

The routing operations of AODV generally consist of two phases: route discovery and route maintenance, shown in 2.1 Route discovery is performed through broadcasting *RREQ* messages. Whenever a node needs to send data packets to a destination, it first checks if it has an existing route in the routing table. If not, the source node will initiate a *RREQ* and broadcast this request to all the neighbours. Then neighbouring nodes will update their routing table according to the received message.

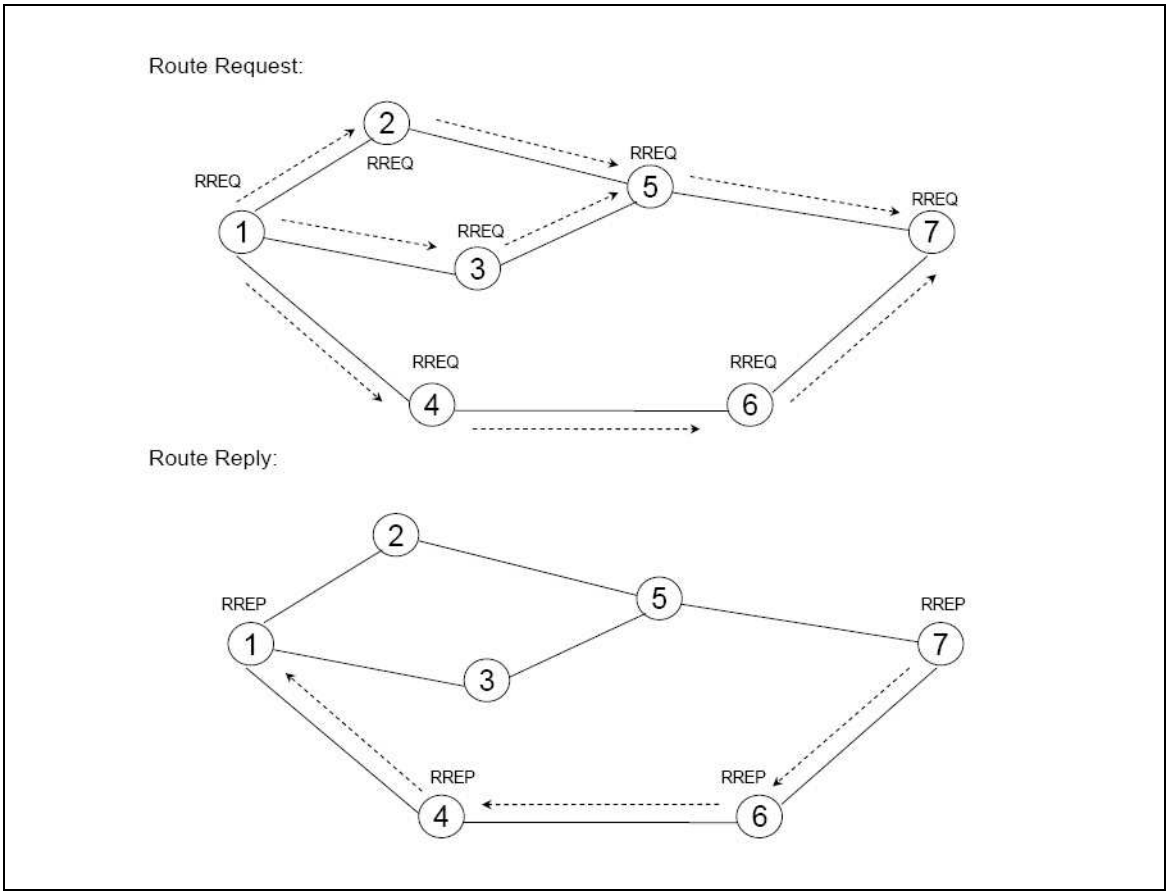

Figure 2.1: The Route Discovery Process for AODV Protocol

When the *RREQ* reaches the destination, a *RREP* will be generated by the destination node as a response to the *RREQ*. The *RREP* will be transmitted back to the originator of the *RREQ* in order to inform the route. If an intermediate node has an active route towards the destination, it can reply the *RREQ* with a *RREP*, which is called Gratuitous Route Reply. The intermediate node will also send an *RREP* to the destination node. The *RREP* will be sent in reverse route of *RREQ* if a bidirectional link exists.

Route maintenance is performed with two additional messages: *Hello* and *RRER* messages. Each node broadcast *Hello* messages periodically to inform neighbours about its connectivity. The receiving of *Hello* message proves that there is an active route towards the originator. Each forwarding node should keep track of its continued connectivity to its active next hops. If a link to the next hop cannot be detected during a period of time-out, a *RRER* message will be broadcast to inform the loss of connectivity. On receiving this *RRER*, usually a local repair will be performed just for maintenance. The expired route will be deleted after the confirmation of its unavailability.

2.2.2 DSR Protocol

DSR stands for dynamic source routing protocol, presented by Johnson and Maltz [43] in 1996. It is an on-demand routing protocol based on the concept of source routing, which means the initiator knows the complete hop-by-hop route to the destination. This specific feature brings efficiency, but also results in the scaling of routing message overhead. To perform DSR, each node is required to maintain a route cache which contains the topology information of the network. The route cache is consistently updated to reflect the current status of the network.

Similar to AODV, this protocol consists of two major phases: route discovery and route maintenance, as shown in 2.2 When a source node originates a packet addressed to a certain destination, the initiator first searches its route cache for a route. If there exists an active route towards the destination, this route will be used. Otherwise, the node generates a route request packet (*RREQ*) which consists of a data structure called *route record* listing the IP addresses of all the intermediate nodes. This *RREQ* will be broadcast to neighbours. The receiving node will have two choices.

1. If it is not the target node of this route discovery, it appends its own address to the route record in the Route Request and propagates it by transmitting it as a local broadcast packet (to its neighbours).

2. If it is the target node, it returns a Route Reply to the initiator, giving a copy of accumulated route record from the Route Request.

This process will be continued until the *RREQ* packet reaches the destination. The original message is not changed during the transmission (except the *RREQ* data length field which is a number). The resulting route will be found in the *route record*.

The data structure of *RREQ* consists of two fields: IP fields and route request fields. IP fields contains *source address*, *destination address* and *hop limit*. Route

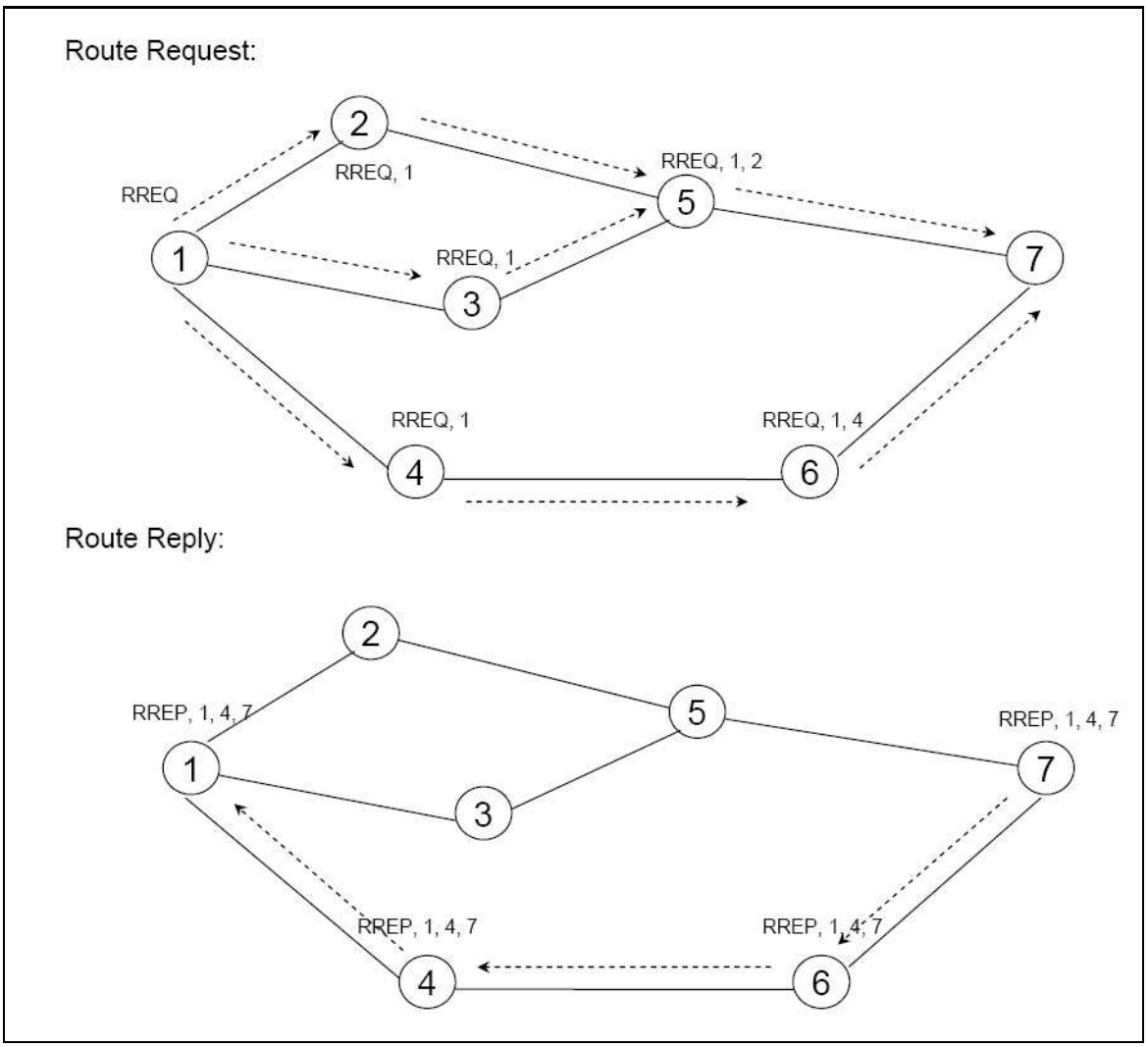

Figure 2.2: The Route Discovery Process for DSR Protocol

request fields contains *option type*, *option data length*, *identification*, *target address*, and *route record*. When a *RREQ* is received, the option data length fields will be increased by 4 and the node's IP address will be appended to the end of the route record. Other fields will remain unchanged during the whole route discovery process.

In replying the *RREQ*, the target node generates a route reply packet (*RREP*) and sends it back to the initiator by two ways. It can simply reverse the sequence of hops in the route record and use it as the source route on the Route Reply. Otherwise, it searches its own route cache for a route back to the initiator. If such route does not exist, the target should initiate a Route Request back to the initiator.

During transmission, each node on the route is responsible for confirming that data

can flow over the link from that node to the next hop. Since periodic routing advertisement is not available, nodes use the acknowledgement (ACK) to provide confirmation that a link is capable of carrying data. The acknowledgement can be required by a node. If the acknowledgement request has been retransmitted for the maximum number of times without being replied, the sender should treat this link as currently "broken". It should remove this link from its route cache and should return a Route Error ($RERR$) to each node that has sent a packet routed over that link since an acknowledgement was last received.

2.2.3 Other Routing Protocols

The increased interests in MANET results in a rapid growth in the number of MANET routing protocols. Besides AODV and DSR, There are some other on-demand routing protocols in the literature. Light-weight mobile routing (LMR) [1] is an on-demand routing protocol which utilises flooding technique to determine the routes. Nodes maintain multiple routes to each required destination, which increases the reliability of the protocol by allowing nodes to select the next available route to a particular destination without initiating a route discovery procedure. In addition, nodes only maintain information for neighbours, which reduces the storage overheads.Temporally ordered routing algorithm (TORA) [69] is based on LMR protocol. It has all the benefit of LMR and has the extra benefit that TORA has reduced the far-reaching control messages to a set of neighbouring nodes, where the topology change has occurred. It also supports multicasting.

Besides the on-demand routing strategy, the table-driven routing protocols are also preferred. The destination-sequenced distance vector (DSDV) [72] makes use of distributed Bellman-Ford algorithm to guarantee the loop free routes. To reduce the overhead caused by flooding the update packets over the network, it introduced the conception of "full dump" packets and "incremental" packets. The full dump packets carry all the available routing information, whereas the incremental packets carry only the information changed since the last full dump.

The global state routing (GSR) [18] is based on the traditional link state algorithm. It improved the way information is disseminated in link state algorithm by restricting the update message between intermediate nodes only. In GSR, nodes maintain link state tables based on the information received from neighbours. The link state information is exchanged periodically among neighbours only, which significantly reduced the number of control messages transmitted through the network.

Distance routing effect algorithm for mobility protocol (DREAM) [7] conducts routing operation by using geographic coordinates through a GPS device. These coordinates are periodically exchanged between each node and stored in routing table called location table. This approach can significantly reduce the bandwidth consumption since the overhead is much smaller than exchanging the complete link state information.

Hierarchical state routing (HSR) [71] is an adaption of traditional link state routing protocols. It maintains a hierarchical addressing and topology map, and makes use of clustering algorithms such as CGSR to organize the nodes with close proximity. There are three types of nodes involved:

- a cluster-head node which acts as a local coordinator for each node;
- gateway nodes which are nodes that lie in two different clusters;
- internal nodes that are all the other nodes in each cluster.

Nodes within each cluster broadcast their link state information to each other. Each node has a unique ID and a hierarchical ID (HID). Logical clustering provides a logical relationship between the cluster-head at a higher level. The logical nodes are connected via logical links, which form a "tunnel" between lower level cluster. The logical link information along with the summary information of the lower level clusters are exchanged through logical link, then flooded down the the lower levels. The physical nodes at the lowest level will then have a "hierarchical" topology of the network. The advantage of HSR is the separation of mobility management from the physical hierarchy.

Optimised link state routing (OLSR) [41] is a point-to-point routing protocol based on the traditional link state algorithm. In OLSR, each node maintains topology information about the network by periodically exchange link state messages. However, the size of each control message and the number of rebroadcasting nodes during each route update are minimized by employing multipoint replaying (MPR) strategy.

2.3 MANET Routing Security Requirements

Mobile ad hoc network suffers from many threats which can be categorised as follows [14].

External Attacks: which are committed by parties that are not legally parts of the network.

Internal Attacks: which are originated from inside a particular network.

Passive Attacks: which do not involve of any disruption of the services, they are merely intended to steal information and eavesdrop on communication within the network.

Active Attacks: which actively alter the data, with the intension of overloading the network, obstructing the operation or cut off certain nodes from their neighbours so that they cannot use the network serviced effectively anymore.

External attacks can be prevented through implementing a firewall or deploying proxy. Internal attacks performed by network peers are much more difficult to be detected and prevented. These attacks are usually originated from compromised nodes' malicious behaviours. Passive attacks do not disturb routing operations, but they are usually the first step of launching other active attacks. By eavesdropping communication, attackers may be able to learn the topology information, such as which node is the bottleneck of the network, and then launch attacks against that node. There are also some sophisticated attacks, exploiting design flaws of basic routing protocols, including black hole [3,80] and rushing attacks [36]. The rest of this chapter will introduce some common attacks against MANET.

2.3.1 Threats and Countermeasures

- **Threats Using Modification**

  This kind of attack [31,52] is usually performed by modifying the routing information, aiming at compromising the integrity of routing computations. In MANET routing, network topology is maintained by flooding routing information through out the network. Any alteration of those messages will cause dramatic topology change, and the result could be devastating. Current ad hoc routing protocols generally assume that nodes will not alter the routing message fields, which makes this kind of attack extremely easy to be launched.

  The only way to protect the routing information is to introduce message integrity into routing, which is usually achieved through using a digital signature with a public key certificate. However, in MANET, there is no PKI or central control existing, plus computing digital signature is not efficient enough, some other techniques are introduced to achieve message integrity. One commonly used technique is keyed message authentication code (MAC). MAC is calculated over

the routing packet to produce a fingerprint of the routing information. A secret key is used to guarantee that no one can forge the MAC.

To protect values such as hop count, an easier method is to use hash chain [49]. A secret seed is hashed repeatedly to create a chain of hash values. This chain is used in the reverse direction, in which the last element in the chain is regarded as an anchor. Authentication is performed through hashing some element repeatedly again to check it with the anchor. This method is not attack free. It is only able to prevent attackers from decreasing the value, whereas they are still capable of increasing the values unexpectedly or leaving it unchanged. However, these are not regarded as serious attacks.

- **Threats Using Impersonation (Spoofing)**

  Spoofing [14, 20, 31, 52] means an attacker assumes the identity of another node in the network, thus receiving messages that are directed to the node that it fakes. This kind of attack is commonly known in wired network, but becomes more serious in wireless networks. Because current ad hoc routing protocols do not authenticate the source IP address, attackers can easily masquerade other nodes. It is usually the first step to intrude a network so as to carry out further attacks to disrupt operations.

  To defend against spoofing, nodes need to be authenticated. The authentication can be performed using the public key certificate or the shared secret between nodes. Even so, it still does not prevent an attacker from impersonating other nodes' IP addresses or MAC addresses. Only the public key certificate issued by a trusted third part can perfectly solve this problem, but it is generally not possible to be implemented in MANET.

- **Threats Using Fabrication**

  Fabrication attacks are usually conducted by generating false routing messages, trying to disturb network topology [31, 62]. It is regarded as route misbehaviour, which is very difficult to be detected.

  AODV and DSR are especially vulnerable to this kind of attack. In AODV, a malicious node can prevent communication between any two nodes by flooding spoofed *RRER* messages along the path. *RRER* messages claim that the next hop of the originator is currently unavailable. Any nodes receiving this message will mark this link as "broken". Further, a malicious node can continue sending

spoofed *RRER* if the link is re-established, resulting in complete isolation of a targeting node.

Currently fabrication is defended by making use of the non-repudiation property of digital signature, because usually nodes are required to sign on the false message it created. Once the false message is detected, the digital signature provides an evidence to exclude this misbehaving node. Nevertheless, it has no effort on detecting such malicious behaviours.

- **Threats Using Tunneling**

  A tunneling attack [39] is launched where two or more nodes collaborate to encapsulate and exchange messages between them among existing routes. It causes false routes being generated by replacing the true path potions with the tunnel created between two or more malicious nodes. For example, assume a *RREQ* is supposed to be transmitted along the authentic route $O \rightarrow A \rightarrow B \rightarrow C \rightarrow T$. Two malicious nodes M_1 and M_1 establish a tunnel between each other to launch the following attack. When M_1 receives the *RREQ*, it sends this *RREQ* by tunneling to M_2. Then M_2 forwards the *RREQ* to the target node T and tunnels back a *RREP*. By this means, the resulting route becomes $O \rightarrow M_1 \rightarrow M_2 \rightarrow T$ of hop count 3. The tunnel could be created using the existing path $A \rightarrow B \rightarrow C$, or some other channel of the outside current network. Whatever methods used, the resulting route is a fake but will be accepted because it is the shortest route.

2.3.2 Security Requirements

We identify the following properties required by MANET to secure its routing communication.

- **Availability**

  Availability for MANET is primarily concerned with routing, since the availability of network connection is maintained by routing. It is the basis of networking.

- **Authenticity**

  Authenticity consists of user authenticity and data authenticity. In MANET, it is essential for a node to identify itself to other nodes. Since the network is dynamic and open, to defend against malicious attacks, when communicating with a certain node, it is important to ensure that the node is the one we expect to

communicate with. This can be achieved through user authentication. Besides, the data transmitted through the network should be authenticated too, to ensure that the content of the data is valid. For routing, data authenticity guarantees that the received routing information is up-to-date, namely the network topology regarding that information remains unchanged.

- **Integrity**

  Integrity of data ensures that the data remains unchanged during transmission. Obviously the information can be altered, but the alteration can be detected.

- **Non-repudiation**

  Through non-repudiation, the sender cannot deny that it has sent the message because the message is authorised by the sender when sending out. It is used for intrusion detection through which attackers can be identified.

In order to ensure the basic feature-availability of the network, we must guarantee the faultless performance of routing operations. Because routing contacts with the network structure, the routing algorithms developed for wired networks cannot be simply applied to MANET without adaptation. Therefore, designing a routing protocol that is able to take on all the required routing operations, at the same time fulfilling relatively high security demand, is of major concern for researchers.

2.4 Secure Routing protocols

The security problems in MANET routing operations have been observed for a long time. Lots of works have been done in order to provide a secure routing protocol for MANET [34,37,88,103–106,109]. However, current efforts towards the design of secure routing protocols are mainly oriented to on-demand routing protocols such as DSR and AODV. In the literature, a number of secure routing protocols have been proposed to address different security problems.

2.4.1 ARAN Authentication Scheme

ARAN [88] stands for Authenticated Routing for Ad hoc Networks. It is motivated to detect and protect against malicious actions by third parties and peers in an ad hoc

environment. ARAN is a security scheme, which can be applied to any on-demand routing protocol. It takes the advantages of PKI based digital signature schemes to provide security features including authentication, message integrity and non-repudiation.

Routing operations of ARAN are performed using three data structures: the route discovery packet (*RDP*), the reply packet (*REP*), and the error packet (*ERR*). Each of them contains necessary routing information as well as the public key certificate. When a node wants to initiate a route discovery, it creates a signed *RDP* and broadcasts it to the next hop. The next hop node verifies the originator's signature. If it is authentic, it adds its own certificate and signs the whole packet again. The following hop node performs the same operation, however, after the verification of all the signatures of the received *RDP* it replaces previous hop node's signature with its own. Operations repeated until the packet reaches the target, as shown in Fig. 2.3.

Route Request along path: $(O \rightarrow A \rightarrow B \rightarrow C \rightarrow T)$

$O \rightarrow$ **broadcast:** $Sign_{SK_O}(RDP, IP_T, Cert_O, N_O, t)$

RDP - packet type identifier
IP_T - IP address of the target node T
$Cert_O$ - public key certificate of the originator O
N_O - monotonically increasing nonce of O
t - current time

$A \rightarrow$ **broadcast:** $Sign_{SK_A}(Sign_{SK_O}(RDP, IP_T, Cert_O, N_O, t)),\ Cert_A$

$B \rightarrow$ **broadcast:** $Sign_{SK_B}(Sign_{SK_O}(RDP, IP_T, Cert_O, N_O, t)),\ Cert_B$

$C \rightarrow$ **broadcast:** $Sign_{SK_C}(Sign_{SK_O}(RDP, IP_T, Cert_O, N_O, t)),\ Cert_C$

Figure 2.3: ARAN *RDP* Packet Propagation

When the target node receives this *RDP*, it replies with a route reply packet (REP). This packet is in the same format of the *RDP*, containing destination's signature and certificate. Each forwarding node verifies the signature, removes previous hop node's signature, and then adds its own outside the packet. If this route reply reaches the originator, it is guaranteed that the route found is authentic. The REP propagation is shown as in Fig. 2.4

The authentication scheme provided by ARAN defends against exploits using modification, fabrication and impersonation. However, the use of public key cryptography

Route Reply along path: $(T \rightarrow C \rightarrow B \rightarrow A \rightarrow O)$
$T \rightarrow C$: $Sign_{SK_T}(RDP, IP_O, Cert_T, N_T, t)$
$C \rightarrow B$: $Sign_{SK_C}(Sign_{SK_T}(RDP, IP_O, Cert_T, N_T, t)),\ Cert_C$
$B \rightarrow A$: $Sign_{SK_B}(Sign_{SK_T}(RDP, IP_O, Cert_T, N_T, t)),\ Cert_B$
$A \rightarrow O$: $Sign_{SK_A}(Sign_{SK_T}(RDP, IP_O, Cert_T, N_T, t)),\ Cert_A$

Figure 2.4: ARAN REP Packet Propagation

is very costly. The computational overhead caused by signature generation and verification brings tremendous burden for mobile nodes. A group of malicious nodes may exploit this vulnerability to launch a deny of service attack by simply broadcasting large number of *RDP* packets. The receiving nodes have to exhaust their computational resources to verify the signature and then generate new ones. In addition, the extra bandwidth used to transmitting certificate is also another burden.

2.4.2 Other Secure Protocols

Secure Routing Protocol (SRP) proposed by Papadimitratos and Haas [68] is a DSR based protocol. SRP makes use of the security association (SA) between every two nodes to enable the authentication and message integrity. The trust relationship is instantiated by the knowledge of public keys of each communication part. With the help of SA, a pair of nodes can share a secret key and use this shared secret key to construct the MAC. By doing this, SRP is robust against IP spoofing and unauthorised modification of routing packets. Nevertheless, the establishment of SA is not practical in reality because it requires pre-deployment before the network formed.

Ariadne, proposed by Hu, Perrig and Johnson [37], is also a DSR based secure routing protocol. Similar to SRP, it requires pre-deployment of authentication keys between the source and the destination. Somehow, Ariadne provides three key sharing approaches corresponding to three authentication methods: pairwise shared secret keys, TESLA [77] keys and digital signatures. With the help of the security deployment, Ariadne guarantees:

- the target node of a route discovery process can authenticate the initiator;
- the initiator can authenticate each intermediate node on the path to the target,

which presents in the *RREP*;

- no intermediate node can remove a previous node from the node list of *RREQ* or *RREP*.

However, the pairwise key sharing approach requires huge key setup overhead where $n(n+1)/2$ keys are needed for a network with n nodes. The TESLA protocol assumes clock synchronisation, which has been generally regarded as impractical. The last approach, digital signature, is not computationally efficient, and requires the existence of a trust third party such as the public key infrastructure (PKI).

Another famous secure routing protocol in this context is called secure AODV routing protocol (SAODV) [105]. It is an extension of AODV routing protocol, aimed to provide security adds-on to AODV. SAODV uses conventional digital signatures to provide authentication and message integrity. The signature is constructed over the immutable fields of routing packets. Mutable fields, such as hop count, are secured by hash chains [49]. However, to make the scheme more efficient, SAODV does not require the existence of any trusted third party, which highly reduce the security level of the scheme.

Besides three routing protocols introduced above, there are around ten secure routing protocol proposals existed in the literature. However, none of them is able to give a perfect answer to the open problem on how to provide full scale security to MANET routing operations with the least cost.

2.5 Summary

Mobile ad hoc networks have been extensively studied in the literature since its first appearance. Used to be exclusive to military applications, the concept of *commercial ad hoc networks* arrived with notebook computers and other viable communications equipment in the 1990s. It has been found desirable in the commercial sector to provide highly dynamic and scalable wireless communications. On the other hand, it presents some embarrassment for the research sector to find out efficient network construction approaches.

As a highly dynamic, infrastructureless network, how to find peer nodes and establish links, namely routing, become the major issue to be solved. Since the debut of MANET in 1970s, a large number of routing protocols have been proposed. Among these proposals, AODV and DSR stand out above the rest, becoming the two most

popular targets of the research community, as well as any adversaries. Attacks disrupt the normal routing process by taking advantage of the insecured communication channel, which presents great threats in the popularisation of MANET.

However, various attacks are of help in clarifying the weak points of the network. Learning from trial-and-error, we are now able to identify the security requirements of MANET, including: availability, authentication, message integrity and non-repudiation.

In addition, the existence of distinct routing protocols presents difficulties in standardisation of routing models. Accordingly, it affects the design of authentication schemes because we always take the specific routing procedures into consideration. Therefore, in this thesis, we bring forward the problem of designing an adaptable authentication scheme for both AODV and DSR protocols and try to introduce the first scheme in order for the other to come up with more valuable opinions.

Chapter 3

Cryptographic Preliminaries

Digital signatures have long been used to provide authentication, integrity and non-repudiation. The variations of normal digital signature schemes offer additional properties suitable for different applications. In this thesis, we focus on signature schemes that are efficient in signature generation in individual and group situations. Online/offline signature schemes, offering quick signing, and multisignature schemes, adept in group signature construction, are of the best candidates for our application. Besides, the identity based cryptosystem, having good properties in solving public key distribution problems, is also a desirable scheme in our application. Thus, in this chapter, we aim to study the above three digital signature schemes and their security requirements. We start by introducing related cryptographic primitives including one-way hash functions, the random oracle model, elliptic curve cryptography, and bilinear pairing.

3.1 Cryptography Basics

3.1.1 One-Way Functions

A one-way function [98] is a mathematical function that is significantly easier to compute in one direction (the forward direction) than in the opposite direction (the inverse direction). Informally, a function f is a one-way function if:

1. The description of f is publicly known and does not require any secret information for its operation.

2. Given x, it is easy to compute $f(x)$.

3. Given y, in the range of f, it is hard to find an x such that $f(x) = y$. More precisely, any efficient algorithm solving a P-problem succeeds in inverting F with negligible probability.

One-way functions are usually constructed using some secret information called *trapdoor*. A trapdoor one-way function is a one-way function which is easy to be inverted with the knowledge of the trapdoor, but difficult otherwise.

Definition 1 *[57] A* trapdoor one-way function *is a one-way function $f : X \to Y$ with the additional property that given some extra information (called the* trapdoor information*) it becomes feasible to find for any given $y \in Im(f)$[1], an $x \in X$ such that $f(x) = y$.*

It remains to be rigorously established whether there actually are any (true) one-way functions. That is to say, no one has yet definitively proved the existence of such functions under reasonable definitions of "easy" and "computationally infeasible". Since the existence of one-way functions is still unknown, the existence of trapdoor one-way functions is also unknown. However, there are a number of good candidates for one-way and trapdoor one-way functions, e.g. the factorisation of a product of two large primes. While selecting and verifying two large primes and multiplying them together is easy, factoring the resulting product is known as difficult. This is the basis for RSA encryption [44], which is conjectured to be trapdoor one-way.

The trapdoor one-way functions are the basis of public key cryptography. The public key gives information about the particular instance of the function, whereas the private key gives information about the trapdoor. Whoever knows the trapdoor can compute the function easily in both directions, but anyone lacking the trapdoor can only perform the function easily in the forward direction. The forward direction is used for encryption and signature verification; the inverse direction is used for decryption and signature generation.

In almost all public key cryptosystems, the size of the key corresponds to the size of the inputs to the one-way function; the larger the key, the greater the difference between the efforts necessary to compute the function in the forward and the inverse directions (for someone lacking the trapdoor). For a digital signature to be secure for years, for example, it is necessary to use a trapdoor one-way function with inputs large enough that someone without the trapdoor would need many years to compute the inverse function in order to generate a legitimate signature.

[1] *$Im(f)$ denotes the the set of all elements in $f(x)$*

3.1.2 Cryptographic Hash Functions

Cryptographic hash functions play a fundamental role in modern cryptography. Hash functions take a message as input and produce an output referred to as a hash value. The basic idea of cryptographic hash functions is that a hash value serves as a compact representative image (digital fingerprint) of an input string, and can be used as if it is uniquely identifiable with that string. Formally, we define hash functions as follows.

Definition 2 *A hash function is a function h which has, as a minimum, the following two properties:*

1. compression - *h maps an input x of arbitrary finite bit length, to an output $h(x)$ of fixed bit length n.*

2. ease of computation - *given h and an input x, $h(x)$ is easy to compute.*

At the highest level, hash functions may be split into two classes: unkeyed hash functions, whose specification dictates a single message as its input parameter; and keyed hash functions, whose specification dictates two distinct inputs, a message and a secret key.

The unkeyed hash functions are also regarded as *modification detection codes* (MDCs). MDCs can be constructed using block cipher such as MDC (using DES) and modular arithmetic algorithms such as MASH-1 (Modular Arithmetic Secure Hash algorithm 1). However, what we used most is the one called customized hash functions, which are specifically designed "from scratch" for the explicit purpose of hashing, with optimized performance in mind, and without being constrained to reuse existing system components such as block ciphers or modular arithmetic. Those having received the greatest attention in practice are based on the MD4 hash function, including MD5 [83] and SHA-1 [6], etc.

The unkeyed hash functions can be further classified as one-way hash functions (OWHFs), which is difficult in finding an input which hashes to a pre-specified hash-value; and collision resistant hash functions (CRHFs) which is difficult in finding any two inputs having the same hash value. To facilitate further definitions, the following properties are identified for unkeyed hash functions [57]. For an unkeyed hash function h with inputs x, x' and outputs y, y':

1. *pre-image resistance* - for essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output, i.e., to find any

pre-image x' such that $h(x') = y$ when given any y for which a corresponding input is not known.

2. *2nd-pre-image resistance*it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given x, to find a 2nd-pre-image $x' \neq x$ such that $h(x) = h(x')$

3. *collision resistance* - it is computationally infeasible to find any two distinct inputs x, x' which hash to the same output, i.e., such that $h(x) = h(x')$. (Note here free choice of both inputs.)

Definition 3 *A one-way hash function (OWHF) is a hash function h as in Definition 2 (i.e., offering ease of computation and compression) with the following additional properties, as defined above: pre-image resistance, 2nd-pre-image resistance.*

Definition 4 *A collision resistant hash function (CRHF) is a hash function h as in Definition 2 (i.e., offering ease of computation and compression) with the following additional properties, as defined above: 2nd-pre-image resistance, collision resistance.*

On the other hand, the keyed hash functions are regarded as message authentication code (MACs). The purpose of a MAC is (informally) to facilitate, without the use of any additional mechanisms, assurances regarding both the source of a message and its integrity. MACs have two functionally distinct parameters, a message input and a secret key. They are a subclass of keyed hash functions.

Definition 5 *A message authentication code (MAC) algorithm is a family of functions h_k parameterised by a secret key k, with the following properties:*

1. ease of computation - *for a known function h_k, given a value k and an input x, $h_k(x)$ is easy to compute. This result is called the MAC-value or MAC.*

2. compression - *h_k maps an input x of arbitrary finite bit length to an output $h_k(x)$ of fixed bit length n.*

3. computation-resistance - *given a description of the function family h, for every fixed allowable value of k (unknown to an adversary), given zero or more text-MAC pairs $(x_i, h_k(x_i))$, it is computationally infeasible to compute any text-MAC pair $(x, h_k(x))$ for any new input $x \neq x_i$ (including possibly for $h_k(x) = h_k(x_i)$ for some i).*

The applications of cryptographic hash functions are very broad. In this thesis, we mainly focus on *data integrity* and *message authentication*. According to [57], we have the following definitions.

Definition 6 *Data integrity is the property whereby data has not been altered in an unauthorised manner since the time it was created, transmitted, or stored by an authorised source.*

Definition 7 *Data origin authentication is a type of authentication whereby a party is corroborated as the original source of specified data created at some typically unspecified time in the past.*

Definition 8 *Message authentication is a term used analogously with data origin authentication. It provides data origin authentication with respect to the original message source and data integrity, but no uniqueness and timeliness guarantees.*

To provide message authentication, we can make use of the message authentication code (MAC) or digital signatures. However, the authentication schemes based on key-sharing do not provide non-repudiation of the data origin.

3.1.3 Hash Chain

One application of cryptographic hash function in authentication is the hash chain. A hash chain, introduced by Lamport [49], is a sequence of hash values generated with a certain seed. It is an efficient authentication scheme which has enormous applications, including password based authentication [32, 49], certificate revocation [2, 56], secure address resolution [30], micropayment [33, 70, 84], online auctions [93], digital cash [67], efficient multicasting [75–77], server-supported signatures [5, 22], spam fighting protocols [25], one-time signature schemes [74], sensor network security protocols [53, 78], and securing routing information [34, 35, 108].

Fig. 3.1 shows the construction of a hash chain. To construct a chain of length ℓ, we randomly pick up a value as the *seed* s_ℓ, which will be the last value in the chain. We repeatedly apply a hash function f to this seed ℓ times, in order to obtain $\ell + 1$ hash values $s_0, ..., s_\ell$. These values is supposed to be revealed in the reverse order of its generation, where s_0 is the first one and s_ℓ is the last one. We call the first value of a hash chain as an *anchor*.

The verification of any element in the hash chain is through the seed s_0. For example, to verify an element s_i is of the index i, we check that whether the the

following equation holds.

$$f^i(s_i) = s_0$$

More generally, if we know s_i is the ith element of the hash chain, we can use it to verify s_j if $i < j$ by compute

$$f^{j-i}(s_j) = s_i$$

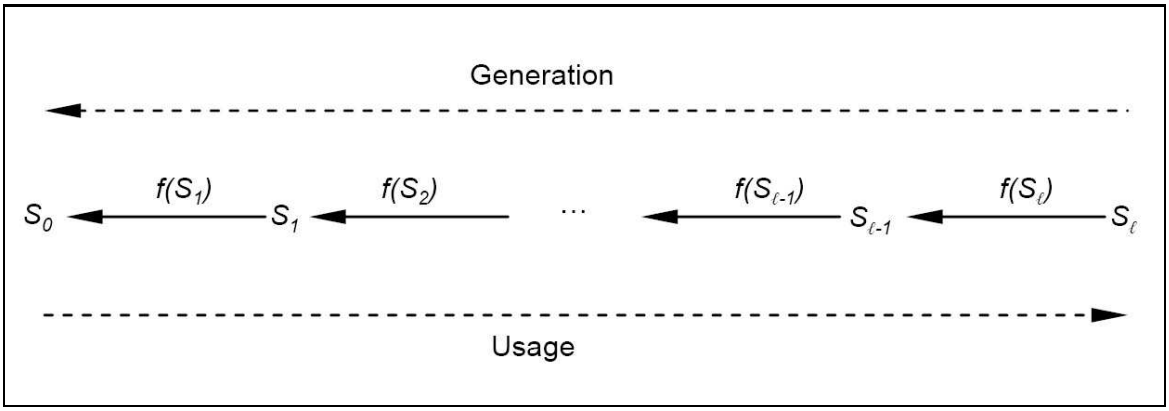

Figure 3.1: Hash Chain Construction

However, hash chain suffers from the limitation that the length of the hash chain is finite, which means it needs to be re-initialised after being used for a certain number of times. It is possible to create the hash chain as long as possible but extra memory spaces are needed to store the whole hash chain. On the other hand, frequently re-initialising hash chains consumes the computational power of a device. It is necessary to make a trade-off between memory usage and computational power consumption.

The problem of re-initialisation of hash chains has been widely discussed in the literature. Bicaki and Baykal [10] proposed the construction of infinite length hash chains using public key cryptography with the cost of compromising on efficiency. Goyal [29] presented a new idea of Re-initialisable Hash Chain (RHC) which can be securely re-initialised in an non-repudiable manner. Their method makes use of the one-time signature scheme to establish the link between the old chain and the newly generated on. The re-initialisation process can be repeated indefinitely to create an infinite number of finite length hash chains which are all tied together.

Another problematic issue of a hash chain is its traversing cost. Two extreme approaches for this problem is (for a hash chain of size n):

1. Store only the seed of the hash chain, which is of computational complexity $\mathcal{O}(n)$;

2. Store the entire hash chain, which is of storage complexity $\mathcal{O}(n)$.

Above complexities have been reduced firstly by Jakobosson [42] to $\lceil \log(n) \rceil$ for computation and $\lceil \log(n) \rceil + 1$ for memory, by using the technique called "fractal hash chain traversal". Copersmith and Jakobsson [21] later further reduced the computational complexity by approximately a factor of two, i.e., approximately $\frac{1}{2}\log(n)$ for computation and a little more than $\log(n)$ for storage. Sella [90] then reduced the storage complexity to $k\sqrt[k]{n}$ where $k = m+1$ and m is a constant bound on the number of hash function evaluations allowed per each exposed link in a chain.

3.1.4 Random Oracle model

A *random oracle* is a mathematical abstraction used in cryptographic proofs. It is typically used in proofs to provide sufficient mathematical properties, which cannot be implemented in reality, in order to satisfy the proof of security. Proofs which make use of random oracles are referred to as secure in the "*random oracle model*" which was formalised by Bellare and Rogaway [8], as opposed to the "standard model".

In practice, random oracles are typically used to model cryptographic hash functions in schemes where strong random assumptions are needed of the hash function's output. Such proofs indicate that systems or protocols are secure by showing that an attacker must require impossible behaviour from the oracle, or solve some other mathematical problems believed hard, in order to break the protocol. Not all usage of cryptographic hash functions require random oracles: schemes which require only the property of collision resistance can be proven secure in the standard model.

According to Canetti, Goldreich and Haleviz [15], in a scheme working in the random oracle model, all parties including the adversary are modeled by probabilistic polynomial time interactive machines with oracle access. It is assumed that all oracle queries, regardless of the identity of the party making them, are answered by a single function denoted $\mathcal{O}$ which is uniformly selected among all possible functions [15]. The set of possible functions is determined by a function $\ell_{out}(\cdot)$, which has a constant length output, and by the security parameter of the system.

Formally, a random oracle $\mathcal{O}$ is a mapping $G : \{0,1\}^* \rightarrow \{0,1\}^\infty$, chosen by selecting each bit of $\mathcal{O}(x)$ uniformly and independently for every x. In a system with security parameter k, the random oracle function is a mapping $G : \{0,1\}^{poly(k)} \rightarrow \{0,1\}^{\ell_{out}(k)}$. In addition, when a random oracle is given a query x it does the following:

- If the oracle has been previously given the query x, it responds with the same

value it gave the last time.

- If the oracle has not been previously given the query x, it generates a random response which has uniform probability of being chosen from anywhere in the oracle's output domain.

In the random oracle model, the definition of Goldwasser, Micali and Rivest's signature scheme [28] $(Gen, Sign, Verify)$ is randomized as $(Gen^R, Sign^R, Verify^R)$, where:

Gen^R On input 1^k, the generator produces a pair of matching public and secret keys (SK, PK).

$Sign^R$ To sign message m, compute $\sigma \leftarrow Sign^R(SK, m)$.

$Verify^R$ to verify (m, σ), compute $Verify^R(PK, m, \sigma) \in \{0, 1\}$. It must be the case that $Verify^R(PK, m, \sigma) = 1$ for all $\sigma \in Sign^R(SK, m)$.

In this sense, for a polynomial time adversary $\mathcal{F}$ with the access to the random oracle $\mathcal{O}$ and the signing oracle, we define its advantage as:

$$\mathsf{Adv} = \mathsf{Pr}[\mathcal{O} \leftarrow \{0,1\}^{\ell_{\mathsf{out}}(\mathsf{k})}; (\mathsf{m}, \sigma) \leftarrow \mathcal{F}^{\mathcal{O}, \mathsf{Sign}^{\mathsf{R}}(\mathsf{SK}, \mathsf{m})}(\mathsf{PK}) : \mathsf{Verify}^{\mathsf{R}}(\mathsf{PK}, \mathsf{m}, \sigma) = 1]$$

The signature scheme is secure if for every adversary $\mathcal{F}$, the Adv is negligible.

The problem with the random oracle model is that it does not really model real life. In reality we use a single fixed hash function, such as MD5 or SHA-1, to replace the random oracle. Assume the number of functions $\{0,1\}^k \leftarrow \{0,1\}$ is 2^{2^k}, a random function takes $\log 2^{2^k} = 2^k$ bits to represent. However, since MD5 and SHA-1 are polynomial-time computable, it is theoretically possible that the adversary would fail for a random function, but not for a polynomial-time one [97]. Canneti, Goldreich and Halevi [15] constructed an artificial counterexample: one that is provably secure in the random oracle model but insecure when the random oracle is instantiated in real-life with any polynomial-time computable function.

The random oracle model makes it possible to prove the security of a lot of practical signature schemes and encryption schemes [27, 89], but the meaning of these proofs is uncertain in reality [81]. The security based random oracle model resides on the intuitive level, which cannot be proven secure in the standard model. Nevertheless, the random oracle model is still a powerful tool in security proofs.

3.1.5 Elliptic Curve Cryptology Basics

Elliptic curve cryptography (ECC) was proposed by Miller [64] and Koblitz [46] in the mid 1980s. An *elliptic curve* [54] is a set of solution (x, y) to an equation of the form $y^2 = x^3 + Ax + B$, denoted by $E(\mathbb{F}_q)$, together with an extra point O, which is called *the point at infinity*. For applications to cryptography, we consider finite fields of q elements. The set of points on an elliptic curve forms a group under a certain addition rule, where the point O is the identity element of the group.

Formally, we have Weierstrass equation to define an elliptic curve as followed:

Definition 9 *The abelian group $E(\mathbb{F}_q)$ is defined by*

$$E(\mathbb{F}_q) = (x, y) \in \overline{\mathbb{F}_q} | y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \cup \infty$$

where $\overline{\mathbb{F}_q}$ denotes the algebraic closure[2] *of $\mathbb{F}_q$, and ∞, referred to as "the point at infinity", is the identity. The group operation, $P+Q$ for $P = (x_1, y_1)$, $Q = (x_2, y_2) \in \mathbb{F}_q$ behaves as followed*

$$\begin{aligned} P + Q &= \infty, \text{ if } x_1 = x_2 \text{ and } y_1 + y_2 + a_1x_2 + a_3 = 0 \\ P + Q &= (x_3, y_3), \text{ otherwise where} \\ x_3 &= \lambda^2 + a_1\lambda - a_2 - x_1 - x_2 \\ y_3 &= -(\lambda + a_1)x_3 - \nu - a_3 \end{aligned}$$

where λ and ν are rational functions of the curve coefficients and x_1, x_2, y_1, y_2.

One hard problem over elliptic curve is Elliptic Curve Discrete Logarithm Problem (ECDLP), which is generally regarded as the foundation of elliptic curve cryptography. To utilise ECDLP in construction of ECC, the cyclic subgroup of $E(\mathbb{F}_q)$ is generated by firstly choosing a point $P \in E(\mathbb{F}_q)$ and an integer n where

$$nP = \underbrace{P + P + ... + P}_{n}$$

We say n is the order of P if $nP = \infty$, denoted by $\#E(\mathbb{F}_q)$, and use $\langle P \rangle$ to denote the cyclic subgroup of $E(\mathbb{F}_q)$. Then the ECDLP is defined as followed:

Definition 10 *Given $\langle P \rangle$ of order n where $P \in E(\mathbb{F}_q)$ and a point $Q \in \langle P \rangle$, to find the smallest integer ℓ, $0 \leq \ell \leq n - 1$ such that $Q = \ell P$.*

[2] *The field $\overline{\mathbb{F}}$ is called an algebraic closure of $\mathbb{F}$ if $\overline{\mathbb{F}}$ is algebraic over and if every polynomial $f(x) \in \mathbb{F}[x]$ splits completely over $\overline{\mathbb{F}}$, so that $\overline{\mathbb{F}}$ can be said to contain all the elements that are algebraic over $\mathbb{F}$.*

The application of ECDLP in ECC is the elliptic curve scalar multiplication, which is a fundamental operation in elliptic curve cryptosystems. The result of scalar multiplication is another point Q on the curve. It is normally expressed as $Q = kP$. For cryptographic applications $\#E(GF(q)) = rh$ where r is prime, and h is a small integer and P and Q have order r. Scalars such as k are random integers where $1 < k < r$. Since $r \approx q$, the binary representation of $k = \Sigma_{i=1}^{n-1} k_i 2^i$ has n bits where $n \approx m = \lceil \log q \rceil$, which is frequently used in ECC efficiency descriptions.

Scalar multiplication is the most dominant computation part of elliptic curve cryptography. It is normally performed by repeating point addition (ECADD) and doubling (ECDBL) operations over the curve in some special way. ECADD and ECDBL operations in turn rely on finite field operations such as addition/subtraction, multiplication and inversion. Ansari and Hasan [4] proposed a high performance architecture of elliptic curve scalar multiplication over the finite field $GF(2^m)$. Implemented in hardware, this system performs a scalar multiplication in approximately $6\lceil m/w \rceil (m-1)$ clock cycles and the gate delay in the critical path is equal to $T_{AND} + (\log 2w) T_{XOR}$, where T_{AND} and T_{XOR} are delays due to two-input AND and XOR gates respectively.

3.1.6 Bilinear Pairing

Bilinear pairing maps a point on an elliptic curve to a finite field. According to Boneh, Mironov and Shoup [13], we have the following definitions. Let $\mathbb{G}_0, \mathbb{G}_1$ be a cyclic additive group generated by P, with a prime order q, and $\mathbb{G}_2$ be a cyclic multiplicative group with the same prime order p.

Definition 11 *(**Bilinear map**). A function $e : \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is bilinear if for any four elements $g_1, g_2 \in \mathbb{G}_0,\ H_1, H_2 \in \mathbb{G}_1$, the following holds:*

1. $e(g_1 \circ g_2, H_1) = e(g_1, H_1) \circ e(g_2, H_1)$
2. $e(g_1, H_1 \circ H_2) = e(g_1, H_1) \circ e(g_1, H_2)$

Definition 12 *(**Secure bilinear map**). A bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is (t, ε)-secure if for all t-time adversaries $\mathcal{A}$*

$$\mathsf{AdvBLM}_{\mathcal{A}} = \mathsf{Pr}[\mathsf{e}(\mathcal{A}(\mathsf{g}, \mathsf{G}, \mathsf{H}), \mathsf{H}) = \mathsf{e}(\mathsf{g}, \mathsf{G}) | \mathsf{g} \leftarrow^{\mathsf{R}} \mathbb{G}_0; \mathsf{G}, \mathsf{H} \leftarrow^{\mathsf{R}} \mathbb{G}_1] < \varepsilon$$

The probability is taken over the coin tosses of the algorithm $\mathcal{A}$ and random choice of $g,\ G,\ H$.

Secure bilinear maps include:

- $e(\cdot,\cdot) : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{F}^*_{p^r}$ is the Weil or Tate pairings on an elliptic curve $E/\mathbb{F}_p$ and $\mathbb{G}_0, \mathbb{G}_1$ are distinct subgroups of $E[q]$ for some prime q. For certain curve, the Weil pairing can be slightly modified so that we may take $\mathbb{G}_0 = \mathbb{G}_1$.
- $r(\cdot,\cdot) : \mathbb{Z}^*_N \times \mathbb{Z}^+_{\varphi(N)} \to \mathbb{Z}^*_N$. defined as $r(g, H) = g^H$, where N is a product of two primes.

In generally, a bilinear is $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a map with with the following properties:

1. Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1, a, b \in \mathbb{Z}^*_q$;
2. Non-degeneracy: There exists $P, Q \in \mathbb{G}_1$ such that $e(P, Q) \neq 1$;
3. Computability: There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in \mathbb{G}_1$;

The Non-degeneracy implies that when P is the generator of $\mathbb{G}_1$, $e(P, P)$ is the generator of $\mathbb{G}_2$. We call such bilinear map as an admissible bilinear pairing. Problems considered in the additive group $\mathbb{G}_1$ are:

- Decisional Diffie-Hellman Problem (DDHP): For $a, b, c \in \mathbb{Z}^*_q$, given P, aP, cP decide whether $c \equiv ab \mod q$.

- Computational Diffie-Hellman Problem (CDHP): For $a, b \in \mathbb{Z}^*_q$, given P, aP compute abP.

Definition 13 *(**Computational Diffie-Hellman Problem**). The Computational Diffie-Hellman problem is (t, ε)-hard if for all t-time adversaries $\mathcal{A}$ we have*

$$\mathsf{AdvCDH}_{\mathcal{A}} = \mathsf{Pr}[\mathsf{e}(\mathcal{A}(\mathsf{g}, \mathsf{H}, \mathsf{H}^{\mathsf{a}}) = \mathsf{g}^{\mathsf{a}} | \mathsf{g} \leftarrow^{\mathsf{R}} \mathbb{G}_0; \mathsf{H} \leftarrow^{\mathsf{R}} \mathbb{G}_1; \mathsf{a} \leftarrow^{\mathsf{R}} \mathbb{Z}_{|\mathbb{G}_1|}] < \varepsilon$$

In bilinear pairings, Decision Diffie-Hellman problem (DDHP) is easy and Computational Diffie-Hellman problem (CDHP) is still hard. That is, for $a, b \in \mathbb{Z}^*_q$, given P, aP, bP, computing abP is infeasible.

Definition 14 *A group $\mathbb{G}$ is a gap Diffie-Hellman(GDH) if there exists a polynomial time probabilistic algorithm to compute the decisional Diffie-Hellman problem but exists no such algorithm to solve the computational Diffie-Hellman problem in $\mathbb{G}$.*

Above system parameters can be obtain through running the **GDH Parameter Generator** [12] $\mathcal{IG}$ which takes a security parameter $k \in \mathbb{Z}^+$ as input, runs in polynomial time in k, and outputs a prime number q, the description of two groups $\mathbb{G}_1, \mathbb{G}_2$ of order q, and the description of an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.

Definition 15 *The advantage of an algorithm $\mathcal{A}$ in solving CDHP in group $\mathbb{G}$ is*

$$\mathsf{Adv}_{\mathcal{A}}^{CDH} = \Pr[\mathcal{A}(P, aP, bP) = abP : a, b \overset{R}{\leftarrow} \mathbb{Z}_q^*]$$

where the probability is over the choice of a and b, and the coin tosses of $\mathcal{A}$. We say that an algorithm $\mathcal{A}(t, \epsilon)$-breaks CDHP in $\mathbb{G}$ if $\mathcal{A}$ runs in time at most t, and $\mathsf{Adv}_{\mathcal{A}}^{CDH} > \epsilon$.

3.2 Digital Signature Schemes

A handwritten signature has long been used as a proof of the authorship to the content of a document. A digital signature of a message is a number dependent on some secrets only known by the signer, and additionally, on the content of the document being signed. As an asymmetric cryptographic authentication scheme, it is different from MAC in that the digital signature scheme is able to provide non-repudiation of the signer, where MAC can only protect message integrity.

3.2.1 Generic Scheme

The signing paradigm is performed as follows. Assume there are two users: Alice (A) and Bob (B). Each of them holds a public-private key pair ((PK_A, SK_A) for Alice and (PK_B, SK_B) for Bob). To sign a message m, Alice launches the signing algorithm $Sign$ along with her secret key SK_A to generate a signature S over the message. Alice then publishes the signature as well as her public key PK_A. When Bob receives the signature and Alice's public key, he will be able to verify if the signature is generated by Alice using the verification algorithm $Verify$. If the signature is authentic, Alice's public key will make the verification equation hold.

Formally, the digital signature scheme is defined by Goldwasser *et al.* [28] as follows:

Definition 16 *A* digital signature scheme *contains the following components:*

- *A* security parameter *k, which is chosen by the user when he creates his public and secret keys. This parameter determines a number of quantities, such as the length of signatures, length of signable messages, running time of the signing algorithm, overall security, etc.*

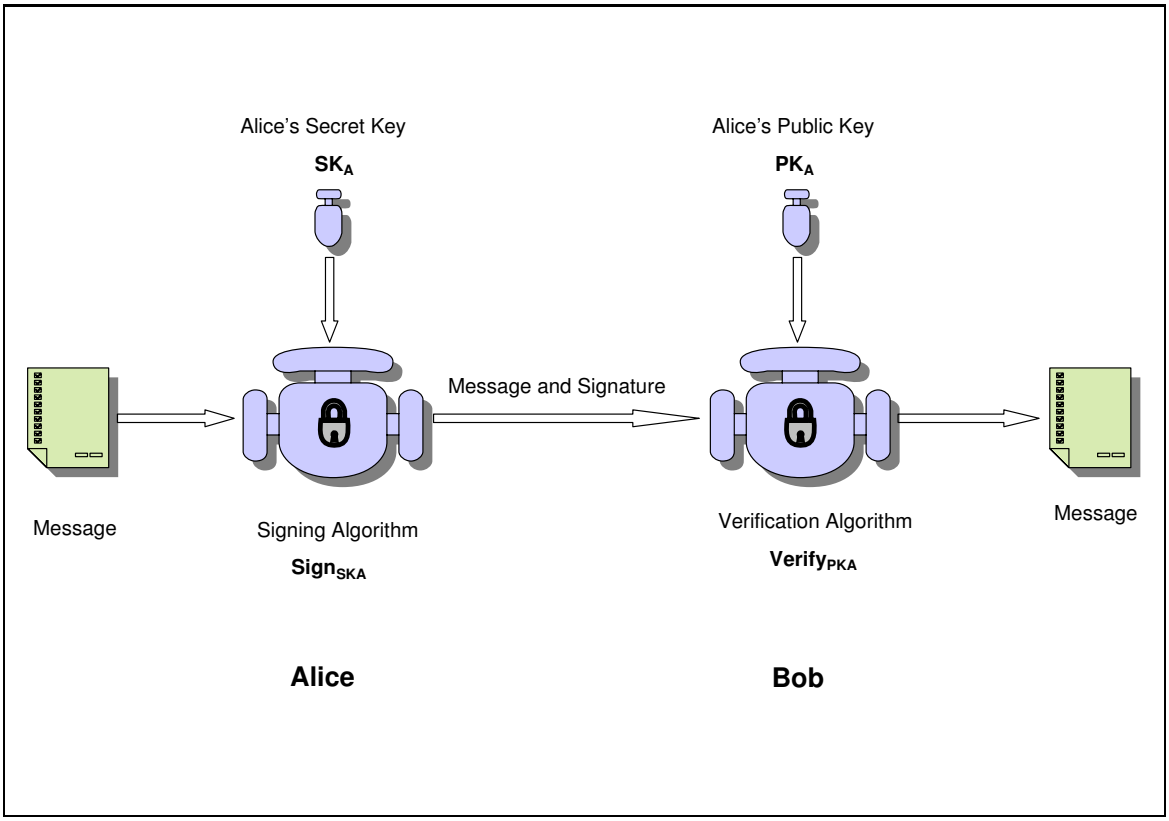

Figure 3.2: Digital Signature Scheme

- *A* message space $\mathcal{M}$ *which is the set of messages to which the signature algorithm may be applied. The messages can be regarded as binary strings, i.e.* $\mathcal{M} \subseteq 0, 1^+$. *The length of messages to be signed is bounded by* k^c *for some constant* $c > 0$.
- *A* signature bound B *which is an integer bounding the total number of signatures that can be produced with an instance of the signature scheme. This value is typically bounded above by a low-degree polynomial in* k, *but may be infinite.*
- *A* key generation algorithm G *which on input* 1^k *by any user* A, *generate a pair* (P_A^k, S_A^k) *of matching* public *and* secret *keys in polynomial time.*
- *A* signature algorithm σ *which produces a signature* $\sigma(M, S_A)$ *for a message* M *using the secret key* S_A.
- *A* verification algorithm V *which tests whether* S *is a valid signature for message* M *using the public key* P_A.

In general, the key generation algorithm and the signing algorithm are probabilistic.

Whereas the verification algorithm is deterministic.

3.2.2 Security Requirements for Digital Signature Schemes

The standard notion of security for a signature scheme is called *existential unforgeability under a chosen message attack*(EF-CMA) [28]. Assume the existence of a polynomial time adversary $\mathcal{A}$ and a challenger, who cooperate to perform the following game:

1. The challenger runs the key generation algorithm to generate the public-private key pair (PK, SK). It sends PK to the adversary and keeps SK as secret.

2. The adversary $\mathcal{A}$ produces a message m under PK and submits it to the challenger. The challenger responds the query with a signature $\sigma = Sign(m, SK)$. $\mathcal{A}$ can request at most q_S messages of his choice under PK, where $m_1, ..., m_{q_S} \in 0, 1^*$.

3. Eventually, $\mathcal{A}$ produces a pair (m^*, σ^*). The adversary wins if σ^* is a valid signature of m^* according to the verification algorithm, and m^* is not queried during the signature query phase.

Formally, we have

Definition 17 *An adversary $\mathcal{A}$ (t, q_S, ϵ) breaks a signature scheme, if $\mathcal{A}$ runs in time at most t, makes at most q_S signature queries, and the advantage that $\mathcal{A}$ wins the game is at least ϵ. A digital signature scheme is (t, q_S, ϵ)-existentially unforgeable under a chosen message attack if no adversary (t, q_S, ϵ) breaks it.*

$$\mathsf{AdvSig}_{\mathcal{A}}^{\mathsf{EF-CMA}} = \Pr\left[\mathsf{Verify}(\mathsf{pk}, \mathsf{m}, \sigma) = \text{accept} \;\middle|\; \begin{array}{l} (PK, SK) \leftarrow \mathsf{G}(1^{\mathsf{k}}); \\ for\ i = 1, 2, ..., q_S; \\ m_i \leftarrow \mathcal{A}(pk, m_i); \\ \sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{m}); \\ (m, \sigma) \leftarrow \mathcal{A}(pk, m_i, \sigma_i); \\ m \neq m_1, ..., m_{q_S}; \end{array}\right] \leq \epsilon$$

3.3 One-time Signature Schemes

The one-time signature scheme was first proposed by Lamport [48] in 1979. It is a digital signature mechanism which can be used to sign, at most, one message. Otherwise, signatures can be forged. A new public key is required for each message being signed. The public information necessary to verify one-time signatures is often referred to as validation parameters. When one-time signatures are combined with techniques for authenticating validation parameters, multiple signatures are possible.

3.3.1 Generic Scheme

One-time signature schemes generally consist of three algorithms: *key generation*, *signing* and *verification*.

KeyGen. On input security parameters 1^k, key generation algorithm generates a one-time public and secret key pair.

Sign. On input a message M, signing algorithm produces a one-time signature S.

Verify. On input a message M and a signature S, verification algorithm checks whether S is a valid signature for message M.

The above scheme does not show the speciality of one-time signatures. The major difference between one-time signatures and other signatures is the algorithm used to in signature generation. Normal signatures often use a trapdoor based algorithm to generate signature. However, one-time signatures use one-way function such as hash function to conduct signature generation. In addition, the public-secret key pair is also generated using one-way function. The public key is often the hash value of the secret key.

The formal definition of one-time signature is provided by Bleichenbacher and Maurer [11] using directed acyclic graphs.

Definition 18 *[11] Minimal verifiable sets will in the following be called* signing patterns*. The associated poset of DAG $\mathcal{G}$, denoted $\mathcal{G}^*$, is the poset $(\mathcal{G}^*, \leq)$ of signature patterns of $\mathcal{G}$. A* one-time signature scheme Γ *for $\mathcal{G}$ is an anti-chain of the associated poset $\mathcal{G}^*$, and the maximal size of an anti-chain in $\mathcal{G}$ is denoted by $w(\mathcal{G})$.*

The advantage of one-time digital signature schemes is that signature generation and verification are very efficient. Signing and verification are often one or more calls of a hash function. It is useful in applications such as single chip cards, where a low computational complexity is required.

The security of one-time signature schemes is often based on the random oracle model since a one-way function is the major component of the signing algorithm. However, there are also some other schemes, such as HORS [82], where security is based on complexity-theoretic assumptions.

3.3.2 Detailed Schemes

The very first one-time signature scheme proposed by Lamport [48] is only possible to sign one bit information. Based on this trivial, many researchers have subsequently

proposed more efficient schemes. Merkle [59] proposed an improvement of Lamport's scheme which reduce the number public key components by almost two-fold. Winternitz [60]'s improvement focused on the signature size. Using his scheme, the signature size can be reduced by several folds but at the expense of increased computational overhead. Later, Merkle proposed a tree-authentication scheme which is possible to sign multiple message using the same public key. More recently, Perrig [74] proposed a BiBa (Bins and Balls) one-time signature scheme for broadcast authentication scenario. His approach reduced the signature size and verification overhead with the cost of larger public key size and higher signature generation overhead.

In this section, we introduce two schemes: HORS and IOSP, in more detail, because these two schemes are related to our construction.

HORS Short One-time Signature Scheme

HORS stands for Hash to Obtain Random Subset. It was proposed by Reyzin *et al.* [82] in 2002, motivated to provide a more efficient signing algorithm better than BiBa One-time signature [74]. HORS consists of three algorithms: key generation, signing and verification.

Key Generation. On constructing this scheme, several security parameters are predefined. To sign b-bit messages, we firstly pick t and k such that

$$\left(\begin{array}{c} t \\ n \end{array} \right) \geq 2^b$$

Then choose a security parameter ℓ, and a one-way hash function f that operates on ℓ-bit strings. To generate public key, randomly generate ℓ-bit string $(s_1, s_2, , s_t)$. Let $v_i = f(s_i)$ for $1 \leq i \leq t$. The resulting public key is $PK = (n, v_1, v_2, ..., v_t)$, private key is $SK = (n, s_1, s_2, ..., s_t)$.

Signing. To sign a message m, with secret key $SK = (n, s_1, s_2, ..., s_t)$, firstly let $h = hash(m)$; then split h into k substrings $h_1, h_2, ..., h_n$, of length $\log t$ bits each; finally, interpret each h_j as an integer i_j for $1 \leq j \leq n$. The resulting signature is $\sigma = (s_{i_1}, s_{i_2}, ..., s_{i_n})$.

Verification. he verification is the same as the signature generation. Suppose the verifier has the message m, signature $\sigma = (s'_{i_1}, s'_{i_2}, ..., s'_{i_n})$,and public key $PK = (t, v_1, v_2, ..., v_t)$. Firstly, let $h = hash(m)$; then split h into k substrings $h_1, h_2, ..., h_n$, of length $\log t$ bits each and interpret each h_j as an integer i_j. If for each $1 \leq j \leq n$, $f(s'_j) == v_{i_j}$, accept the signature; otherwise, reject the signature.

In HORS, the public key component can be used multiple times. Signature generation requires only one call to hash function. Verification requires k calls to hash function. One impressive advantage of HORS is the shorter signature size. For their most efficient construction, the signature size can be reduced to 20480 bits.

Chained One-time Signature Protocol

Chained One-time Signature Protocol (IOSP) was proposed by Zhang [108] in 1998. His motivation was to produce an efficient signature scheme for routing protocols. This scheme makes use of one-way hash chain as public keys to construct the signature. It is similar to HORS in the sense that the resulting signature is a selection of secret key components in terms of the message.

IOSP includes three algorithms as followed.

Setup. 1. Each router chooses n secret key components x_j, $(j = 1, , t)$ at random.

2. Each router creates a n hash chain of length k.

$$\begin{pmatrix} 0 & h^0(x_1) & ... & h^0(x_n) \\ 1 & h^1(x_1) & ... & h^1(x_n) \\ ... & ... & ... & ... \\ k & h^k(x_1) & ... & h^k(x_n) \end{pmatrix}$$

3. Each router broadcasts the kth row of his table signed using public key cryptography.

4. Each router verifies the received row using public key cryptography and stores them as $v_j, j = 1, ..., n$.

Signing. 1. Obtain a n-bit binary string g by concatenating $f(M_i)$ with a count field using Merkle tree [59].

2. Form the signature by concatenating the hash value $h^{k-i}(x_j)$ in the $(k-i)$th row of the table for all j such that $g_j = 1$, where g_j is the jth bit of string g.

Verification. 1. Obtain a n-bit binary string g by concatenating $f(M_i)$ with a count field using Merkle tree.

2. For all j such that $g_j = 1$, check if $h^{k-i}(r_j) = v_j$, where r_j and v_j are the received and stored value for the jth bit respectively, and v_j is last updated for message i'.

3. If true, accept the message and update v_j with value r_j.

3.4 Identity based Signature Schemes

The concept of identity based cryptosystem was firstly introduced by Shamir [91] in 1984. His motivation was to simplify the public key certificate management process. In such cryptosystems, the public key of a user is derived from his identity information and his secret key is generated by a trust third party called Private Key Generator(PKG). When Alice generates a signature and sends it to Bob, Bob will be able to verify the signature by using Alice's identity information. The identity information, such as Alice's IP address or email address, can be obtained easily, therefore the public key management burden in traditional certificate based cryptosystems is reduced.

3.4.1 Generic Scheme

In general, an identity based cryptosystem has the following properties:

- user's public key is his identity, or derived from his identity.
- no requirement of the public key directory.
- message encryption and signature verification processes require only receiver's and signer's identity respectively along with some system parameters.

However, although public key distribution process is simplified, the key escrow problem that PKG knows the users' secret keys is still inherited. Besides, ID-based cryptosystems require users to authenticate themselves to the PKG in the same way as they authenticate themselves to Certificate Authority (CA).

The identity based signature scheme (IBS) contains four algorithms: *Setup*, *Extract*, *Sign*, *Verify*.

Setup. The master key and parameters generation algorithm, is a probabilistic algorithm that on input a security parameter 1^k, outputs a master key s and a parameter list $params$.

Extract. The signing key issuing algorithm, is a deterministic algorithm that on input a user's identity id and a master key s, returns a pair of matching public and secret keys (PK_{id}, SK_{id}).

Sign. The signing algorithm, is a probabilistic algorithm that on input a message m, returns an online signature σ.

Verify. The verification algorithm, is a deterministic algorithm that on input a message m, a user's identity id, a parameter list $params$, and a signature σ, returns 1 (*accept*) or 0 (*reject*).

3.4.2 Security Arguments

The security of traditional digital signature schemes is defined by Goldwasser *et al.* as existential unforgeability under adaptive chosen message attack, as described in previous section. Cha and Cheon [16] extended this notion to *existential unforgeability under adaptive chosen message and ID attack* for ID-based signature schemes. Assume the existence of a polynomial time adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, who cooperate to perform the following game:

1. $\mathcal{C}$ runs Setup algorithm of the scheme and sends the resulting system parameters to $\mathcal{A}$.

2. $\mathcal{A}$ issues the following queries as he wants:

   Message Hash Query $\mathcal{C}$ computes the value of the hash function for the request input and sends the value to $\mathcal{A}$.

   Extract Query Given an identity id and a message m, $\mathcal{C}$ returns the private key corresponding to id which is obtained by running Extract algorithm.

   Sign Query Given an identity id and a message m, $\mathcal{C}$ returns a signature which is obtained by running Sign.

3. $\mathcal{A}$ outputs (id, m, σ), where id is an identity, m is a message, and σ is a signature, such that (id, m) are not equal to the input of any query to Extract and Sign respectively. $\mathcal{A}$ wins the game if σ is a valid signature of m for id.

Formally, we have

Definition 19 *An adversary $\mathcal{A}$ (t, q_S, ϵ) breaks a signature scheme, if $\mathcal{A}$ runs in time at most t, makes at most q_S signature queries, and the advantage that $\mathcal{A}$ wins the game is at least ϵ. A digital signature scheme is (t, q_S, ϵ)-existentially unforgeable under*

chosen message and ID attack if no adversary (t, q_S, ϵ) *breaks it.*

$$\mathsf{AdvSig}_{\mathcal{A}}^{\mathsf{EF-CMA}} = \Pr\left[\mathsf{Verify}(\mathsf{id},\mathsf{m},\sigma) = \text{accept} \;\middle|\; \begin{array}{l} (s, \mathsf{params}) \leftarrow \mathsf{Setup}(1^{\mathsf{k}}); \\ (PK_{id}, SK_{id}) \leftarrow \mathsf{Extract}(\mathsf{id},\mathsf{s}); \\ for\ i = 1, 2, ..., q_S; \\ m_i \leftarrow \mathcal{A}(PK_i, m_i); \\ \sigma_i \leftarrow \mathsf{Sign}(\mathsf{SK_{id}},\mathsf{m}); \\ (id, m, \sigma) \leftarrow \mathcal{A}(PK_i, m_i, \sigma_i); \\ (id, m) \neq (id_1, m_1), ..., (id_{q_S}, m_{q_S}); \end{array} \right] \leq \epsilon$$

3.5 Online/Offline Signatures

The online/offline signature scheme was first introduced by Even, Goldreich and Micali in 1990 [26]. The motivation was to produce a digital signature scheme suitable for applications where pre-computations are tolerable, however signatures have to be generated as fast as possible once the message is presented. The notion of the online/offline signature scheme is splitting the signing phase to an online phase and an offline phase. The offline phase which is independent of the message can be performed in advance. Whereas the online phase is quickly performed when a message is given.

However, the computational overhead for generating a digital signature cannot be reduced by reducing the algorithm complexity. To enable quick signing in online phase, in designing the algorithm, the overall computational overhead has to be shifted to the offline phase. The online phase therefore undertakes the minimum computation.

3.5.1 Constructions Based on One-time Signatures

The first online/offline signature proposal [26] makes use of an ordinary digital signature scheme, a one-time signature scheme and a one-way hashing scheme to fulfill the construction. The essence of the construction is to use the ordinary signature scheme to sign a random instance of information enabling the one-time signature which will be used to sign the message in the future.

The one-time signature scheme was first introduced by Lamport in 1979. The original scheme is able to sign only one bit message. The signature is generated as followed.

1. Choose two values x_1 and x_2 at random as secret key component;
2. Compute $y_1 = h(x_1)$ and $y_2 = h(x_2)$ and public key component;

3. To sign a single bit message g, reveal the pre-image corresponding to the actual '0' or '1':

$$\sigma = \begin{cases} y_1 & \text{if g=0;} \\ y_2 & \text{if g=1;} \end{cases}$$

Thenceforward, one-time signature schemes have been broadly studied in the literature because of its advantage in efficient signing. The original scheme has been largely improved in order to sign messages that are more than one bit. For instance, Even, Goldreich and Micali's scheme [26] makes use of the property of the *extended square root* to construct an efficient one-time signature, which requires only one hash in signing. However, the size of the resulting signature and necessary public keys are comparatively large. For instance, Even *et al.*'s most efficient construction requires 33 DES keys (56-bit) and 4736-bit. Perrig's BiBa one-time signature [74] requires pre-computed hash chain for n nodes. Reyzin's short one-time signature HORS [82] reduces the signature size to 1280-bit, but still requires 81920-bit public key.

Since using one-time signature in signature generation, online-offline signatures inherit the problem of requiring large amount of public keys, or even constructing of hash chains. Therefore this type of online/offline signatures is not popularised in the literature. However, it brought up a inspiring point that later researchers could work on.

3.5.2 Other Construction Approaches

During the last 30 years, since the first scheme came into existence, the online/offline signature scheme has not been found much favour in the application domain, hence few new constructions have appeared in the literature. However, breakthroughs in shortening the public key size and the signature size have been made by various authors.

Construction Based on Trapdoor Hash Family

To overcome the problem of signature size, Shamir and Tauman [92] introduced a *Hash-Sign-Switch* paradigm to construct online/offline signature schemes. This novel approach combines any trapdoor hash family and any normal signature schemes to produce an online/offline signature scheme.

In their scheme, the offline phase consists of one trapdoor hash and an ordinary signature generation, whereas the online phase consists of a single collusion finding computation, which is efficient when the trapdoor key is known. To make the scheme the most efficient, in [92] the authors provide three constructions of the trapdoor hash

family which are comparatively efficient in collision finding. In addition, there are only two sets of public-secret key pairs needed, for normal signature and the trapdoor hash respectively. The signature size is reduced to the size of a normal signature, plus the size of a random trapdoor hash component.

The prosperity of elliptic curve and pairing based cryptographic primitives opens a new way for the design of more efficient online/offline signature schemes. In 2005, Zhang, Mu and Susilo [107] proposed a pairing based online/offline signcryption scheme. In their scheme, the signer performs scalar multiplications to generate offline signatures and stores them for future use. When the message arrives, the online signature is generated simply using a hash. The resulting signature size is comparative shorter since pairing based cryptosystem can provide a higher level of security with a shorter size.

3.6 Multisignature Schemes

The multisignature scheme was first introduced by Itakura and Nakamura [40] in 1983. It is an exception of a group signature in which the signature is generated cooperatively by many individuals for the same messages. Specially, the size of the signature is independent of the number of signers. To generate such signature, there exist two approaches in the domain.

- Generate the signature in a parallel manner. Each individual signer firstly generates a partial message that he is responsible for. Then later, the multisignature will be constructed by a "collector" after obtaining all the partial messages.
- Generate the signature in a sequential manner. All the signers are arranged in an order. The first signer generates his own signature and passes to the second signer. The second signer generates his owns signature according to the received signature and passes the new signature to the next signer, and so on. The required multisignature will be generated by the last signer.

Multisignatures have been extensively studied in the literature, but because of the absence of formal definition, many schemes have been proven to be insecure [50, 66].

3.6.1 Constructions based on RSA

Early multisignature schemes make use of RSA [85] based approach in constructions. However, multisignature schemes based on RSA inherits a problem called *moduli clashes* [47], which results from different users using different RSA moduli domains.

For instance, users $1, 2, ..., k$ want to jointly sign a contract so as to achieve the authentication, integrity and non-repudiation properties. Let d_i be the ith user's secret key, and (e_i, n_i) be the public key, where $n_i = p_i \times q_i$, p_i, q_i are large primes, and $e_i \times d_i \equiv 1 \ mod \ \phi(n_i)$. In order to generate a signature on message m, signers compute

$$\begin{aligned} \sigma_1 &= m^{d_1} \ mod \ n_1 \\ \sigma_2 &= \sigma_1^{d_2} \ mod \ n_2 \\ &... \\ \sigma_k &= \sigma_{k-1}^{d_k} \ mod \ n_k \end{aligned}$$

However, this scheme is unlikely to work in practice since the domain of each signers modulus n_i are likely to be different. if $n_i > n_{i+1}$, the message domain will become too large for the $(i+1)$th user to sign. The solutions to this problem are:

1. Reblock the signed message which exceeds the modulus domain;
2. Predetermine the signing order according to the modulus domain.

Harn and Kiesler [45] has proposed a scheme to address the moduli clashes problem by predetermining the signing order according to the signers' moduli domain. Nevertheless, it is difficult in reality when the signer group is determined dynamically.

Chang *et al.* [17] proposed an ID-based multisignature scheme which overcomes this problem. Their scheme is generated under the same modulus N for different signers. It solves the moduli clashes problem without using message reblocking and predefining signing order. Besides, the identity based scheme simplifies the public key certification process. However, the tradeoff is great computational overhead resulting from computing modulo exponentiations. The signing algorithm requires one modulo exponentiations and the verification algorithm requires three modulo exponentiations. This scheme is comparatively inefficient in the application domain.

3.6.2 Accountable Subgroup Multisignature Scheme

In 2001, Micali *et al.* [61] provided the first formal definition of multisignature which is called Accountable Subgroup Multisignature (ASM). In essence, ASM schemes enable any subgroup S, of a given group of potential signers, to efficiently sign a message so that the signature provably reveals the identities of the signers in the subgroup to any verifier. The signing process is in a parallel manner, where each signer has to generate a partial message for the construction of the multisignature. However, the role of the

collector can be performed by each signer after the receipt of all the partial messages from other signers. In the end, each co-signer will be able to generate a copy of the authentic multisignature jointly signed by the group. Informally, we have:

Definition 20 *An accountable-subgroup multisignature of a subgroup of signers S for a message M provides, without any trusted managers or third parties, a self-contained and universally verifiable proof of (1) the composition of S and* (2) *the fact that each member of S stood behind M.*

Besides, an ASM scheme is supposed to satisfy two properties:

Flexibility Any subgroup S of G may easily jointly sign a document. It is then up to the verifier of the signature to decide whether S is sufficient for the signature to be deemed valid.

Accountability without use of trusted third parties, individual signers can be identified from the signed document.

Their scheme is based on Schnorr's signature scheme, which is generally regarded as provable and efficient:

- Only three rounds of communication are required per signature;
- The signing time per signer is the same as for the single-signer Schnorr scheme, regardless of the number of signers;
- The verification time is only slightly greater than that for the single-signer Schnorr scheme;
- The signature length is the same as for the single-signer Schnorr scheme, regardless of the number of signers.

The resulting signature inherits the advantages of Schnorr signature scheme and has the following desirable properties:

1. The signature length does not grow with the number of signers: it is the same as that of a single-signer Schnorr signature.
2. The verification is almost the same as the time needed to verify a single Schnorr signature.
3. The signing protocol requires only three rounds of communication among the members of the subgroup, irrespective of the size of the subgroup.

4. The signing time per member of the subgroup is almost the same as the time required to produce a single-signer Schnorr signature.

5. Only the key generation protocol (done once by the whole group) requires each member of the group to perform communication and computation that is linear in the size of the group.

The security of the scheme is based on the DLP' assumption [61]. Since there is no trusted third party in the scheme, the public keys are authenticated using the Merkle Tree [58]. In the key generation phase, a Merkle Tree is constructed to contain all the public value and attach the path to each of them along with the public key. The multisignature is generated by a collector in the original description. However, it is also possible to regard the collector as each individual signer, who, after the receipt of all the partial messages, construct an identical copy of the signature.

3.6.3 Security Arguments

The security of the accountable subgroup multisignature scheme can be defined by considering the capabilities of a forger $\mathcal{F}$.

1. $\mathcal{F}$ fully controls all message exchanged in the network: whether the sender or the recipient is good or bad, it can read any messages sent, modify it or prevent its delivery. In addition, $\mathcal{F}$ can send any messages needed on behalf of any players, i.e. there are no private authenticated channels - all players communicate via the adversary.

2. $\mathcal{F}$ can corrupt any players at any time, during both key generation and signing. Upon corrupting one single player, $\mathcal{F}$ learns the entire internal state of this player, including all secret information and past coin tosses.

3. $\mathcal{F}$ controls the input of any uncorrupted players during key generation.

4. For any uncorrupted player, $\mathcal{F}$ can conduct an adaptive chosen-message-and-subgroup attack. At any time, it can request that the player executes the signing protocol on some specified message with some specified subgroup co-signers.

Formally, we have

Definition 21 *An ASM is secure if, for all constants $c > 0$ and all sufficiently large security parameters k, no polynomial-time adversary has better than k^{-c} chance of outputting a triple (σ, M, S) such that:*

- *σ is a valid signature on the message M by the subgroup S of players;*
- *there exists an uncorrupted player $P \in S$ who has never been asked by $\mathcal{F}$ to execute the signing protocol M and S.*

3.7 Summary

Recent crypto-analytic advances have caused increasing discussion about public key sizes and the security required. It has been argued that the security provided by 1024-bit RSA is getting insufficient for the current situation. Yet, moving to larger key sizes also brings up arguments regarding the memory limitations and other costs used to protect keys. The emergence of elliptic curve cryptography defends the use of short keys by providing higher computational complexity with comparatively small key sizes. According to Vanstone's [95] survey, to provide 256-bit level of security, RSA requires 15360-bit public keys, whereas ECC only requires 512-bit keys. In addition, it is also possible to reduce the signature size using ECC as in Boneh *et al.*'s pairing based short signature scheme, where the signature size is reduced to 160-bit. In our routing paradigm, shorter key size and signature size can reduce the data transmission overhead. In turn, the network performance can be enhanced by this means.

Chapter 4

ID-based Online/Offline Signature Schemes

In this chapter, we describe the notion of identity based online/offline signature (IOS) schemes. We begin with introducing the generic scheme, explaining the algorithms involved. Its security requirements and the attack model are defined accordingly. We then give two concrete constructions of the ID-based online/offline signature schemes of our proposal along with the proofs.

4.1 Generic Scheme

We derive our identity based online/offline signature scheme (IOS) from the online/offline signature scheme introduced in section 3.5. We also formalise the notations in description, which will be used in the next chapter to construct the transformation.

Definition 22 *ID-based online/offline digital signature scheme $\mathcal{DS}$ is comprised of five polynomial time algorithms:* IO_ParamGen, IO_Ext, IO_OffSign, IO_OnSign, *and* IO_Verify.

IO_ParamGen. *The master key and parameter generation algorithm, is a probabilistic algorithm that on input a security parameter 1^k, outputs a master key $IOSK^*$ and a parameter list params.*

IO_Ext. *The signing key issuing algorithm, is a deterministic algorithm that on input a user's identity id and a master key $IOSK^*$, returns a pair of matching public and secret keys $(iopk_{id}, iosk_{id})$.*

IO_OffSign. *The offline signing algorithm, is a probabilistiqalgorithm that on input a parameter list* params *and a signing key $iosk_{id}$, outputs an offline signature S.*

IO_OnSign. *The online signing algorithm, is a probabilistic algorithm that on input a message m and an offline signature S, returns an online signature σ.*

IO_Verify. *The verification algorithm, is a deterministic algorithm that on input a message m, a user's identity id, a parameter list $params$, an offline signature S, and an online signature σ, returns 1 (*accept*) or 0 (*reject*).*

4.2 Security Arguments

The security of the ID based online/offline signature is the *existential unforgerability under chosen-message and ID attacks*. We follow Cha-Cheon's [16] classic description of ID based signature scheme and have the following definitions.

Definition 23 *An identity based online/offline signature is said to be existentially unforgeable under chosen-message and ID attacks if no probabilistic polynomial time adversary has a non-negligible advantage in this game:*

1. *The challenger $\mathcal{A}$ runs the setup algorithm to generate the system parameters and sends them to the adversary $\mathcal{F}$.*

2. *The adversary $\mathcal{F}$ performs the following queries:*

   - ***ID Hash Query**: For any given identity ID, $\mathcal{F}$ will produce the corresponding public key Q_{ID} to the identity.*
   - ***Key Extraction Query**: For any given identity ID, $\mathcal{F}$ will produce the corresponding secret key D_{ID}.*
   - ***Message Hash Query**: For any given random value r, $\mathcal{F}$ will produce the corresponding hash value.*
   - ***Offline Signing Query**: For any given identity ID and secret key D_{ID}, $\mathcal{F}$ will produce an offline signature S corresponding to the ID.*
   - ***Online Singing Query**: For any given identity ID, message m, and an offline signature, this query will produce an online signature σ corresponding to the ID.*

3. *After a polynomial number of queries, $\mathcal{F}$ produces a tuple $(ID^*, m^*, S^*, \sigma^*)$ of identity ID^*, whose secret key was never asked in the key extraction query. Besides, the pair (ID^*, m^*) was never asked in online/offline signing queries.*

The success probability of winning the above game is defined by $\mathsf{Succ}_{\mathcal{A}}^{EF-IOS-CMA}(\ell)$. An online/offline signature scheme is secure if the success probability of the above attack

is negligible.

$$\mathsf{Succ}_{\mathcal{A}}^{EF-IOS-CMA}(\ell) \leq \epsilon,$$

where ϵ is negligible.

Formally, we have

Definition 24 ***(Security)*** *[28] The ID-based online/offline signature scheme $\mathcal{S} =<$ IO_ParamGen, IO_Ext, IO_OffSign, IO_OnSign IO_Verify $>$ is existentially unforgeable under adaptive chosen message attacks if it is infeasible for a forger to produce a valid message-signature pair after obtaining polynomially many signatures on a message of its choice from the signer.*
Formally, for every probabilistic polynomial forger $\mathcal{A}$ such that:

$$\mathsf{Adv}(\mathcal{A}) = \Pr\left[\begin{array}{l} (pk, sk) \leftarrow \mathsf{IO\_Ext}(1^k); \\ for\ i = 1, 2, ..., k; \quad r_i \in_R \mathbb{Z}_q; \\ m_i \leftarrow \mathcal{A}(pk, m_1, S_1, \sigma_1, ..., m_k, S_k, \sigma_k); \\ S_i \leftarrow \mathsf{IO\_OffSign}(sk, r); \\ \sigma_i \leftarrow \mathsf{IO\_OnSign}(S_i, m); \\ (m, S, \sigma) \leftarrow \mathcal{A}(pk, m_1, S_1, \sigma_1, ..., m_k, S_k, \sigma_k); \\ m \neq m_1, ..., m_k \ and\ \mathsf{IO\_Verify}(pk, m, S, \sigma) = \text{accept}; \end{array}\right] \leq \epsilon$$

4.3 A Concrete Construction

In this section, we provide a concrete ID-based online/offline signature scheme [101]. We also prove the security of these two signature schemes. Our scheme involves four algorithms: Setup, Extract, OffSign, OnSign and Verify.

Setup. Given $\mathbb{G}_1$ and its generator P, pick a random $s \in \mathbb{Z}_q^*$, and set $P_{pub} = sP$. Choose a cryptographic hash function $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. The system parameters are (P, P_{pub}, H_0, H_1). The master key is s. H_0 and H_1 behave as random oracles.

Extract. Given an identity ID, the algorithm computes $D_{ID} = sH_0(ID)$ and outputs it as the private key related to ID corresponding to $Q_{ID} = H_0(ID)$.

OffSign. Given a secret key D_{ID}, pick random $r, x \in \mathbb{Z}_q^*$, output the offline signature pair (S, R), where $S = D_{ID} - xP_{pub}$, $R = rP$.

OnSign. Given a message m and offline signature S, compute the online signature as $\sigma = H_1(m)r + x$. The resulting signature is a triple (S, σ, R).

Verify. Given a signature tuple (S, σ, R) of a message m for an identity ID, check whether $(P_{pub}, S + \sigma P_{pub}, P, Q_{ID} + H_1(m)R)$ is a valid Diffie-Hellman tuple.

4.3.1 Analysis

Similar to the previous scheme, we discuss the correctness and efficiency of our scheme.

Correctness: The correctness can be easily proved as follow:

$$\begin{aligned} e(S + \sigma P_{pub},\ P) &= e(D - xP_{pub} + (H_1(m)r + x)P_{pub},\ P) \\ &= e(D - xP_{pub} + H_1(m)rP_{pub} + xP_{pub},\ P) \\ &= e(D + H_1(m)rP_{pub},\ P) \\ &= e(s(Q_{ID} + H_1(m)rP),\ P) \\ &= e(Q_{ID} + H_1(m)R,\ P_{pub}) \end{aligned}$$

Signature Size: The resulting signature is a tripe (S, σ, R). We assume the safe length of GDH group $\mathbb{G}_1$ is ρ, the size of each element in a signature tuple is $\log \rho$, $\log q$, and $\log \rho$. Therefore, the total length is $2 \log \rho + \log q$.

Performance: Similarly, the online signing phase require one hash. The offline phase of this scheme requires two pairings which are both variables. Compared with the previous scheme, this scheme is less efficient in verification. However, the structure of the second scheme has better properties for signature aggregation, which will be shown in the next chapter.

4.3.2 Security Proof

To prove our scheme is existentially unforgeable under adaptive chosen-message attack, we use Libert and Quisquater's proof technique [51].

Theorem 1 *In the random oracle model, if a probabilistic polynomial time forger $\mathcal{F}$ has an advantage ε in forging an online/offline signature with running time t and asking H_0,H_1,key extraction oracle and online/offline signing oracle q_{H_0}, q_{H_1}, q_e and q_s times respectively, then the CDH problem can be solved with an advantage*

$$\varepsilon' > (\frac{1}{q_e} \cdot (1 - \frac{1}{q_e + 1})^{q_e+1})(\varepsilon - \frac{q_S(q_{H_1} + q_S) + 1}{2^k})$$

with running time $t' < t + (q_{H_0} + q_e + 2q_s)t_m$, where t_m is the time to compute a scalar multiplication in $\mathbb{G}_1$.

Proof Firstly we assume the existence of a forger $\mathcal{F}$, which by performing queries, finally produces a valid online/offline signature tuple. On the other hand, a probabilistic polynomial time algorithm - attacker $\mathcal{A}$ which answers all the queries asked by forger $\mathcal{F}$ finally solves the CDH problem. We further assume $(aP, bP) \in \mathbb{G}_1 \times \mathbb{G}_1$ is a random instance of the CDH problem taken as input by attacker $\mathcal{A}$. The system public key is initialised as $P_{pub} = aP$. Then $\mathcal{A}$ answers all the queries as follows:

ID hash query: when an identity ID is submitted to H_0 oracle, $\mathcal{A}$ flips a coin $T \in 0, 1$ which yields 1 with probability δ and 0 with probability $1 - \delta$. $\mathcal{A}$ then randomly chooses $u_i \in \mathbb{Z}_q^*$. If $T = 0$, $\mathcal{A}$ sets the value Q_i as $u_i P$. Otherwise, Q_i is set as ubP. $\mathcal{A}$ records the tuple (ID_i, u_i, T_i) in a list L_0, and returns Q_i as the answer.

Key extraction query: when $\mathcal{A}$ receives a key extraction query, it firstly checks whether the corresponding tuple (ID_i, u_i, T_i) exists in L_0. If it does not exist, $\mathcal{A}$ outputs "failure" and halts. If it exists $\mathcal{A}$ further checks the value of T_i. If $T_i = 0$, it computes the secret key as $uP_{pub} = uaP$ and returns it to $\mathcal{F}$. Otherwise, it outputs "failure" and halts.

Message hash query: $\mathcal{A}$ maintains a list L_1 for message hash queries. When a tuple (Q_i, m_i) is submitted to H_1 oracle, $\mathcal{A}$ firstly checks the existence of the tuple in L_1. If it exists, the value will be returned. Otherwise, $\mathcal{A}$ randomly chooses $v_i \in \mathbb{Z}_q^*$, stores the tuple (Q_i, m_i, v_i), and returns v_i to $\mathcal{F}$ as the answer.

Offline signing query: $\mathcal{A}$ randomly chooses $\alpha_i, t_i, \beta_i \in \mathbb{Z}_q^*$ and defines offline signature as $S = (t_i - \alpha_i)P_{pub} = (t_i - \alpha_i)aP$, $R_i = \beta_i P$. The pair (S_i, R_i) and α_i are stored for future use.

Online signing query: when $\mathcal{A}$ receives an online signing query on message M_i for an identity ID_i, it firstly retrieves the corresponding u_i from L_0. The previously computed offline signature tuple (S_i, R_i) and value α_i is also retrieved. Then it defines the message hash value $H_1(m_i, R_i) = \beta_i^{-1}(t_i - u_i)$, and the online signature as $\sigma_i = \alpha_i$. If the message hash value has been defined before, $\mathcal{A}$ output "failure" and halts. Otherwise, the signature tuple (S_i, σ_i, R_i) is returned to $\mathcal{F}$.

The resulting signature tuple passes the verification since:

$$\begin{aligned} e(S_i^* + \sigma_i^* P_{pub}, P) &= e(t_i P_{pub} - \alpha_i P_{pub} + \alpha_i P_{pub}, P) \\ &= e(t_i aP, P) \end{aligned}$$

$$\begin{aligned} e(Q_i^* + H_1(m_i^*)R_i^*, P_{pub}) &= e(u_iP + (\beta_i^{-1}(t_i - u_i))\beta_iP, P_{pub}) \\ &= e(t_iP, aP) \end{aligned}$$

Eventually, the forger $\mathcal{F}$ produces a valid signature tuple (S^*, σ^*, R^*) for message M^* of identity ID^* and gives it to $\mathcal{A}$. $\mathcal{A}$ firstly recovers the tuple (ID^*, u^*, T^*) in list L_0 to check the value of T. if $T = 0$, $\mathcal{A}$ outputs "failure" and halts. Otherwise, the entry of (Q^*, m^*, v_i) must be in the list L_1 with overwhelming probability. If this entry does not exist, $\mathcal{A}$ outputs "failure" and halts. As the resulting signature tuple is valid, the following equation holds:

$$e(S^* + \sigma^* P_{pub}, P) = e(Q_{ID}^* + H_1(m^*)R^*, P_{pub}) \tag{4.1}$$

Besides we have $H_1(m^*) = v_iP$, $P_{pub} = ap$, and $Q_i = u_ibP$. According to (1) we can get:

$$\begin{aligned} e(S^* + \sigma^* aP, P) &= e(ubP + v_iR, aP) \\ e(S^* + \sigma^* aP, P) &= e(ubP, ap)e(v_iR, aP) \\ e(S^* + \sigma^* aP - v_iR, P) &= e(ubP, aP) \end{aligned}$$

The solution to the CDH instance (aP, bP) is $u^{*-1}(S^* + \sigma^* aP - v_iR)$.

Similar to Libert and Quisquater's analysis, $\mathcal{A}$'s probability of success involves three parts. Firstly, $\mathcal{A}$'s probability of failure caused by a conflict over H_1 is at most $q_S(q_{H_1} + q_S)/q$. Secondly, since H_1 is a random oracle, the probability of producing a valid forgery without asking $H_1(m^*)$ is $1/2^k$. Finally, the probability of $\mathcal{A}$ succeeds in a key extraction query is $\delta(1-\delta)^{q_e}$. The function $\delta(1-\delta)^{q_e}$ is maximized at $\delta = 1/(q_e+1)$. Thus we have the result

$$\begin{aligned} \delta(1-\delta)^{q_e} &= \frac{1}{q_e+1} \cdot (1 - \frac{1}{q_e+1})^{q_e} \\ &= \frac{1}{q_e} \cdot (1 - \frac{1}{q_e+1})^{q_e+1} \end{aligned}$$

Eventually it comes that $\mathcal{A}$'s advantages is at most

$$(\frac{1}{q_e} \cdot (1 - \frac{1}{q_e+1})^{q_e+1})(\varepsilon - \frac{q_S(q_{H_1} + q_S) + 1}{2^k})$$

□

4.4 A Better Construction

We give another concrete construction of the ID-based online/offline signature scheme [100] based on the general scheme. It consists of five algorithms: Setup, Extract, OffSign, OnSign, Verify.

Setup. Given $\mathbb{G}_1$ and its generator P, pick a random $s \in \mathbb{Z}_q^*$, and set $P_{pub} = sP$. Choose a cryptographic hash function $H_0 : \{0,1\}^* \to \mathbb{G}_1$ and $H_1 : \{0,1\}^* \times \mathbb{G}_1 \to \mathbb{Z}_q^*$. The system parameters are (P, P_{pub}, H_0, H_1). The master key is s. H_0 and H_1 behave as random oracles [8].

Extract. Given an identity ID, the algorithm computes $D_{ID} = sH_0(ID)$ and output it as the private key related to ID corresponding to $Q_{ID} = H_0(ID)$.

OffSign. Given a secret key D_{ID}, pick a random number $r \in \mathbb{Z}_q^*$ and a random secret number $x \in \mathbb{Z}_q^*$, output the offline signature pair (S, R), where $S = \frac{1}{r}D_{ID}$, $R = xP$.

OnSign. Given a message m and offline signature S, compute the online signature as $\sigma = H_1(m, R)x + r$. The resulting signature is a triple (S, σ, R).

Verify. Given a signature tuple (S, σ, R) of a message m for an identity ID, check whether $(P_{pub}, \sigma P - H_1(m, R)R, S, Q_{ID})$ is a valid Diffie-Hellman tuple.

4.4.1 Analysis

In this section, we will discuss the correctness and efficiency of our scheme.

Correctness: The correctness can be easily proved as follow:

$$\begin{aligned} e(\sigma P - H_1(m,R)R, S) &= e(xH_1(m,R)P + rP - H_1(m,R)R, \ \frac{1}{r}D_{ID}) \\ &= e(H_1(m,R)xP + rP - H_1(m,R)xP, \ \frac{1}{r}D_{ID}) \\ &= e(rP, \ \frac{1}{r}sQ_{ID}) \\ &= e(P_{pub}, \ Q_{ID}) \end{aligned}$$

Signature Size: The resulting signature is the tripe (S, σ, R). We assume the safe length of GDH group $\mathbb{G}_1$ is ρ, the size of point over $\mathbb{G}_1$ is $\lceil \log \rho \rceil$, according to the description in section 3.1.2. Thus the size of each element in a signature tuple is $\log \rho$, $\log q$, and $\log \rho$. Therefore, the total length is $2 \log \rho + \log q$. We believe

this size is irreducible since the first two elements are required in all the standard ID-based signature scheme.

Performance: Obviously, the online phase of our scheme is very efficient, which only requires one hash. The computational workload is passed to the offline phase. The signature verification is done through two pairing operations, which is the most expensive part in our scheme. However, since $e(P_{pub}, Q_{ID})$ is a constant, it only needs to be computed once and stored for future use.

4.4.2 Security Proof

To prove our scheme is secure against *adaptive chosen message and ID attack*, the problem is firstly reduced to a *given* ID attack. The adversary $\mathcal{A}$, who is given system parameters and a fixed ID, aims to output a valid signature corresponding to that ID. If no polynomial time algorithm $\mathcal{A}$ can win this game under a non-negligible advantage, we say this scheme is secure against *adaptive chosen message and fixed ID attack*. Then we give algorithm A the access to the extraction oracle to obtain different IDs. If no polynomial time algorithm $\mathcal{A}$ has non-negligible advantage in this game, we say this scheme is secure against *adaptive chosen message and ID attack*. Specifically, we intend to view the scheme as an ordinary ID-based signature scheme which output two signatures. Since the online signing phase does not use the ID information, it is viewed as a sub-phase of the ID-based offline signing phase.

Lemma 1 *Let $\mathcal{A}_0$ be an algorithm for an adaptive chosen message and ID attack to our scheme with running time t_0 and advantage ϵ_0, then there is an algorithm $\mathcal{A}_1$ for an adaptive chosen message and given ID attack which has running time $t_1 \leq t_0$ and advantage $\epsilon_1 \leq \epsilon_0(1-\frac{1}{q})/q_{H_0}$, where q_{H_0} is the maximum number of queries to ID hash oracle H_2 asked by $\mathcal{A}_0$.*

Proof We assume that the number of queries to message hash oracle, extraction oracle and online signing oracle are q_{H_1}, q_E, and q_S. Algorithm $\mathcal{A}_1$ is performed as follow:

1. Randomly choose $l \in \{1, ..., q_{H_0}\}$. Let ID_i denote the input of i^{th} q_{H_0} query asked by $\mathcal{A}_0$. Set ID'_i be ID^* if $i = l$, and ID_i otherwise. Define $H'_0(ID_i)$, $\mathsf{Extract}'(ID_i)$, $\mathsf{Sign}'(ID_i, m)$ to be $H_0(ID'_i)$, $\mathsf{Extract}(ID'_i)$, $\mathsf{Sign}(ID'_i, m)$. Notice that the Sign includes $\mathsf{OffSign}$ and OnSign. However, only the offline signing part is considered in ID attack,since the online signing part does not use any ID information.

2. Run $\mathcal{A}_0$ with the given system parameters. $\mathcal{A}_1$ responds to $\mathcal{A}_0$'s queries to H_0, H_1, Extract, and Sign by evaluating H_0', H_1, Extract', and Sign', respectively. Let the output of $\mathcal{A}_0$ be (ID_{out}, m, S, σ).

3. If $ID_{out} = ID^*$ and (ID_{out}, m, S, σ) is valid, it outputs (ID^*, m, S, σ). Otherwise outputs a fail.

   Since the probability distributions provided by H_0', Extract', and Sign' are indistinguishable from those produced by H_0, Extract, and Sign, $\mathcal{A}_0$ learns nothing from the query result. Besides, the probability that $\mathcal{A}_0$ produces a valid message signature pair (ID, m, S, σ) without any query of $H_0'(ID)$ is greater than $(1-\frac{1}{q})$. Hence, we can say $\mathcal{A}_0$ wins the game with advantage $\geq \epsilon(1-\frac{1}{q})/q_{H_0}$, where ϵ is an upper bound of success.

Lemma 2 *If there is an algorithm $\mathcal{A}_1$ for an adaptive chosen message and given ID attack to our scheme which queries H_1, H_2, Sign and Extract at most q_{H_1}, q_{H_2}, q_S and q_E times respectively, and has running time t_1 and advantage $\epsilon_1 \geq 10(q_S+1)(q_S+q_{H_1})/q$, then CDHP can be solved with probability $\epsilon_2 \geq 1/9$ within running time $t_2 \leq 23 q_{H_1} t_1/\epsilon_1$.*

Proof We assume that for any ID, $\mathcal{A}_1$ queries $H_0(ID)$ and Extract at most once. We have an algorithm $\mathcal{A}_1$, through interacting with a signing simulator $\mathcal{B}$, computes abP for a randomly given instance (P, aP, bP) where P is a generator of G.

1. Fix an identity ID and put $P_{pub} = aP$. Randomly choose $\alpha_i \in \mathbb{Z}_q^* \; for \; i = 1, ..., q_E$ and $\beta_j, x_j \in \mathbb{Z}_q^* \; for \; j = 1, ..., q_S$. Let $ID_i \; and \; ID_{i_k}$ denote the input of the i^{th} H_0 query and the k^{th} Extract query. We define:

$$\begin{aligned}
H_0''(ID) &= \begin{cases} bP & \text{if } ID_i = ID^*, \\ \alpha_j P & \text{otherwise;} \end{cases} \\
\mathsf{Extract}''(ID_{i_k}) &= \alpha_{i_k}(bP); \\
\mathsf{OffSign}''(m_j, x_j,) &= (m_j, h_j, \sigma_j), \; where \; h_j = H_1(m, R_j), \; \sigma_j = h_j x_j + \frac{a}{\beta}; \\
\mathsf{OnSign}''(ID_{i_j}) &= (ID_{i_j}, R_j, S_j), \; where \; R_j = x_j P.
\end{aligned}$$

   The resulting signature is $(ID_j, m_j, R_j, S_j, \sigma_j)$. We observed that $(bP, \sigma P - Rh, S, aP)$ is valid Diffie-Hellman tuple since:

$$\begin{aligned}
e((hx + \frac{a}{\beta})P - hR, S) &= e(hxP - \frac{a}{\beta}P - hxP, \beta bP) \\
&= e(\frac{a}{\beta}P, \beta bP) \\
&= e(aP, bP)
\end{aligned}$$

2. We apply the oracle replay attack invented by Pointcheval and Stern in [79].

   (a) $\mathcal{A}_1$ firstly asks q_{H_1} distinct queries to the random oracle f, obtaining $\rho_1, ..., \rho_{q_{H_1}}$ answers respectively. Assume there is a simulator $\mathcal{B}$ which simulates the activity of the signer without the knowledge of the secret key. For each query of message m_j it output a series of signature message pairs in the form of $(ID_j, m_j, R_j, h_j, S_j, \sigma_j)$. Then algorithm $\mathcal{A}_1$ assumes that $f(m_j, R_j) = h_j$ and stores it.

   (b) If following collisions appear:

   - A (m_j, R_j) pair produced by $\mathcal{B}$ also appears in the list of questions to random oracle asked by $\mathcal{A}_1$;
   - $\mathcal{B}$ produces two (m_j, R_j) pairs which are exactly the same;

   $\mathcal{A}_1$ simply outputs fail and aborts. If no collision appeared, $\mathcal{A}_1$ outputs a valid message signature pair, which is expected to be valid for the fixed ID.

   (c) By replaying $\mathcal{B}$ with the same messages but different choice of H_1, we can obtain two valid signatures (ID, m, R, h, S, σ) and $(ID, m, R, h', S, \sigma')$, where $h \neq h'$. Notice that offline signatures are supposed to be the same since it is closely related to the value of r.

   (d) If both outputs are valid, compute $x = \frac{\sigma - \sigma'}{h - h'}$.

3. Since $(Q_{ID_j}, \sigma_j P - R_j h_j, S_j, P_{pub})$ is valid Diffie-Hellman tuple, we can compute α through $\beta = \frac{a}{\sigma - h_j x_j}$. Apply β_j to S_j, we have

$$\begin{aligned} S &= \frac{a}{\sigma - h_j x_j}(bP) \\ S &= \frac{abP}{\sigma - h_j x_j} \\ abP &= S(\sigma - h_j x_j) \end{aligned}$$

Combining Lemma 1 and 2, we have the following theorems.

Theorem 2 *If there is an algorithm $\mathcal{A}_0$ for an adaptive chosen message and ID attack to our scheme which queries H_0, H_1, Sign and Extract at most q_{H_0}, q_{H_1}, q_S and q_E times respectively, and has running time t_1 and advantage $\epsilon_0 \geq 10(q_S+1)(q_S+q_{H_1})q_{H_0}/(q-1)$, then CDHP can be solved with probability $\geq 1/9$ within running time $\leq \frac{23 q_{H_0} q_{H_1} t_0}{\epsilon_0(1-\frac{1}{q})}$.*

Using another variant of the forking lemma [79], we have the following result:

Theorem 3 *If there is an algorithm $\mathcal{A}_1$ for an adaptive chosen message and given ID attack to our scheme which queries H_0, H_1, Sign and Extract at most q_{H_0}, q_{H_1}, q_S and q_E times respectively, and has running time t_1 and advantage $\epsilon_1 \geq 10(q_S+1)(q_S+q_{H_1})/q$, then CDHP can be solved within expected time $\leq 120686 q_{H_1} t_1/\epsilon_1$.*

Theorem 4 *If there is an algorithm $\mathcal{A}_0$ for an adaptive chosen message and ID attack to our scheme which queries H_0, H_1, Sign and Extract at most q_{H_0}, q_{H_1}, q_S and q_E times respectively, and has running time t_1 and advantage $\epsilon_1 \geq 10(q_S+1)(q_S+q_{H_1})q_{H_0}/(q-1)$, then CDHP can be solved within expected time $\leq \frac{120686 q_{H_0} q_{H_1} t_0}{\epsilon_0(1-\frac{1}{q})}$.*

4.5 Summary

Online/offline signature schemes enable the efficient signing property by reasonably arranging the signing calculations into two parts. In our proposed schemes, the costly scalar multiplications are performed when the signer is idle, whereas the message related signatures are generated using hash with proper parameters. The resulting signature is a triple: the online signature, offline signature, and the necessary commitment, which is actually the linkage between two signatures. The size of the signature is irreducible because the use of identity based schemes.

Verification of both schemes takes two pairing computations. Since the first scheme computes the signature tuples against a constant value, this scheme is a little more efficient than the second scheme, which requires two pairing computations for each round. Yet, the second scheme can be efficiently transformed into an ID-based multisignature scheme discussed in the next chapter.

Chapter 5

ID-based Multisignature Schemes

In this chapter, we describe our ID-based multisignature scheme (IBMS) [102], which is constructed from the ID-based online/offline signature scheme (IOS). Firstly, we define the generic scheme and its security. Then we provide the generic transformation from IOS to IBMS and prove the security. Finally, we make use of the generic transformation to construct our concrete IBMS from the second construction of IOS described in the previous chapter.

5.1 Generic Scheme

According to Micali *et al.* [61], we extend the definition of ASM to ID-based Multisignature. Our scheme does not satisfy the accountability criterion because the ID-based signature scheme requires the existence of Key Generation Centre (KGC), which is a trusted third party. Similarly to section 4.2, we formalise the notations in our description.

Definition 25 *We assume that the subgroup G_{Sub} consists of L signers. An ID-based multisignature consists of four components:* IM_ParamGen, IM_KeyGen, IM_Sign, IM_Verify.

IM_ParamGen. *The master key and parameter generation algorithm, is a probabilistic algorithm that on input a security parameter 1^k, outputs a master key $IMSK^*$ and a parameter list params.*

IM_KeyGen. *The signing key issuing algorithm, is a probabilistic algorithm that on input a subgroup G_{Sub}, a user's identity id and a master key $IMSK^*$, returns a pair of matching public and secret keys $(impk_{id}, imsk_{id})$ for each user in the group.*

IM_Sign. *The signing algorithm, is a probabilistic algorithm that when the following are input from each signer:*

1. *a description of subgroup* G_{Sub}
2. *the public key* $impk_i$ *of each member in* G_{Sub}
3. *the message* m
4. *the signer's secret key* $imsk_i$

produces a signature σ *which is generated jointly by all the members of* G_{Sub}.

IM_Verify. *The verification algorithm, is a deterministic algorithm, on input the following*

1. *a description of subgroup* G_{Sub}
2. *the public key* $impk_i$ *of each member in* G_{Sub}
3. *the message* m
4. *the signature* σ

*outputs 1 (*accept*) or 0 (*reject*).*

5.2 Security Arguments

The security definition of ID-based multisignature can be adapted from the ID-based online/offline signature scheme.

Definition 26 *An ID-based multisignature (IBMS) of subgroup* $S \subseteq G$ *is said to be existentially unforgeable under chosen-message attacks if no probabilistic polynomial time adversary has a non-negligible advantage in producing a tuple* (σ, m, S) *such that:*

1. *The challenger* $\mathcal{A}$ *runs the setup algorithm to generate the system parameters and sends them to the adversary* $\mathcal{F}$.
2. *The adversary* $\mathcal{F}$ *performs the following queries:*
   - ***ID Hash Query*** $\mathbf{O}_{\mathbf{IDH}}^{\mathbf{IM}}$ *For any given identity ID,* $\mathcal{F}$ *will produce the corresponding public key* Q_{ID} *to the identity.*
   - ***Key Generation Query*** $\mathbf{O}_{\mathbf{KGN}}^{\mathbf{IM}}$*:* $\mathcal{F}$ *produces an identity* ID *of the uncorrupted player in* S *and receives corresponding secret*

     key D_{ID} *and its temporary signing commitment* S *for current signing session.*

- ***Signing Query*** $\mathbf{O}_{\mathbf{Sign}}^{\mathbf{IM}}$*: $\mathcal{F}$ produces a message m, and receives a signature generated by signing oracle using the secret key corresponding to ID.*

3. *After a polynomial number of queries, $\mathcal{F}$ produces a tuple (m^*, σ^*, S) such that*

   - *σ^* is a valid signature on the message m by the subgroup S of players.*
   - *there exists an uncorrupted player $P^* \in S$ who has never been asked by $\mathcal{F}$ to execute the signing query on m and S.*

The success probability of winning the above game is defined by $\mathsf{Succ}_{\mathcal{A}}^{EF-IBMS-CMA}(\ell)$. An ID based multisignature scheme is secure if the success probability of the above attack is negligible. In other words,

$$\mathsf{Succ}_{\mathcal{A}}^{EF-IBMS-CMA}(\ell) \leq \epsilon,$$

where ϵ is negligible.

Formally, we have

Definition 27 ***(Security)*** *The ID-based multisignature scheme $\mathcal{S} =$ <IM_ParamGen, IM_KeyGen, IM_Sign, IM_Verify> is existentially unforgeable under adaptive chosen message and ID attack if it is infeasible for a forger to produce a valid message-signature pair after obtaining polynomially many signatures on a message of its choice from the signer.*
Formally, for every probabilistic polynomial forger $\mathcal{A}$ such that:

$$\mathsf{Adv}(\mathcal{A}) = \Pr\left[\begin{array}{l} (impk, imsk) \leftarrow \mathsf{IM\_KeyGen}(1^k); \\ C_i \leftarrow \mathsf{IM\_Sign}(imsk, r_i); \\ for\ i = 1, 2, ..., k; \quad r_i \in_R \mathbb{Z}_q; \\ m_i \leftarrow \mathcal{A}(impk, m_1, C_1, \sigma_1, ..., m_k, C_k, \sigma_k); \\ \sigma_i \leftarrow \mathsf{IM\_Sign}(S_i, m); \\ (m, C, \sigma) \leftarrow \mathcal{A}(impk, m_1, C_1, \sigma_1, ..., m_k, C_k, \sigma_k); \\ m \neq m_1, ..., m_k \text{ and } \mathsf{IM\_Verify}(impk, m, C, \sigma) = \text{accept}; \end{array}\right] \leq \epsilon$$

5.3 Generic Construction of IBMS from IOS

We observe the relevancy between online/offline signature and multisignature:

1. Both of them consists of four algorithms.

2. The offline signing phase is independent of messages and the resulting signature is verifiable against a public key ID. Thus the offline signature can be counted as a *session key* or *session commitment*, and the offline signing phase can be combined with the IBMS's key generation phase.

3. The online signing phase can be used as the stand alone signing phase of IBMS, plus necessary aggregations.

4. The IBMS signature verification requires the message, public keys (including the identity and the session keys), and the signature (online signature).

5.3.1 The Scheme

We provide the generic construction of the multisignature scheme based on identity based online/offline signature scheme.

IM_ParamGen (1_k)
  $(IOSK^*,\ params) \leftarrow$ IO_ParamGen(1^k)
  $IMSK^* \leftarrow IOSK^*$
  return $(IMSK^*,\ params)$

IM_KeyGen $(G_{Sub},\ id,\ IMP_{pub})$
  $(iosk_{id},\ iopk_{id}) \leftarrow$ IO_Ext(id, IMSK*, params)
  $imsk_{id} \leftarrow iosk_{id}$
  $impk_{id} \leftarrow iopk_{id}$ $C_{id} \leftarrow$ IO_OffSign(id, $\mathsf{iosk_{id}}$, params)
  return $(impk_{id},\ imsk_{id},\ C_{id})$

IM_Sign $(m,\ G_{Sub},\ imsk_{id},\ C_{id})$
  $\sigma_{id} \leftarrow$ IO_OnSign(m, id, S, params, $\mathsf{imsk_{id}}$)
  $\widetilde{\sigma} \leftarrow \Sigma^{G_{Sub}}(\sigma_{id})$
  $\widetilde{C} \leftarrow \Sigma^{G_{Sub}}(C_{id})$
  return $(\widetilde{\sigma},\ \widetilde{C})$

IM_Verify $(m,\ G_{Sub},\ \widetilde{\sigma},\ \widetilde{C})$
  $b \leftarrow$ IO_Verify(m, $\mathsf{G_{Sub}}$, params, $\widetilde{\sigma}$, $\widetilde{\mathsf{C}}$)
  return b

Figure 5.1: Generic Construction of IBMS

5.3.2 Security Arguments

Theorem 5 *The ID-based multisignature scheme is secure only if the corresponding ID-based online/offline signature scheme is existentially unforgeable against chosen-message attacks.*

Proof Suppose there is a polynomial time adversary $\mathcal{A}^{EF-IOS-CMA}$ who breaks the ID-based online/offline signature scheme. The ID-based multisignature can be broken by running the same queries performed by $\mathcal{A}^{EF-IOS-CMA}$ with the help of same forger $\mathcal{F}^{EF-IOS-CMA}$.

1. The challenger $\mathcal{A}^{EF-IAMS-CMA}$ runs the IO_ParamGen algorithm to generate the system parameters and sends them to the forger $\mathcal{F}^{EF-IOS-CMA}$.

2. The adversary $\mathcal{F}^{EF-IOS-CMA}$ performs the following queries:

   - **Key Generation Query $\mathbf{O}^{\mathbf{IM}}_{\mathbf{KGN}}$:** $\mathcal{F}^{EF-IOS-CMA}$ provides an identity ID of the uncorrupted player in G_{Sub} to **Key Extraction Query $\mathbf{O}^{\mathbf{IOS}}_{\mathbf{Ext}}$** and **Offline Signing Query $\mathbf{O}^{\mathbf{IOS}}_{\mathbf{OffSign}}$** of ID-based on signature. It receives the corresponding secret key D_{ID}, and an offline signature as its temporary public key for current signing session.
   - **Signing Query $\mathbf{O}^{\mathbf{IM}}_{\mathbf{Sign}}$:** $\mathcal{F}^{EF-IOS-CMA}$ produces a message m, a signature generated by **Online Singing Query $\mathbf{O}^{\mathbf{IOS}}_{\mathbf{OnSign}}$** of ID-based online/offline signature.

3. After a polynomial number of queries, $\mathcal{F}^{EF-IOS-CMA}$ produces a tuple (m^*, σ^*, S) of identity ID^*, whose secret key was never asked in key extraction query and the pair (ID^*, m^*) was never asked in online/offline signing queries. Obviously this resulting tuple satisfies the following:

   - σ^* is a valid signature on the message m by the subgroup G_{Sub} of players.
   - there exists an uncorrupted player $P^* \in G_{Sub}$ who has never been asked by $\mathcal{F}^{EF-IOS-CMA}$ to execute the **Signing Query $\mathbf{O}^{\mathbf{IOS}}_{\mathbf{OnSign}}$** on m and G_{Sub}.

□

5.4 A Concrete Construction

We adapt our online/offline signature scheme to the multisignature scheme according to the generic construction. The resulting scheme consists of four algorithms: system

setup, key generation, signing and verification.

Setup. Given $\mathbb{G}_1$ and its generator P, pick a random $s \in \mathbb{Z}_q^*$, and set $P_{pub} = sP$. Choose a cryptographic hash function $H_0 : \{0,1\}^* \to \mathbb{G}_1$ and $H_1 : \{0,1\}^* \times \mathbb{G}_1 \to \mathbb{Z}_q^*$. The system parameters are (P, P_{pub}, H_0, H_1). The master key is s. H_0 and H_1 behave as random oracles

KeyGen. Each player P_i $(1 \leq i \leq l)$ in G does the following:

1. on input identity ID_i, computes the secret key as $D_i = sH_0(ID_i)$
2. randomly choose $x_i, r_i \in \mathbb{Z}_q^*$
3. compute the signing commitment for the current session as $C_i = D_i - xP_{pub}$, $R_i = r_iP$, and $U_i = x_iP$
4. broadcast (C_i, R_i, U_i) to all the players.

Sign. Suppose the players in a subgroup $S = P_{i_1}, ..., P_{i_l}$ wish to jointly sign a message m. Upon receiving (C_i, R_i) from all the other players, each of them does the following:

1. verifies the received public key by checking the equality of the equation:
$$e(C_i, P) = e(Q_i - U_i, P_{pub})$$
2. if the equality holds, compute $\widetilde{C} = \sum_{j=1}^{l} C_j = \sum_{i=1}^{l} D_j - \sum_{j=1}^{l} x_j P_{pub}$
3. computes $\widetilde{R} = \sum_{j=1}^{l} R_j = \sum_{j=1}^{l} r_j P$
4. computes the signature as
   (a) each signer computes the signature $\sigma_j = H_1(m, \widetilde{R})r_j + x_j$ and broadcasts to all the signer P_{i_j} $(1 \leq j \leq l)$
   (b) upon receiving all the σ_j, each signer computes
$$\begin{aligned} \widetilde{\sigma} &= \sum_{j=1}^{l} \sigma_j \\ &= H_1(m, \widetilde{R}) \sum_{j=1}^{l} r_j + \sum_{j=1}^{l} x_j \end{aligned}$$

   The resulting multisignature for message m is $(\widetilde{\sigma},\ \widetilde{C},\ \widetilde{R})$. To further reduce the signature size, we combine $\widetilde{\sigma}$ and $\widetilde{C}$ to obtain a new parameter $\widetilde{V}$ by
$$\widetilde{V} = \widetilde{C} + \widetilde{\sigma} P_{pub}$$

The final signature is a tuple $(\widetilde{V},\ \widetilde{R})$.

Verify. The multisignature can be verified by all the group members who possess the pair $(\widetilde{C}, \widetilde{R})$. Given signature $\widetilde{\sigma}$, commitment $(\widetilde{pk}, \widetilde{R})$ and message m, check wether $(P_{pub}, \widetilde{V}, P, \sum_{j=1}^{l} Q_j + H_1(m, \widetilde{R})\widetilde{R})$ is a valid Diffie-Hellman tuple.

The equation holds since

$$\begin{aligned} e(\widetilde{V},\ P) &= e(\widetilde{C} + \widetilde{\sigma}P_{pub},\ P) \\ &= e(\sum_{j=1}^{l} D_j - \sum_{j=1}^{l} x_j P_{pub} + (H_1(m, \widetilde{R}) \sum_{j=1}^{l} r_j + \sum_{j=1}^{l} x_j)P_{pub},\ P) \\ &= e(\sum_{j=1}^{l} D_j + H_1(m, R) \sum_{j=1}^{l} r_j P_{pub},\ P) \\ &= e(s(\sum_{j=1}^{l} Q_j + H_1(m, \widetilde{R}) \sum_{j=1}^{l} r_j P),\ P) \\ &= e(\sum_{j=1}^{l} Q_j + H_1(m, \widetilde{R})\widetilde{R},\ P_{pub}) \end{aligned}$$

5.4.1 Security Analysis

We still start by assuming the existence of a forger $\mathcal{F}$ and an attacker $\mathcal{A}$, and initialise the system public key as $P_{pub} = aP$. Since the target subgroup we are supposed to attack contains one uncorrupted signer ID_* (we obtain the secret keys of all the other corrupted signers), $\mathcal{A}$ only needs to simulate $P_{uncorrupted}$ during key generation and signing. A big difference to the previous proof is that instead of letting $\mathcal{A}$ flip a coin to decide whether the corresponding identity is to be attacked or not, we calculate the probability of getting a fixed identity by using Cha-Cheon's ID attack [16]. This probability can be used to replace δ.

Theorem 6 *In the random oracle model, if a probabilistic polynomial time forger $\mathcal{F}$ has an advantage ε in forging an IBMS with running time t and asking H_0,H_1,key extraction oracle and signing oracle q_{H_0}, q_{H_1}, q_e and q_s times respectively, then the CDH problem can be solve with an advantage*

$$\varepsilon' > ((1 - \frac{1}{q})\frac{1}{q_{H_0}})(\varepsilon - \frac{q_S(q_{H_1} + q_S) + 1}{2^k})$$

with running time $t' < t + (q_{H_0} + 4q_e)t_m$, where t_m is the time to compute a scalar multiplication in $\mathbb{G}_1$.

Proof In running our simulation, we construct the similar environment as in IOS described in section 4.4.2. $\mathcal{A}$ answers the queries as followed.

ID hash query: when an identity ID is submitted to H_0 oracle, $\mathcal{A}$ flips a coin $T \in 0, 1$ which yields 1 with probability δ and 0 with probability $1 - \delta$. $\mathcal{A}$ then randomly chooses $u_i \in \mathbb{Z}_q^*$. If $T = 0$, $\mathcal{A}$ sets the value Q_i as $u_i P$. Otherwise, Q_i is set as ubP. $\mathcal{A}$ records the tuple (ID_i, u_i, T_i) in a list L_0, and returns Q_i as the answer.

Key extraction query: when $\mathcal{A}$ receives a key extraction query, it firstly checks whether the corresponding tuple (ID_i, u_i, T_i) exists in L_0. If it does not exist, $\mathcal{A}$ outputs "failure" and halts. If it exists $\mathcal{A}$ further checks the value of T_i. If $T_i = 0$, it computes the secret key as $uP_{pub} = uaP$ and returns it to $\mathcal{F}$. Otherwise, it outputs "failure" and halts.

Message hash query: $\mathcal{A}$ maintains a list L_1 for message hash queries. When a tuple (Q_i, m_i) is submitted to H_1 oracle, $\mathcal{A}$ firstly checks the existence of the tuple in L_1. If it exists, the value will be returned. Otherwise, $\mathcal{A}$ randomly chooses $v_i \in \mathbb{Z}_q^*$, stores the tuple (Q_i, m_i, v_i), and returns v_i to $\mathcal{F}$ as the answer.

Offline signing query: $\mathcal{A}$ randomly chooses $\alpha_i, t_i, \beta_i \in \mathbb{Z}_q^*$ and defines offline signature as $S = (t_i - \alpha_i)P_{pub} = (t_i - \alpha_i)aP,\ R_i = \beta_i P$. The pair (S_i, R_i) and α_i are stored for future use.

Online signing query: when $\mathcal{A}$ receives an online signing query on message M_i for an identity ID_i, it firstly retrieves the corresponding u_i from L_0. The previously computed offline signature tuple (S_i, R_i) and value α_i is also retrieved. Then it defines the message hash value $H_1(m_i, R_i) = \beta_i^{-1}(t_i - u_i)$, and the online signature as $\sigma_i = \alpha_i$. If the message hash value has been defined before, $\mathcal{A}$ output "failure" and halts. Otherwise, the signature tuple (S_i, σ_i, R_i) is returned to $\mathcal{F}$.

The correctness of the answers produced by the signing query is shown in section 4.4.2.

Eventually, the forger $\mathcal{F}$ produces a valid signature tuple (S^*, σ^*, R^*) for message M^* of the identity ID^* and gives it to $\mathcal{A}$. $\mathcal{A}$ firstly recovers the tuple (ID^*, u^*, T^*) in list L_0 to check the value of T. if $T = 0$, $\mathcal{A}$ outputs "failure" and halts. Otherwise, the entry of (Q^*, m^*, v_i) must be in the list L_1 with overwhelming probability. If this entry does not exist, $\mathcal{A}$ outputs "failure" and halts. As the resulting signature tuple is valid, the following equation holds:

$$e(S^* + \sigma^* P_{pub}, P) \quad = \quad e(Q_{ID}^* + H_1(m^*)R^*, P_{pub}) \tag{5.1}$$

Besides we have $H_1(m^*) = v_iP$, $P_{pub} = ap$, and $Q_i = u_ibP$. According to (5.1) we can get:

$$\begin{aligned} e(S^* + \sigma^* aP, P) &= e(ubP + v_iR, aP) \\ e(S^* + \sigma^* aP, P) &= e(ubP, ap)e(v_iR, aP) \\ e(S^* + \sigma^* aP - v_iR, P) &= e(ubP, aP) \end{aligned}$$

The solution to the CDH instance (aP, bP) is $u^{*-1}(S^* + \sigma^* aP - v_iR)$.

To calculate the probability, we still consider three parts:

1. $\mathcal{A}$'s probability of failure caused by a conflict over H_1 is at most $q_S(q_{H_1} + q_S)/q$.
2. the probability of producing a valid forgery without asking $H_1(m^*, R^*)$ is $1/2^k$
3. the probability of $\mathcal{A}$ succeeds in a key extraction query is $\delta(1-\delta)^{q_e}$

In this case, δ involves two parts:

- the probability of producing a valid message-signature tuple (ID_*, m, σ^*) without any query of H_0 is at least $1 - \frac{1}{q}$
- the probability thatID^* is chosen randomly from the space of H_0 is at least $\frac{1}{q_{H_0}}$

The resulting probability that a target ID appears in our simulation is $\delta \geq (1 - \frac{1}{q})\frac{1}{q_{H_0}}$. Thus in (3) we have

$$1 - \delta = 1 - \frac{1}{q_{H_0}} + \frac{1}{q \cdot q_{H_0}} > 1 - \frac{1}{q_{H_0}}$$

$$\begin{aligned} \delta(1-\delta)^{q_e} &= ((1 - \frac{1}{q})\frac{1}{q_{H_0}})(1 - \frac{1}{q_{H_0}})^{q_e} \\ &> (1 - \frac{1}{q})\frac{1}{q_{H_0}} \end{aligned}$$

Eventually it comes that $\mathcal{A}$'s advantages is at most

$$((1 - \frac{1}{q})\frac{1}{q_{H_0}})(\varepsilon - \frac{q_S(q_{H_1} + q_S) + 1}{2^k})$$

□

5.4.2 Efficiency Comparison

To compare the efficiency, we assume the safe length of GDH group $\mathbb{G}_1$ is ρ and the order of multiplicative group is q. We analyse the efficiency of signature schemes in relation to four indicators: signature size, pre-computation cost (*Pre-comp. Cost*), signature generation cost (*SG Cost*), verification cost (*V Cost*) and the type of problems (*Problem*). We define the pre-computation phase to include all the operations taken irrelevant to the message to be signed. Whereas the signing phase only contains the operation aimed at the message. Therein, the signing cost and pre-computation cost are justified in terms of how many elliptic curve scalar multiplications (ESM), or exponentiations are used. The verification cost is justified by counting how many pairings are used. We also assume the multisignature is generated by n signers.

We choose from the literature five ID-based multisignature schemes to perform this comparison, in which three of them use bilinear pairing and the other two are based on RSA. Besides our scheme (**IBM**, based on IOS), another four schemes include: **SOK-IBMS** [87] produced by Sakai *et al.*, **CZK-IBMS** [19], the ID-based blind multisignature scheme produced by Chen *et al.*, based on Cha-Cheon scheme [16], **WH-IBMS** [99] produced by Wu and Hsu, **CLL-IBMS** [17] produced by Chang *et al.*

We firstly look at the comparison between single ID-based signature schemes in 5.1. It is obvious that our online/offline signature scheme is efficient in online signing since no ESM needs to be performed. Two RSA based signature schemes are efficient in signing and verification, but signature sizes are apparently larger than others.

	Signature Size	Pre-comp Cost	SG Cost	V Cost	Problem
SOK-IBS	$2\log\rho$	1 ESM	1 ESM	2 pairings	CDHP
Cha-Cheon	$2\log\rho$	1 ESM	1 ESM	2 pairings	CDHP
IOS	$2\log\rho + \log q$	2 ESM	0 ESM	2 pairings	CDHP
WH-IBS	$\log N$	N/A	1 expon.	2 expon.	RSA
CLL-IBS	$\log N$	N/A	1 expon.	3 expon.	RSA

Table 5.1: ID-based Signatures Efficiency Comparison

The comparison between ID-based multisignature schemes are listed in Table 5.2. We can see that the Chen *et al.*'s scheme is very efficient in average, requiring $2n$ scalar multiplications in the pre-computation phase and the signing phase. Our scheme preforms the same number of scalar multiplications ($2n$) in pre-computation phase. However, the actual signing phase needs only 1 scalar multiplication. We can draw the conclusion that our ID-based multisignature scheme preserves the advantage of

its base scheme, ID-based online/offline signature scheme, which is able to shift the computational overhead to the pre-computation phase, therefore enables quick signing when messages arrive.

	Signature Size	Pre-comp Cost	SG Cost	V Cost	Problem
SOK-IBMS	$(n+1)\log\rho$	n ESM	$\Sigma_{i=1}^{n} i$ ESM	3 pairings	CDHP
CZK-IBMS	$2\log\rho$	n ESM	n ESM	2 pairings	CDHP
IBMS	$2\log\rho$	$2n$ ESM	1 ESM	2 pairings	CDHP
WH-IBMS	$\log q$	N/A	n expon.	$(n+1)$ expon.	RSA
CLL-IBMS	$\log q$	N/A	$2n$ expon.	3 expon.	RSA

Table 5.2: ID-based Multisignatures Efficiency Comparison

5.5 Summary

In the cryptographic domain, the construction of a signature scheme is always based on an application environment. A researcher firstly identifies the specific security requirements, then certain signature is constructed against each requirement. In general, signatures for one application are not adaptable to others. However, we observe that the adaptability of signature schemes is preferable in some cases where standard operation procedures are not available. The adaptability of signature schemes will then be able to deal with the variability of the application domain, and in turn provide generic security for different procedures with the help of a single signature scheme.

In this chapter, we described a transformation from ID-based online/offline signature schemes to ID-based multisignature schemes. We presented the generic transformation and proved it is secure against existential forgery under adaptive chosen message and ID attacks in the random oracle model assuming that the CDHP problem is hard with an assumption that there exists no polynomial time algorithms for any attacker to solve the CDHP. We compared our scheme with other ID-based multisignature schemes and concluded the transformation could inherit the quick signing capability from the online/offline signature scheme.

Chapter 6

Authentication Schemes for MANET Routing Operations

AODV is one of the most popular routing protocols. It has been extensively studied in order to provide efficient security adds-on to the original protocol. One of the famous constructions, SAODV [105], uses normal signatures to secure packets, which is a straightforward approach without regard to the protocol structure. In this section, we focus on offering more efficient authentication schemes to AODV by using specific signatures.

6.1 AODV Security Considerations

AODV is a simple and efficient on-demand ad hoc routing protocol. It is vulnerable to the following attacks.

Threats using modification AODV uses the destination sequence number to guarantee loop free routing. The destination sequence number is a monotonically increasing number representing the freshness of a routing request. A malicious node can decrease the destination sequence number on *RREP*. The destination node receiving this *RREQ* will compare the latest received destination sequence number to the new one. If the new one has a smaller value, this *RREQ* will be discarded. By making *RREQ*s towards certain destination to be discarded, a denial-of-service attack is launched. Besides, attackers can also modify the hop count field to advocate the "shortest route". Since AODV always chooses routes having the least hop count value, the attacker can easily accept any route that passes through it.

Threats using fabrication In AODV, a malicious node can interrupt the communication between any two nodes by flooding fake *RRER* messages along the path. Suppose there is a path $A \rightarrow B \rightarrow C \rightarrow D \rightarrow X$, if a malicious node M wants

to interrupt the communication between A and X, he can pretend to be a node in this route, for example C and broadcast a *RERR* to B, claiming the route to X is broken. Then B will delete the entry for X and forward this *RRER* to A. Hence, A will believe the route to X is broken and delete the entry as well. If M is powerful enough to broadcast *RRER* throughout all the routes towards X, X will be isolated in the network.

Threats Using Impersonation In AODV route discovery, a node A establishes a route to another node B by sending a *RREQ* message towards it. Node B is supposed to reply the *RREQ* with a *RREP*. However, any node who receives the *RREQ* is able to reply this *RREQ*. A malicious node can pretend to be the node B and reply a *RREP*, in order to redirect packets addressed to B to itself. In the absence of any higher level authenticating information, a malicious node can mislead A into believing that it is communicating with B. Thus, even though A will finally receive multiple *RREP*s, the fake *RREP* having the shortest hop count will be accepted anyway.

SAODV is not secure against modification attack because the non-existence of public key infrastructure. The originator signs its own public key along with the message to prove that it holds a matching secret key. It is very easy for a malicious node to exploit this vulnerability in order to modify the routing packet. For example, assume an *RREQ* is supposed to be transmitted along the path $O \rightarrow A \rightarrow M \rightarrow B \rightarrow T$, where M is a malicious node trying to modify this *RREQ*. As shown in 6.1, in the beginning, O computes the signature S_O on the *RREQ* and its public key PK_O according to SAODV. This signature is sent to A who processes the *RREQ* normally. However, when M receives the *RREQ*, it will do the following steps:

1. drop the received signature S_O and public key PK_O;
2. generate a new pair of keys SK'_O, PK'_O;
3. modify the *RREQ* as it wishes, and replaces the *public key field* with the new public key generated;
4. generate a new signature over the modified *RREQ* and the new public key;
5. broadcast the new *RREQ*.

The next hop node or even the destination will not be able to detect the modification because this fake signature is verifiable by the public key.

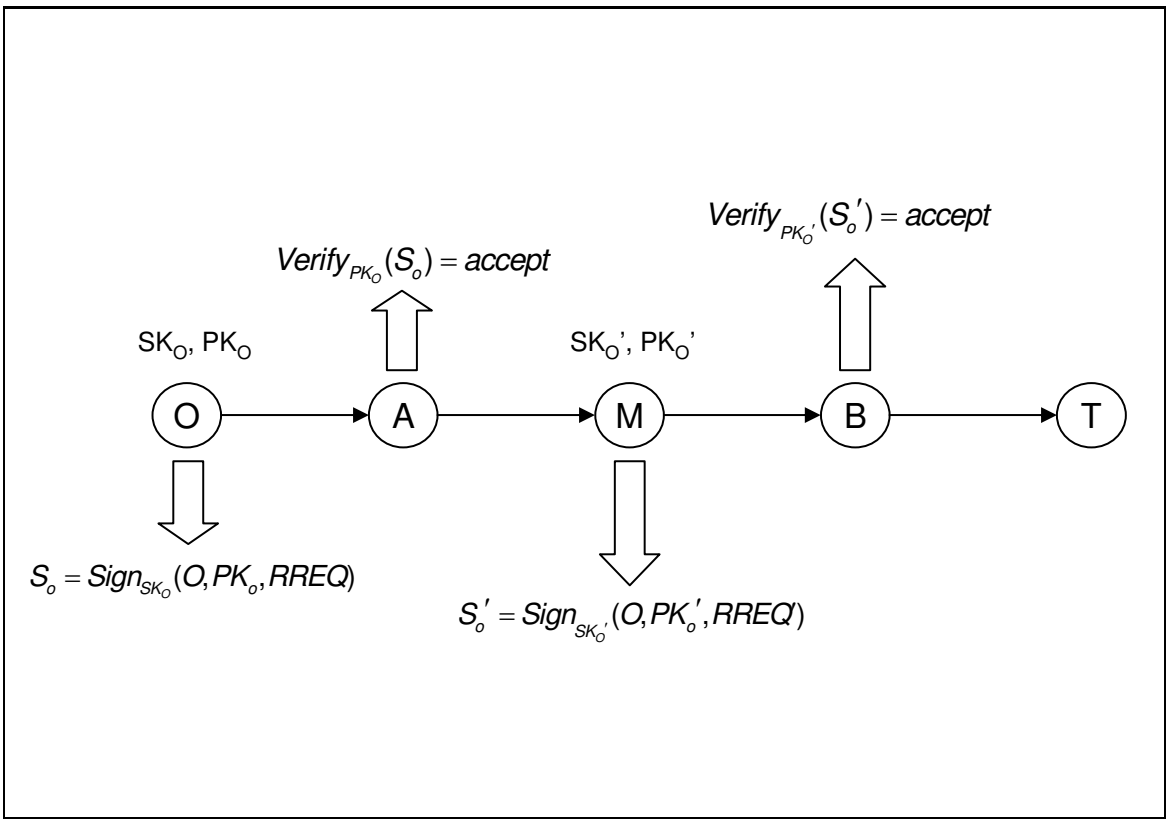

Figure 6.1: A Possible Attack in SAODV

Hence we argue the necessity of the existence of an *offline trusted third party*, such as PKI or KGC, in order to provide full scale security. We also argue that the *hop-by-hop authentication* in AODV is needed, because with the help of strong authentication, we will be able to prevent the impersonation attacks which, as mentioned before, is the fundamental step of launching other more complicated attacks. However, to detect attacks such as tunneling, this requires that a node must be aware of all its neighbours in a timely manner. This feature is usually enabled using active or passive acknowledgement (ACK). In AODV, we can make use of the periodically broadcasted *Hello* messages. A node will consistently broadcast *Hello* messages as long as it is part of an active route. Every node can keep a list of all its active neighbours in the network and update the list according to the *Hello* message. By this means, nodes will be aware of their surrounding environment.

6.2 Authentication Schemes for AODV

To achieve this goal, we borrow the ARAN authentication scheme described in section 2.4.1. We apply our ID-based online/offline signature schemes and Reyzin *et al.*'s HORS one-time signature scheme to construct our authentication schemes [100, 101]. We also make use of ARAN authentication scheme to achieve a higher level of authentication.

6.2.1 A Scheme based on ID-based Online/offline Signatures

We assume the existence of an offline key generation centre (KGC) to enable our ID-based constructions. Every node, before entering the network, must submit its identity to KGC which runs the Setup algorithm. The KGC will generate a public-secret key pair according to the node's identity information. The key pair, along with necessary system parameters, are sent to the node through a secure channel. In reality, the company can play the role of a KGC because most companies provide laptops or PDAs to their employees. A more cost effective approach is to distribute a smart card containing all the necessary information to each employee. Thus each time an employee wants to enter the network, he only needs to plug in the smart card so that his device will become an authentic node in the network.

After entering the network, nodes start with computing offline signatures for communications using the OffSign algorithm. Since the offline signature is created over a random value, a node can randomly choose several values and compute signatures respectively. The signatures and other necessary values are stored for future online signature generation.

When a node O is about to transmit data to another node T, to whom node O does not have an active route, it generates a $RREQ$ according to AODV protocol. Then this $RREQ$ is treated as the message input to online signature generation algorithm OnSign. This phase is very efficient since signature generation only requires one hash for both of our schemes. Then the sender node broadcasts $RREQ$ to neighbours along with its signature S^O_{IOS}.

Upon the second hop node A receives this $RREQ$, it will then verify O's signature using O's identity as the public key. It is desirable to use nodes' IP addresses as public keys, because IP addresses can be easily obtained from routing packets' IP headers. If the signature is authentic, A will process the $RREQ$ according to AODV. It then retrieves its offline signature to generates its online signature over the $RREQ$ and

appends it to the received message. The new message $(RREQ, S^O_{IOS}, S^A_{IOS})$ is finally broadcast again. The scheme is shown in Fig. 6.2

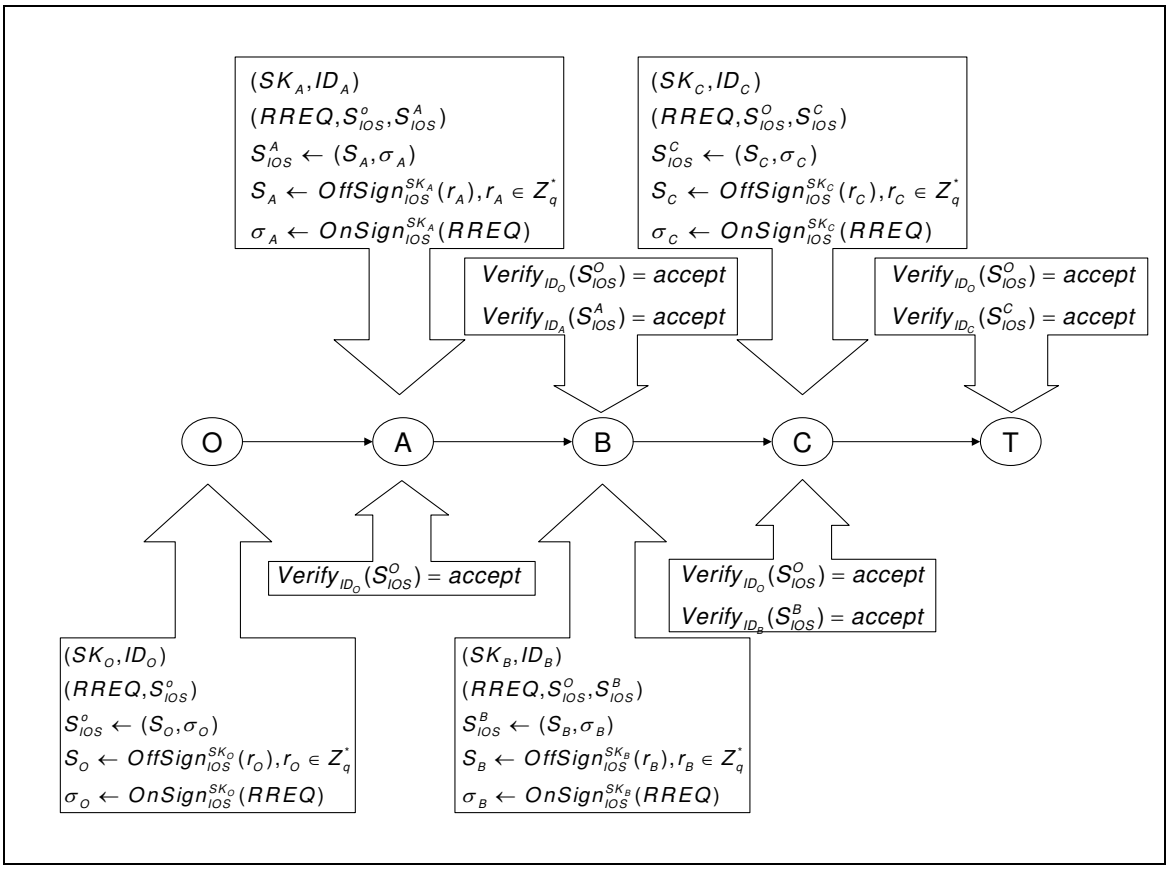

Figure 6.2: IOS based Authentication Scheme for AODV Route Request Processing

Notice that the S^A_{IOS} is supposed to be generated over the received $RREQ$ because all the fields of $RREQ$ except the *hop count* field, is immutable during a route discovery, whereas the *hop count* field is zeroed when generating signature.

At the third hop node B, it verifies both O's signature and A's signature using their identities. If both verifications are passed, B acts the same as A to process $RREQ$ and generate its signature S^B_{IOS}. It then replaces S^A_{IOS} with S^B_{IOS} and broadcast the new message $(RREQ, S^O_{IOS}, S^B_{IOS})$. The same procedures will be performed by future receiving nodes until $RREQ$ reaches the target T.

The target T, upon the receipt of $RREQ$, will respond with a $RREP$. It then acts the same as O to produce its signature over $RREP$. The resulting message $(RREP, S^T_{IOS})$ is to be multicasted or unicasted back to the originator O along the route found. The other nodes along the route perform the $RREP$ as they did to $RREQ$. The route error

message *RERR* and the *Hello* message can be performed as in SAODV.

Remarks

Signature verification can be delayed to achieve better efficiency, i.e. the receiving node broadcasts the *RREQ* or *RREP* right after it increases the value of the *hop count* field. Signature verification can be performed when the node is idle. By this means, the routing operations can be accelerated since no delay is presented. However, a node will not update its routing table entries using any unauthenticated routing packets. Only if the received *RREQ* or *RREP* pass the verification, will the routing table entries be updated according to the information carried in the packet.

This ID-based online/offline scheme has two major advantages:

1. It is efficient in signature generation. Our scheme takes one hash in online signing. If the verification is delayed at each hop, the propagation of routing packets bears almost no delay.

2. It solves the public key distribution problem. Public keys can be extracted from nodes' identity information using a public known algorithm obtained from KGC. The bandwidth used for transmitting public key certificates is saved.

On the other hand, the use of ARAN's hop-by-hop authentication scheme requires extra bandwidth in transmitting one more signature during the route discovery. We notice that without applying ARAN's scheme, our scheme still provides basic security features including sender authentication, message integrity and non-repudiation, because the nature of AODV is that the originator and the target are unaware of the route in between. However, hop-by-hop authentication scheme provides even more complete authentication within the neighbourhood, which largely prevents the impersonation attack.

6.2.2 A Scheme based on One-time Signatures

The first one-time signature based routing authentication scheme was proposed by Zhang [108] in 1998. The application domain of his COSP protocol, standing for *chained one-time signature protocol*, is the fixed network. COSP makes use of hash chains to generate one time key components and the resulting signatures are of different sizes. In this section, we provide another authentication scheme which is efficient in signature verification. In order to construct our scheme [102], we combine COSP's key

chaining scheme with HORS one-time signature [82] as shown in Fig. 6.3. We also fit our scheme into the ARAN authentication scheme, so that our scheme will provide sender authentication and hop-by-hop authentication.

6.2.3 Key Chain Construction

In order to construct a key chain of length n to sign k messages, we do the following:

1. chooses n secret key components x_j where $1 \leq j \leq n$) at random.
2. creates n hash chains each of length k as in Fig. 6.4.

Notice that key chain is generated from bottom to top, whereas its usage is from top to bottom. To sign a message at each time, one row of the key chain will be used.

6.2.4 The Scheme

Our scheme makes use of both our ID-based online/offline signature and HORS one-time signature. ID-based online/offline signature is used to provide sender authentication, whereas HORS one-time signature is to enable the ARAN's hop-by-hop authentication. Firstly, we still assume the existence of an offline KGC. Each node submits its identity to KGC before entering the network, in order to obtain a public-secret key pair and necessary system parameters.

Once entering the network, nodes start to produce offline signatures as described in the previous section. In addition, each nodes N has to decide on the security parameters ℓ, k and n of the key chain to be generated. The decision is made in terms of the message length b. To build a key chain to sign b-bit messages, the key chain length n and the signature size t are supposed to satisfy

$$\left(\begin{array}{c} t \\ n \end{array} \right) \geq 2^b$$

where ℓ is the length of a one-way function's output.

Each node then uses the security parameter as input to run the key chain generation algorithm. In the key chain generated in Fig. **??**, the bottom line is the hash chain seed and the top line is the hash chain anchor. Nodes start to use the key chain from the top row, thus the first public key component is the first line $pk_O^1 = h^k(x_1), ..., h^k(x_n)$ and its matching secret key component is $sk_O^1 = h^{k-1}(x_1), ..., h^{k-1}(x_n)$. Because of the one-wayness of hash chains, revealing the public key component which is a later value does not enable the disclosure of the secret key component which is a earlier value.

Key Generation

Input: Parameters ℓ,t, n
Generate n random ℓ-bit strings $s_1, s_2, ..., s_n$
Let $v_i = f(s_i)$ for $1 \leq i \leq n$

Output: $PK = (t, v_1, v_2, ..., v_t)$ and $SK = (t, s_1, s_2, ..., s_n)$

Sign

Input: Message m and secret key $SK = (t, s_1, s_2, ..., s_n)$
Let $h = Hash(m)$
Split h into t substrings $h_1, h_2, ..., h_n$, of length $\log tbitseach$ interpret each h_j as an integer i_j for $1 \leq j \leq t$

Output: $\sigma = (s_{i_1}, s_{i_2}, ..., s_{i_t})$

Verify

Input: Message m, signature $\sigma = (s_{i_1}, s_{i_2}, ..., s_{i_t})$,and public key $PK = (t, v_1, v_2, ..., v_t)$
Let $h = Hash(m)$
Split h into t substrings $h_1, h_2, ..., h_n$, of length $\log tbitseach$ interpret each h_j as an integer i_j for $1 \leq j \leq t$

Output: "accept" if for each j, $1 \leq j \leq t$, $f(s'_j) = v_{i_j}$; "reject" otherwise

Figure 6.3: HORS One-time Signature Scheme

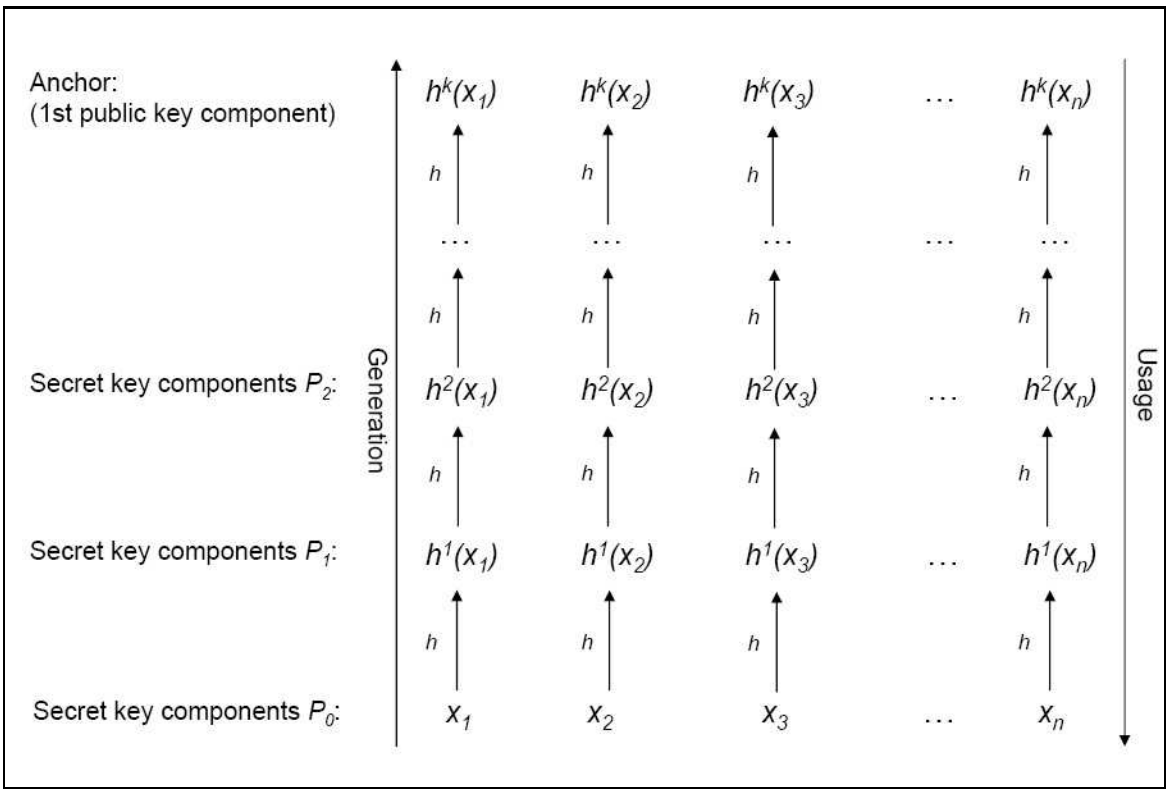

Figure 6.4: Key Chain Construction

The first public key component is broadcasted to neighbours. To enable the authentication of this value, we use our ID-based online/offline signature scheme to generate a signature over this value.

$$N \rightarrow Broadcast: \ Sign_{IOS}^{SK_N}(pk_N^1, counter)$$

where the *counter* indicate the life time of the public key component. Neighbours upon the receipt of N's public key component, they verify the signature using N's identity to see the public key component is authentic.

$$Verify_{IOS}^{ID_N}(Sign_{IOS}^{SK_N}(pk_N^1, counter)) = accept$$

If it passes the verification, the public key component will be stored during its lifetime.

Now nodes are ready to communication with each other. When a node O is about to send data to another node T to whom it does not have an active route, O generate a $RREQ$ and retrieve offline signature S_O to compute the online signature σ_O. The

RREQ along with the signature is broadcasted to the next hop.

$$
\begin{aligned}
O \rightarrow Broadcast: \ & S_{IOS}^{SK_O} \\
& S_{IOS}^{SK_O} \leftarrow (S_O, \sigma_O) \\
& S_O = OffSign_{IOS}^{SK_O}(r_O),\ r_O \in \mathbb{Z}_q^* \\
& \sigma_O = OnSign_{IOS}^{SK_O}(RREQ)
\end{aligned}
$$

The next hop node A verifies the signature of O. If it proves to be authentic, A generates a one-time signature according to HORS over the received message and rebroadcasts it.

$$
\begin{aligned}
A \rightarrow Broadcast: \ & S_{IOS}^{SK_O},\ S_{OT}^{A} \\
& S_{OT}^{A} \leftarrow (s_{i_1}, s_{i_2}, .., s_{i_k}) \\
& S_{OT}^{A} = Sign_{OT}^{sk_A^1}(RREQ)
\end{aligned}
$$

When other nodes along the path receive this double signed *RREQ*, each of them firstly retrieves the previous hop's public key component to verify the one-time signature. If the signature is fine, it verifies the originator using O's identity ID_O, which is extracted from *RREQ*. Only if both of the signatures are fine, does the second hop node update its routing table entry according to *RREQ*. Then the *RREQ* is processed the same way by all the intermediate nodes until it reaches the target, as shown in Fig.6.5.

When the *RREQ* reaches the target T, T performs verifications as in each intermediate node. Then a *RREP* is generated and signed the same as *RREQ*. Each intermediate node will transmit it back to the originator through the reverse route and same operations are performed along the route.

6.2.5 A Feature for Gratuitous Route Reply

In AODV, gratuitous route reply [73] enables an intermediate node to reply *RREQ*s which has an active route towards the destination. This feature is optional in AODV, though turning on this feature will highly enhance the efficiency of routing discovery. However, to enable this feature, an additional technique is needed. The conceptual idea is that since we used digital signature to protect each routing message at each hop,

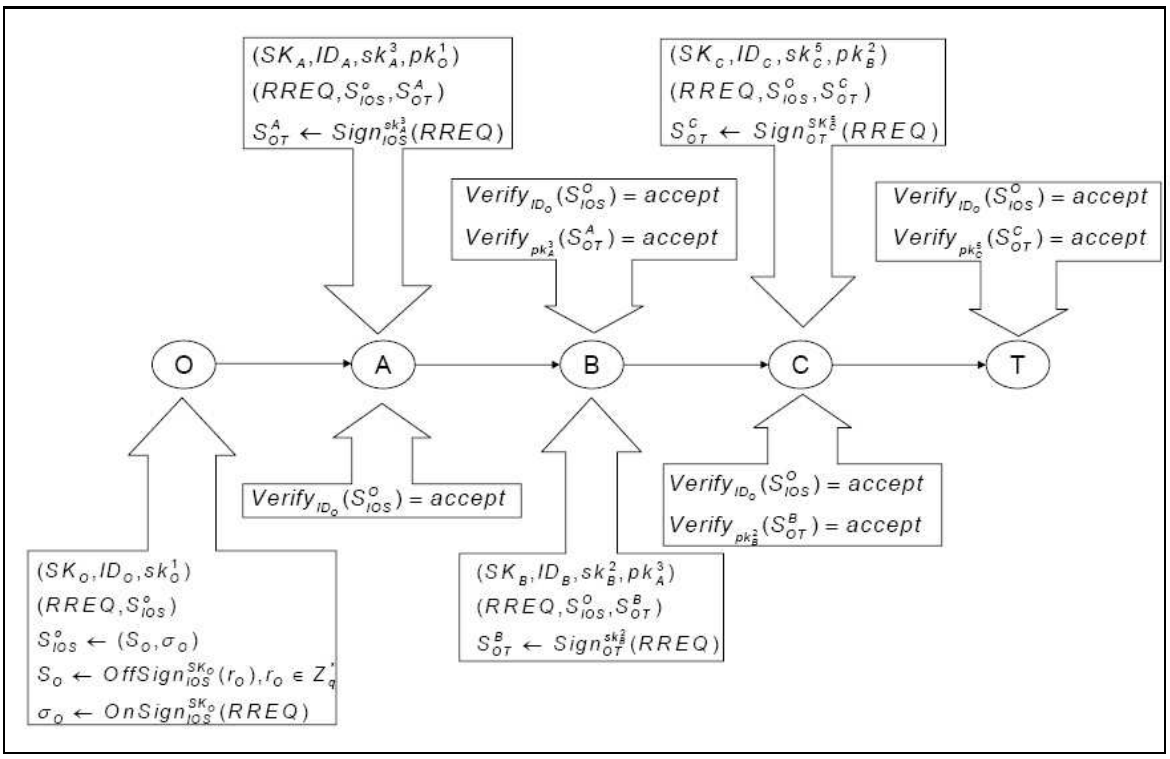

Figure 6.5: HORS based Authentication Scheme for AODV Route Request processing

for an intermediate node to reply *RREQ*s instead of the destination, the intermediate node should be able to sign the *RREQ* properly on behalf of the destination.

To solve this problem, we borrow the idea from proxy signature proposed by Varadharajan *et al.* [96], in which delegation is enabled by using a warrant. The warrant appears as a delegation token, containing the identities of primary signer and proxy signer, the privilege (Pr_N) given to proxy signer, an identifier (r_N) used by primary signer, and a timestamp (t_n). This delegation token is signed by the primary signer.

We simplify the above delegation token into three fields: the destination's identity, an identifier r_N and a timestamp t_N. It is possible because the token does not need to be designated to certain nodes. Any node that has received the token from a target is automatically proved to be having an active route towards the target. Otherwise, it would not be able to obtain this token. The token is signed by the creator using our IOS signature for our scheme.

$$N \rightarrow Broadcast: \ (N, Token_N)$$
$$Token_N = Sign_{IOS}^{N}(N, r_N, t_N)$$

We assume that each node is able to store the received token for future gratuitous route reply.

If the gratuitous route reply option is turned on, node O broadcasting $RREQ$s must create tokens for gratuitous route reply delegation. The whole message including the token will be signed again, using the same public key as in signing the token. Then, the originator broadcasts the RREQ as usual.

$$O \rightarrow Broadcast: \ (O, RREQ, S^O_{IOS}, Token_N)$$
$$S^O_{IOS} = Sign^O_{IOS}(RREQ)$$

Upon receiving the $RREQ$, node processes the authentication as normal. Then it checks the timestamp to see if the token has expired. If the token is valid, the nodes will store the token for future use. The originator firstly checks if this $RREP$ was created by destination or by intermediate node. If it is a gratuitous route reply, the originator checks the timestamp to determine if the route is still active. Then the token and the $RREP$ will be authenticated as described before.

The originator firstly checks if this $RREP$ was created by the target or an intermediate node. If it is a gratuitous route reply, the originator checks the timestamp to determine if the route is still active. Then the token and the $RREP$ will be authenticated as previously described.

Remarks

Nodes' one-time public key components are supposed to be distributed locally among the neighbours. However, when the lifetime of a public key component is over, new public key components must be distributed. For example, for node N, if the previous public key component is $pk^i_O = h^i(x_1), ..., h^i(x_n)$, the new public key component will be $pk^{i-1}_O = h^{i-1}(x_1), ..., h^{i-1}(x_n)$ and the new secret key component is $sk^{i-2}_O = h^{i-2}(x_1), ..., h^{i-2}(x_n)$. The authentication of the update is done through hashing the newly received row to compare with the old public key component. The authenticity is guaranteed because we use IOS to protect the very first public key component and use hash chain to protect the rest updates.

The mechanism that we suggest using in updating public key components is the *Hello* message. This message is broadcast periodically to neighbours if the node is a part of an active route. Using *Hello* messages to bear the new public key components can reduce the number of packets being transmitted.

The number of key components to be generated by a hash chain is infinite. When a key chain is used out, a new key chain must be generated. Creating a large key chain requires huge memory and computations. Frequently re-initialising small key chains also brings on an enormous burden to nodes. The size of the key chain depends on the number of nodes in the network and nodes' routing performances, which is not discussed in this thesis.

Our scheme takes the advantages of ARAN's authentication scheme and HORS one-time signature, providing a strong authentication method secure against exploits using modification, fabrication and impersonation. One strong point of our scheme is that both signing and verification are very efficient in comparison with the previous scheme. The IOS signing requires one hash, which is the same as the previous scheme. The HORS one-time signature requires also only one hash in signature generation. Besides, its signature verification takes only k hashes. Yet, the tradeoff in this scheme is the large signature size and public key component size, plus the extra cost in computing and storing hash chains.

6.3 Authentication Scheme for DSR Using ID-based Multisignatures

In section we present an installation of authentication scheme for DSR protocol. We begin with identifying the security requirements needed in securing DSR protocol, and then we describe the detail scheme along with the algorithm.

6.3.1 DSR Security Considerations

The original DSR protocol does not provide any security features. It faces several attacks. We identify the most serious attacks in the followed.

Threats using modification In DSR, when transmitting packets, the initiator will append the previous learned source route on the data packets as a header, and transmit packets to the next hop. During this transmission, if an intermediate node alters the source route by removing some hops from the route, the target node will become unreachable. Even worse, the attacker can redirect the route by adding itself to the route. Therefore, all the data packets will go through this attacker.

Threats using fabrication In DSR, broadcasting fake routing messages can exploit the vulnerability of the promiscuous receiving mode. When a node is working in a promiscuous mode, it is able to overhear its neighbours' packets and add the routing information contained in the packets header into its own route cache. If a malicious broadcast fake packets with source route to some destination, its neighbours who overhear this transmission will add this route to its route cache. This kind of attack is named as *route cache poisoning*.

Threats Using Impersonation Impersonation attack means a malicious node pretends to be some other legitimate nodes in the network. It is usually the first step of launching other even serious attacks such as route cache poisoning, because the malicious node always wants to prevent revealing its identity during the attack. In DSR, a malicious node can take further advantage from impersonation by appending other nodes' IP address to DSR routing packets. This action will lead to fake routes to be created, and in the worst situation, this fake route will be spread throughout the network.

The above attacks are usually prevented using digital signatures. By introducing digital signatures to the DSR protocol, we can provide both data integrity (against unauthorised modification of routing packets) and the hop-by-hop authentication. However, although digital signatures do not prevent fabricating spoofed routing packets, it enables the tracing of malicious nodes, then secures the routing process in a proactive manner.

Normal digital signature schemes are not applicable to DSR situation because of the computation complexity: the number of signatures is increasing during the routing processing, and the verification time linearly increases. We further observe that in DSR routing packets described in 2.2.2, most fields are immutable during the whole route discovery. The only mutable fields are the packet length field and the hop count field, which are increasing linearly and usually protected by hash chains. Thus through the route discovery process, signatures only have to be created over immutable fields (mutable fields are zeroed during signature generation). Furthermore, all the signatures are to be created over the same message.

One signature scheme applicable to the DSR situation is called the multisignature scheme. Since most fields in a *RREQ* packet are immutable during the whole routing process, it is possible to create a multisignature over the immutable fields by each node. The option data length fields which is mutable can be protected by a hash chain as in AODV for protecting hop count field. The other mutable field, route record field, due

to containing only IP addresses, can be regarded as the public key of each node in an ID-based signature scheme.

6.3.2 Installation of IBMS over DSR

To enable the installation over DSR, we firstly define the total signers' group to include all the mobile nodes in MANET. The maximum size of the total group G should agree with the network capacity. We then define the subgroup S to include the mobile nodes involved in a routing operation. Therefore, each routing operation will accordingly form a subgroup whose maximum size equals the maximum hop count allowed by DSR protocol. Before a DSR based network is initialised, the total group is set as empty $G \leftarrow \phi$. Mobile nodes will be added to the total group G once they enter the network. Similarly, the subgroup S is initialised as empty ϕ as well, and mobile nodes will be added to the subgroup S when they are involved in some routing operations.

To perform the ID-based authentication, we assume the existence of an offline key generation centre (KGC). KGC runs the system Setup algorithm to generate all the parameters required. Each node, before entering the network, has to submit its credential to KGC. The KGC will run the key generation algorithm (KeyGen) to generate a public-secret key pair for each node. One straightforward method is to use a node's IP address as its public key and get the secret key generated over it. The parameters and keys will be transmitted to mobile nodes through a secure channel. Once a node has obtained all the necessary parameters, it can start to do all the pre-computations according to signing algorithm, in order to achieve the best efficiency in signing.

When a *RREQ* is issued, the initiator runs the signing algorithm Signing to generate a signature over all the immutable fields. The mutable fields, the *RREQ* data length field and the route address field are excluded and their values are set to 0 during the signature generation.

The *RREQ* along with the signature will be broadcasted to next hop neighbours. The next hop nodes will firstly run the verification algorithm Verifying to evaluate the signature validity. To run this algorithm, the verifier firstly needs to extract the IP addresses, which are also the public keys of previous hop nodes, from the *RREQ*. Accordingly, if a malicious node deliberately removes some IP addresses from the *RREQ*, the signature will not pass the verification and the route carried by the *RREQ* will be considered as incorrect and rejected. Therefore, by preforming the signature verification, both the signature and the route are authenticated.

If the signature is valid, the verifier (the next hop neighbour) will produce a new

signature over the immutable field of the original received message. This node then appends its own IP address to the *RREQ* and broadcasts the *RREQ* along with the signature. The neighbours of the third hop will perform the same operations that the neighburs of the second hop did to produce signatures over the original *RREQ* generated by the initiator. This process will continue until the *RREQ* reaches the target node. The target node, after verifying and accepting the *RREQ*, will respond with a *RREP*. This *RREP* will be transmitted back to the initiator along the route discovered. In this condition, the signature of the *RREP* will be processed the same as the *RREQ*. The signing process is shown in Fig. 6.6.

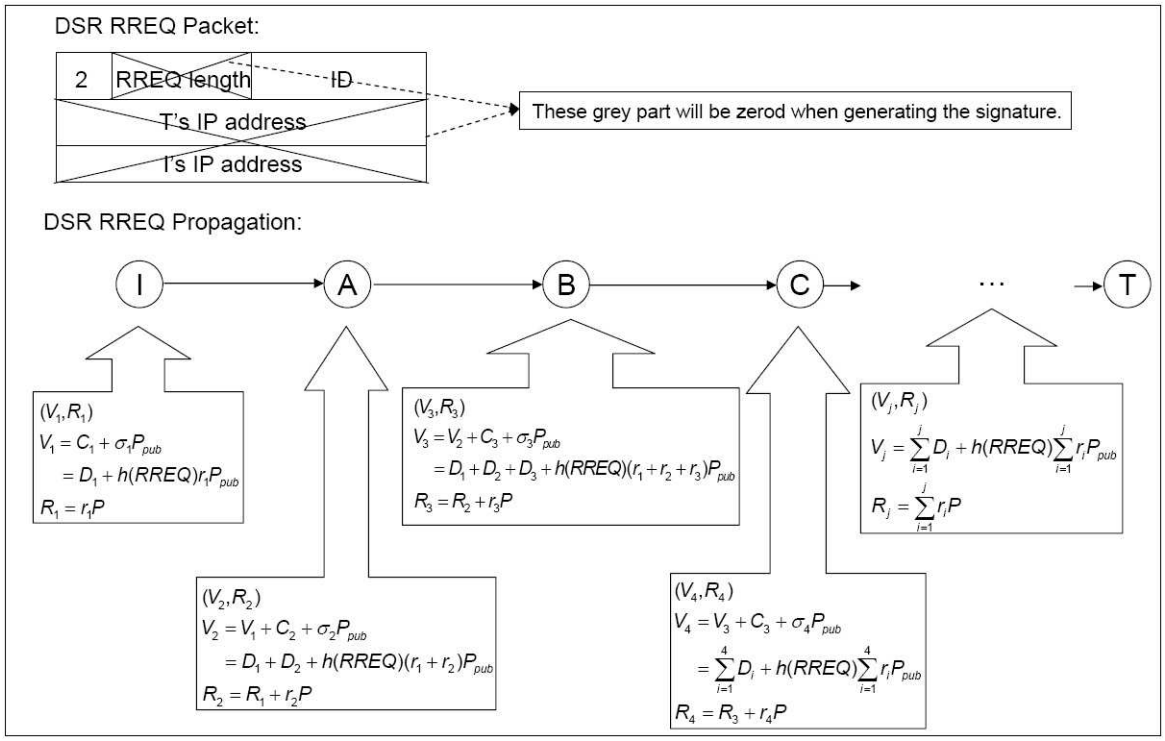

Figure 6.6: IASM signature generation process in DSR

Our signing algorithm is given in Fig. 6.7. In addition, to further improve efficiency, the verification process can be delayed. In this sense, when a node receives the *RREQ* along with the signature, it will generate a new signature before verifying the received one. However, this node will not update its route cache until the received signature is verified.

One arguable point of using multisignature in a sequential form is that the node is able to remove itself from the path. We argue that removing itself does not make any sense. To remove itself, a node passes the routing packet to its next hop neighbour

without changing anything. For example, the node M receives a route packet from node A and passes the packet to node B without adding its IP address to *route record* and increasing the value of data length field. There are two situations that could happen. Firstly, if node A is in the neighbourhood of node B and node M's behaviour actually results in a legal route which is one hop shorter. This route will be accepted by node B, or generated by node B sooner or later. On the other hand, if node A is not in the neighbourhood of node B, removing node M results in node B to receive a packet from a distant node. Since node B constantly uses the acknowledge packet (ACK) to confirm the link, it will detect the illegality of this packet and finally drop it.

6.4 Summary

The security deployment of MANET routing operations has been extensively discussed in the literature. Dozens of secure routing protocols have been proposed to offer authentication for two famous MANET routing protocols: AODV and DSR. Yet, these discussions mainly focus on applying the existing cryptographic primitives to routing protocols and crossing our fingers to see if it will work. Therefore, the existing secure routing protocols are not working properly in the sense of achieving security and efficiency.

In this chapter, we presented comparatively efficient authentication schemes for both AODV and DSR. We used ID-based online/offline signature and HORS one-time signature to implement a strong authentication paradigm from ARAN. By this means, we are able to protect AODV protocol against modification, impersonation and fabrication attacks. Then we utilised our ID-based multisignature scheme to provide efficient signature aggregation and authentication for DSR protocol. In addition, our ID-based multisignature can be easily adapted from an ID-based online/offline scheme. Hence, one single signature scheme can be used by two distinct routing protocols.

The node n_t $(1 \leq t \leq L)$ in G receives the signature $(\widetilde{V_{t-1}}, \widetilde{R_{t-1}})$ from its previous hop, where

$$\begin{aligned} \widetilde{V_{t-1}} &= \sum_{j=1}^{t-1}(C_j + \sigma_j P_{pub}) \\ &= \sum_{j=1}^{t-1} D_j + H_1(RREQ)\sum_{j=1}^{t-1} r_j P_{pub} \\ R_{t-1} &= \sum_{j=1}^{t-1} r_j P \end{aligned}$$

Signing. The node n_t does the followed:

1. randomly chooses $x_t, r_t \in \mathbb{Z}_q^*$
2. computes $C_t = D_t - x_t P_{pub}$, $R_t = \sum_{j=1}^{t-1} R_j + r_t P$
3. compute $\sigma_t = H_1(RREQ)r_t + x_t$

With previous received signature $(\widetilde{V_{t-1}}, \widetilde{R_{t-1}})$, the current signer computes:

$$\begin{aligned} \widetilde{V_t} &= \widetilde{V_{t-1}} + (C_t + \sigma_t P_{pub}) \\ &= \sum_{j=1}^{t} D_j + H_1(RREQ)\sum_{j=1}^{t} r_j P_{pub} \end{aligned}$$

The final signature is $(\widetilde{V_t}, \widetilde{R_t})$.

Verifying. The node n_t checks if the equation holds

$$e(\widetilde{V_{t-1}},\ P) = e(\sum_{j=1}^{t-1} Q_j + H_1(RREQ)R_t,\ P_{pub})$$

Figure 6.7: Detailed Algorithm for DSR *RREQ* packet

Chapter 7

Conclusions and Future Work

In this thesis, we focus on the deployment of security features for mobile ad hoc network routing procedures. Our motivation is to provide authentication, integrity and non-repudiation for routing in MANET. We identified three difficulties we had to face in our design: inexistence of trusted third party, resource constraint mobile nodes, and variety of routing protocols. We employed them as guidance in design of our algorithms later.

To achieve a comprehensive understanding of the application domain, we conducted a thorough study over the characteristics of mobile ad hoc networks. We delved into the routing operations and data structure of AODV and DSR protocol, so that our later design could take advantage of the nature of routing protocols.

We reviewed the literature and identified four categories of attack which are generally regarded as the most serious attacks in disrupting routing operations. By analysing their attacking approaches and existing countermeasures, we were able to come up with the security requirements to be achieved in mobile ad hoc networks.

Using the security requirements as a scale, we studied some existing secure routing protocols and justified their performance according to our security requirements. We noticed that the security of the existing proposals is not established from a realistic point of view. Some of the assumptions, such as the pre-establishment of security associations, and the requirement of time synchronisation, generally conflict to the characteristics of mobile ad hoc networks. However, we also observed the strongpoint provided by ARAN's authentication scheme, where we can achieve sender authentication, hop-by-hop authentication, integrity and non-repudiation.

To enable the design of algorithms, we formalised the cryptographic primitives to be used. The digital signature which has long been used to provide authentication, integrity and non-repudiation is recognised as our primary goal. In addition, we identified three categories of digital signatures suitable to our application domain.

- An ID-based signature scheme is able to solve the public key distribution problem in MANET;
- An Online/offline signature scheme is efficient in signature generation;
- A One-time signature scheme is efficient in both signature generation and verification;
- A Multisignature scheme can reduce the signature size through signature aggregation. Besides, By combining with ID-based signature scheme, it will become especially suitable to secure DSR protocol since the resulting route is the list of public keys of the multisignature generated by nodes among the route. Hence through verifying the signature, the route is authenticated.

Thus, we formalised the definition of the ID-based online/offline signature schemes and ID-based multisignature schemes. Observing the similarity between these two schemes, we provided a generic construction from an ID-based online/offline signature schemes to an ID-based multisignature schemes, which enables the above transformation. Hence, an ID-based online/offline signature scheme design for AODV protocol can also provide the same security features to DSR protocol through the transformation. This feature solves the diversity problem of MANET routing protocols identified in the first chapter.

To show the actual construction, we proposed two concrete ID-based online/offline signature schemes and transformed the second one to an ID-based multisignature scheme using the generic construction algorithm. We also proved the security of our signature scheme as existentially unforgeable under adaptive chosen message and ID attack.

We then presented the installation of our signature schemes over AODV and DSR. For AODV, we combined an ID-based online/offline signature scheme with ARAN's authentication scheme to enable efficient signature generation. We also combined an ID-based online/offline signature scheme and HORS one-time signature scheme with ARAN, in order to enable efficient signature generation and verification. Comparing the two schemes, the first scheme is slower in signature verification but the size of the signature is shorter, whereas the second scheme is more efficient in signing and verification with the cost of larger key size and signature size. For DSR, we presented efficient authentication using ID-based multisignature scheme.

Future Work

We provided two installations for AODV using different signature schemes. Because of the distinct parameters used in these two installations, it is impossible for us to analyse which one is more efficient. We are expecting to conduct an experiment over these two schemes in order to figure out the most efficient construction.

The installation described in Chapter 6 is merely secure against attacks using modification and impersonation. For more complicated attacks such as fabrication and tunneling, it is not possible to be prevented by using cryptographic techniques. In chapter 6, we argued that since nodes cannot preform impersonation attacks, any nodes fabricating the route error packets will be traced using their public keys (node's IP addresses). However, this passive approach offers no protection to nodes already under attacks. How to propose a more sophisticated authentication scheme which defends against those kind of attacks is still an open problem.

The generic construction of an ID-based multisignature scheme from an ID-based online/offline signature scheme is applicable to all the ID-based online/offline signature schemes. However, depending on the algorithm used, the transformation does not guarantee to achieve an efficient multisignature scheme in the sense of signature aggregation and verification. For example, the second ID-based online/offline signature scheme cannot achieve the shortest signature size since it offline signature is in the form of $\frac{1}{r}D_{ID}$. We would like to work on improving this scheme.

In conclusion, we summarised all the possible future work below:

1. Implementation of two secure schemes for AODV routing protocol using online/offline signature and one-time signature respectively. Perform an experiment to justify their efficiency.

2. Improve our authentication scheme so that it would be able to detect more complicated attacks such as fabrication attacks and tunneling attacks.

3. Refine our generic construction from IOS to IBMS so that all the input IOS will end up with an IBMS with the maximum efficiency.

Providing an efficient authentication scheme to secure the routing operations in mobile ad hoc networks has long been an open problem. We hope our research has provided some valuable lessons, which will be of benefit to other researchers.

Appendix A

Glossary

ACK	Acknowledgement
AODV	Ad hoc On-demand Distance Vector
ARAN	Authenticated Routing for Ad-Hoc Networks
ASM	Accountable Subgroup Multisignature
CA	Certificate Authority
CDHP	Computational Diffie-Hellman Problem
CLL-IBMS	Chang *et al.*'s Identity Based Multisignature Scheme
CRHF	Collision Resistant Hash Function
CZK-IBMS	Chen *et al.*'s Identity Based Multisignature Scheme
DBF	Distributed Bellman-Ford Algorithm
DDHP	Decisional Diffie-Hellman Problem
DES	Data Encryption Standard
DSR	Dynamic Source Routing
DoS	Deny-of-Service
ECC	Elliptic Curve Cryptography
ECDLP	Elliptic Curve Discrete Logarithm Problem
EF-CMA	Existential unforgeablility under adaptive Chosen Message Attack
ERR	Error
GDHP	Gap Diffie-Hellman Problem
HORS	Hash to Obtain Random Subset
IBS	Identity Based Signature Scheme
ID	Identity
IBMS	ID-based Multisignature
IOS	ID-based Online/offline Signature
IOSP	Independent One-time Signature Protocol
KGC	Key Generation Center
MANET	Mobile Ad Hoc Networks
MASH-1	Modular Arithemic Secure Hash Algorithm 1
MAC	Message Authentication Code
MD 5	Message Digest Algorithm
MDS	Modification Detection Code
OWHF	One-Way Hash Function
PKG	Private Key Generator
RDP	Route Discovery Packet
REQ	Reply
RREQ	Route Request
RREP	Route Reply
RRER	Route Error
SA	Security Association
SAODV	Secure Ad hoc On-demand Distance Vector
SHA	Secure Hash Algorithm
SOK-IBMS	Sakai *et al.*'s Identity Based Multisignature Scheme
SRP	Secure Routing Protocol
TESLA	Timed Efficient Stream Loss-tolerant Authentication
PKI	Public Key Infrastructure
WH-IBMS	Wu *et al.*'s Identity Based Multisignature Scheme

Table A.1: Glossary

Bibliography

[1] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Elsevier ADHOC Networks Journal*, (2):1–22, 2004.

[2] W. Aiello, S. Lodha, and R. Ostrovsky. Fast digital identity revocation. In *Advances in Cryptology*, volume 1462 of *Lecture Notes in Computer Science: Crypto'98*. Springer-Verlag, 1998.

[3] M. Al-Shurman, S-M. Yoo, and S. Park. Black hole attack in mobile ad hoc networks. In *Proceedings of the 42nd Annual Southeast Regional Conference*, 2004.

[4] B. Ansari and M. A. Hasan. High performance architecture of elliptic curve scalar multiplication. Technical report, CACR, 2006.

[5] N. Asokan, G. Tsukik, and M. Waidners. Server-supported signatures. *Journal of Computer Security*, 1997.

[6] Available from http://csrc.nist.gov/fips/. *FIPS publication 180-1: Secure hash standard*, 1995.

[7] S. Basagni, I. Chlamtac, V.R. Syrotivk, and B.A. Woodward. A distance effect algorithm for mobility (dream). In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'98)*, 1998.

[8] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the First Annual Conference on Computer and Communications Security*. ACM, November 1993.

[9] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall Inc., 1992.

[10] K. Bicakci and N. Baykal. Infinite length hash chains and their applications. In *Proceedings of the 11th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises: WETICE'02*, pages 57–61, 2002.

[11] D. Bleichenbacher and U. M. Maurer. Directed acylic graphs, one-way functions and digital signatures. In *Advanced in Cryptology: Crypto'94*, volume 839 of *Lecture Notes in Computer Science*, pages 75–82. Springer-Verlag, 1994.

[12] D. Boneh, B. Lynn, and H. Shacham. Short signature fromt the weil pairing. In *Proceedings of Asiacrypt '01, Lecture Notes in Computer Sciences*, volume 2248, pages 514–532. MANET working group, 2001.

[13] D. Boneh, I. Mironov, and V. Shoup. A secure signature scheme from bilinear maps. In *Topics in Cryptology: CT-RSA'03*, 2003.

[14] A. Burg. Ad hoc network specific attacks. In *Seminar Ad Hoc networking: Concepts, Applications and Security*. Technische University Munchen, 2003.

[15] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proceedings of the 30th Annual ACM Symposium on the The-ory of Computing*, 1998.

[16] J. Cha and J. Cheon. An id-based signature from gap-diffie-hellman groups. In *Proceedings of Public Key Cryptography - PKC 2003*, volume 2567, pages 1–24. Springer-Verlag, 2003.

[17] C. Chang, I. Lin, and K. Lam. An id-based multisignatures scheme without re-blocking and predetermined signing order. In *Computer Standards and Interfaces*, pages 407–413. Elsevier Science Inc., 2004.

[18] T. W. Chen and M. Gerla. Global state routing: A new routing scheme for ad-hoc wireless networks. In *Proceedings of the IEEE ICC*, 1998.

[19] X. Chen, F. Zhang, and K. Kim. Id-based multi-proxy signature and blind multisignature from bilinear pairings. In *Proceedings of KIISC'2003*, pages 11–19, 2003.

[20] J. Y. Choi. Security problems for ad hoc routing protocols survey paper. Technical report, Indiana University at Bloomington, 2003.

[21] D. Coppersmith and M. Jakobsson. Almost optimal hash sequence traversal. In *Proceedings of the 5th Conference on Financial Cryptography: FC'02*, 2002.

[22] X. Ding, D. Mazzocchi, and G. Tsudik. Experimenting with server-aided signatures. In *Proceedings of Network and Distributed System Security Symposium: NDSS'02*, 2002.

[23] J. R. Douceur. The sybil attack. In *Proceedings of the First International Workshop on Peer-to-Peer Systems: IPTPS02 Workshop*, pages 251–260, 2002.

[24] K. Dunne, E. Roche, D. O'Loghlin, and L. Rhatigan. Bluetooth for ad-hoc networking. Technology Survey, 2005.

[25] C. Dwork, A. Goldberg, and M. Naor. On memory-bounded functions for fighting spam. In *Advanced in Cryptology: Crypto'03*, volume 2729 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[26] S. Even, O. Goldreich, and S. Macali. On-line/off-line digital signatures. In *Proceedings of Advances in Cryptology: Crypto '89*. Springer, 1990.

[27] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology: Crypto'86*, volume 263 of *Lecture Notes in Computer Science*, page 186C194. Springer-Verlag, 1986.

[28] S. Glodwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17:281–308, 1988.

[29] V. Goyal. How to re-initialize a hash chain. Technical report, Cryptology ePrint Archive, 2004.

[30] V. Goyal. *A Solution to the ARP Cache Poisoning Problem*, 2004.

[31] S. Gupte and M. Singhal. Secure routing in mobile wireless ad hoc networks. In *Proceedings of Ad Hoc Networks*, volume 1, pages 151–174. Elsevier, 2003.

[32] N. Haller. The s/key one-time password system. In *Proceedings of the ISOC Symposium on Network and Distributed System Security*, pages 151–157, 1994.

[33] R. Hauser, M. Steiner, and M. Waidner. Micro-payments based on ikp. In *Proceedings of the 14th Worldwide Congress on Computer and Communications Security Protections*, pages 67–82, 1996.

[34] Y. C. Hu, D. Johnson, and A. Perrig. Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications: WMCSA'02*, pages 3–13, 2002.

[35] Y-C. Hu, A. Perrig, and D. Johnson. Efficient security mechanisms for routing protocols. In *Proceedings of Network and Distributed System Security Symposium: NDSS'03*, pages 263–276, 2003.

[36] Y. C. Hu, A. Perrig, and D. B. Johnson. Rushing attacks and defence in wireless ad hoc network routing protocols. In *Proceedings of the 2003 ACM workshop on Wireless*, volume 11.

[37] Y-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the Eighth ACM International Conference on Mobile Computing and Networking (Mobicom 2002)*, 2002.

[38] Y-C Hu, A. Perrig, and D. B. Johnson. Packet leashes: A defense against wormhole attacks in wireless networks. In *Proceedings of IEEE INFOCOM 2003*, 2003.

[39] Y. Huang and W. Lee. Attack analysis and detection in ad hoc routing protocols. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection: RAID'04*. French Riviera, 2004.

[40] K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. NEC Research and Development, 1983.

[41] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *IEEE INMIC*, 2001.

[42] M. Jakobsson. Fractal hash sequence representation and traversal. In *Proceedings of IEEE International Symposium on Information Theory: ISIT'02*, 2002.

[43] D. B. Johnson, D. A. Maltz, and Y. C. Hu. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, 2004.

[44] C. Kaufman, R. Perlman, and M. Speciner. *Network Security: Private Communication in a Public World*. Prentice Hall PTR, second edition, 2002.

[45] T. Kiesler and L. Harn. *RSA Blocking and Multisignature Schemes with No Bit Expansion*, electronic letters 26 edition, 1990.

[46] M. Koblitz. Elliptic curve cryptosystems. In *Mathematics of Computation*, volume 48, pages 203–209, 1987.

[47] L. M. Kohnfelder. On the signature reblocking problem in public-key cryptography. In *Communications of the ACM*, volume 21, 1978.

[48] L. Lamport. Constructing digital signature from a one-way function. Technical Report CSL-98, SRI International, 1979.

[49] L. Lamport. Password authentication with insecure communication. In *Communications of the ACM*, volume 24 of *11*, pages 770–772, 1981.

[50] N. Y. Lee, T. Hwang, and C. H. Wang. The security of two id-based multisignature protocols for sequential and broadcasting architectures. *Inf. Process*, 1999.

[51] B. Libert and J. Quisquater. The exact security of an identity based signature and its applications. In *Cryptology ePrint Archive, Report 2004/102*, 2004.

[52] S. E. D. Lim and X. Ee. Study of secure reactive routing protocols in. mobile ad hoc networks. Technical report, National University of Singapore, 2003.

[53] D. Liu and P. Ning. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *Proceedings of Network and Distributed System Security Symposium: NDSS'03*, pages 263–276, 03.

[54] J. Lopez and R. Dahab. An overview of elliptic curve cryptography. Technical report, Institute of Computing, State University of Campinas, 2000.

[55] Y. Lu, W. Wang, Y. Zhong, and B. Bhargava. Study of distance vector routing protocols for mobile ad hoc networks. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications: PerCom'03*, pages 187–194, 2003.

[56] S. Macali. Efficient ceritificate revocation. In *Proceedings of RSA'97*, 1997. Patent No. 5,666,416.

[57] A. Menezes, P. V. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[58] R. Merkle. A certified digital signature. In *Advances in Cryptology: Crypto'89*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer-Verlag, 1990.

[59] R. C. Merkle. Secrecy, authentication, and public key systems. *UMI Research Press*, 1982.

[60] R. C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology: Crypto'87*, volume 239 of *Lecture Notes in Computer Science*, pages 369–378. Springer-Verlag, 1988.

[61] S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures. In *Proceedings of the 8th ACM Conference on Computer and Communications Security: CCS '01*. Springer, 2001.

[62] P. Michiardi and R. Molva. Ad hoc networks security. *ST Journal of System Research*, 2003.

[63] N. Milanovic, A. Radovanovic, B. Cukanovic, A.Beric, N. Tesovic, and G. Marinkovic. Bluetooth ad hoc sensor networks. White Papers, 2001.

[64] V. Miller. Uses of elliptic curves in cryptography. In *Advances in Cryptology: Crypto'85*, pages 417–426. Springer-Verlag, 1986.

[65] P. Misra. Routing protocols for ad hoc mobile wireless networks. Technical report, The Ohio State University, Computer Science and Engeneering, 1999.

[66] C. J. Mitchell. An attack on an id-based multisignature scheme. Technical report, University of London, Mathematics Department Technical Report RHUL-MA-2001-9, 2001.

[67] K. Q. Nguyen, Y. Mu, and V. Vradharajan. Digital coins based on hash chain. In *Proceedings of the 20th National Information Systems Security Conference*, pages 72–79, 1997.

[68] P. Papadimitratos and Z. J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference: CNDS 2002*, 2002.

[69] V.D. Park and M.S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of INFORCOM*, 1997.

[70] T. P. Pedersen. Electronic payments of small amounts. In *Security Protocols-International Workshop*, volume 1189 of *Lecture Notes in Computer Science*, 1997.

[71] G. Pei, M. Gerla, X. Hong, and C. Chiang. A wireless hierarchical routing algorithm for mobile wireless networks. In *Proceedings of Wireless Communications and Networking*, 1999.

[72] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, volume 24. ACM SIGCOMM Computer Communication Review, 1994.

[73] C. E. Perkins, E. M. Royer, and S. R. Das. *Ad Hoc On-Demand Distance Vector (AODV) Routing*, 2003.

[74] A. Perrig. The biba one-time signature and broadcast authentication protocol. In *Proceedings of the 8th ACM Conference on Computer and Communication Security*.

[75] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient authentication and signing of multicast streams over lossy channels. In *Proceedings of IEEE Security and Privacy Symposium: S&P'00*, 2000.

[76] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient and secure source authentication for multicast. In *Proceedings of Network and Distributed System Security Symposium: NDSS'01*, 2001.

[77] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The tesla broadcast authentication protocol. In *Proceedings of RSA CryptoBytes*, volume 5, pages 2–13, 2002.

[78] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar. Spins: Security protocols for sensor networks. In *Wireless Networks*, volume 8, pages 521–534, 2002.

[79] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13:361–396, 2000.

[80] S. Ramaswamy, H. Fu, M. Sreekantaradhya, J. Dixon, and K. Nygard. Prevention of cooperative black hole attack in wireless ad hoc networks. In *Proceedings of the International Conference on Wireless Networks 2003*, pages 570–575, 2003.

[81] L. Reyzin. *Notes for Lectures 19-20*. Boston University CAS CS 538.

[82] L. Reyzin and N. Reyzin. Better than biba: Short one-time signatures with fast signing and verifying. In *Proceedings of the 7th Australasian Conference on Information Security and Privacy*. Springer, 2002.

[83] R. L. Rivest. *IETF RFC 1321: The MD5 Message-Digest Algorithm*. Internet Activities Board, 1992.

[84] R. L. Rivest and A. Shamir. Payword and micromin-two simple micropayment schemes. In *Proceedings of 1996 International Workshop on Security Protocols*, volume 1189 of *Lecture Notes in Computer Science*. Springer, 1996.

[85] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. In *Communications of the ACM*, volume 21, 1978.

[86] E. M. Royer, S Barbara, and C-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 6:46–55, 2002.

[87] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Proceedings of Symposium on cryptography and Information Security: SCIS 2000*, 2000.

[88] K. Sanzgiri, B. Dahill, D. LaFlamme, B. N. Levine, C. Shields, and E. Belding-Royer. A secure routing protocol for ad hoc networks. *IEEE Journals on Selected Areas in Communications Special issue on Wireless Ad hoc Networks*, 2005.

[89] C. P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology: Eurocrypto'89*, volume 434 of *Lecture Notes in Computer Science*, page 688C689. Springer-Verlag, 1989.

[90] Y. Sella. On the computation-storage trade-offs of hash chain traversal. In *Proceedings of Financial Cryptography 2003*, 2003.

[91] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology: Crypto'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1984.

[92] A. Shamir and Y. Tauman. Improved online/offline signature schemes. In *Proceedings of Advances in Cryptology: Crypto '01*. Springer-Verlag, 2001.

[93] S. Stubblebine and P. Syverson. Fair on-line auctions without special trusted parties. In *Financial Cryptography'01*, 2001.

[94] CISCO Systems. *Internetworking Technology Handbook*, 2002.

[95] S.A. Vanstone. *Next Generation Security for Wireless: Elliptic Curve Cryptography*. Certicom, 2005.

[96] V. Varadharajan, P. Allen, and S. Black. An analysis of the proxy problem in distributed systems. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 255–275, 1991.

[97] X. Y. Wang, D. G. Feng, X. J. Lai, and H. B. Yu. Collisions for hash functions md4, md5, haval-128 and ripemd. In *Short talk presented at CRYPTO'04 Rump Session and IACR eprint Archive*, 2005.

[98] E. W. Weisstein. One-way function. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/One-WayFunction.html, 1999.

[99] T. Wu and C. Hsu. Id-based multisignatures with distinguished signing authorities for sequential and broadcasting architectures. In *Applied Mathematics and Computation*, pages 349–356. Elsevier Science Inc., 2002.

[100] S. Xu, Y. Mu, and W. Susilo. An efficient authentication scheme for manet routing. In *Proceedings of the Embedded and Ubiquitous Computing: EUC 2005 Workshops*. Springer, 2005.

[101] S. Xu, Y. Mu, and W. Susilo. Securing aodv routing protocol using one-time signature. In *Proceedings of International Conference onMobile Ad-hoc and Sensor Networks: MSN'05*. Springer, 2005.

[102] S. Xu, Y. Mu, and W. Susilo. Online-offline signatures and multisignatures for aodv and dsr routing security. In *submitting to ACISP'06*, 2006.

[103] P-W. Yau and C. J. Mitchell. 2harp: A secure routing protocol to detect failed and selfish nodes in mobile ad hoc networks. In *Proceedings of the 5th World Wireless Congress: WWC 2004*, pages 1–6. Delson Group Inc, 2004.

[104] S. Yi, P. Naldurg, and R. Kravets. A security-aware routing protocol for wireless ad hoc networks. In *Proceedings of the 6th World Multi-Conference on Systemics, Cybernetics and Informatics: SCI'02*, 2002.

[105] M. G. Zapata. *Secure Ad hoc On-Demand Distance Vector (SAODV) Routing*, 2004.

[106] M. G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proceedings of the ACM workshop on Wireless security*, 2002.

[107] F. Zhang, Y. Mu, and W. Susilo. Reducing security overhead for mobile networks. In *Proceedings of The 19th International Conference on Advanced Information Networking and Applications (AINA 2005)*, pages 398–403. IEEE Computer Society, 2005.

[108] K. Zhang. Efficient protocols for signing routing messages. In *Proceedings of Network and Distributed System Security Symposium: NDSS'983*.

[109] L. Zhou and Z. J. Haas. Securing ad hoc networks. In *Proceedings of the ACM workshop on Wireless security*, 2002.