國立交通大學

資訊工程學系

碩士論文

有效且適用於無線微型網路的廣播認證方法

An efficient broadcast authentication scheme for wireless sensor networks

研 究 生：張升銘

指導教授：謝續平 博士

中華民國九十三年六月

有效且適用於無線微型網路的廣播認證方法

An efficient broadcast authentication scheme for wireless sensor networks

研 究 生：張升銘　　　　　Student: Shang-Ming Chang
指導教授：謝續平 博士　　　Advisor: Dr. Shiuh-Pyng Shieh

國 立 交 通 大 學
資 訊 工 程 學 系
碩 士 論 文

A Thesis
Submitted to
Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master
in
Computer Science and Information Engineering

June 2004
Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 三 年 六 月

有效且適用於無線微型網路的廣播認證方法

研究生：張升銘　　　　　　　　　　指導教授：謝續平 博士

國立交通大學資訊工程學系

摘要

在無線微型網路下要提供廣播認證機制並不容易，但卻很重要。困難主要來自於無線微型網路是一個資源受制的環境，如有限的記憶體, 頻寬, 電力和計算能力等。不論有多麼困難，廣播認證對於無線微型網路是一個重要的安全機制，接受端必須能辨別出廣播封包是合法的基地台發出抑或是攻擊者假造。

許多研究人員已提出各種廣播認證機制，但這些被提出的方法並不是針對資源受限的環境來設計，所以效率不足是使得這些方法無法直接應用於無線微型網路的主要原因。我們提出一個有效率且適於無線微型網路的廣播認證機制，比較之前無線微型網路的廣播認證方法有許多優點，如整個網路不須時間同步的限制，接受端能立即且獨立地認證封包。

An efficient broadcast authentication scheme in wireless sensor networks

Student: Shang-Ming Chang Advisor: Shiuh-Pyng Shieh

Department of Computer Science and Information Engineering

National Chaio Tung University

Abstract

Providing authentication mechanism for broadcast messages is difficult but important in wireless sensor networks. The challenges come from the resource constraint environment of wireless sensor networks, such as limited storage, bandwidth, energy and processing. Despite facing the challenges, we must provide authenticated broadcast no matter how difficult to achieve. Broadcast messages from the base station to sensor nodes should be authenticated to avoid the forged messages from adversary.

Many conventional schemes for broadcast authentication are not suitable for resource-limited environment. To cope with the problem, we propose an efficient broadcast authentication scheme for wireless sensor networks. The proposed scheme has the advantages that no time synchronization is required and receiver can authenticate packets instantly without buffering packets.

誌　謝

感謝謝續平老師這兩年來的諄諄教導，帶領我進入研究領域的大門，感謝實驗室學長的指導，使我遭遇難題時能迎刃而解，還有感謝實驗室的同學及學弟與我相互討論切磋，讓我獲益良多。

另外也感謝我的父母，全心全力地支持我完成我的學業，感謝我的女友雯諭，陪我走過兩年的研究生活。

Table of contents

List of Figures

List of Tables

Chapter 1 Introduction

Wireless sensor network (WSN) is an emerging sensing technique [25. 26]. The purpose of WSN is to detect the circumstance of interest. WSN can be used for many applications (e.g., health, military, home) [23, 24].

WSN is usually composed of several base stations and thousands of sensor nodes which are resource limited devices with low processing capability, energy, and storage. In WSN, distributing data through wireless communication is usually limited in Bandwidth.

Broadcast authentication is a basic and important security mechanism in WSN, because broadcast is a nature communication method in wireless communication environment. When base stations want to commit commands to thousands of sensor nodes, broadcast is more efficient method than unicast to every node individually.

A message authentication code (MAC) is an authentication tag derived by applying an authentication scheme, together with a secret key, to a message. MAC is an efficient symmetric cryptographic primitive for two-party authentication. However, MAC is not suitable for broadcast communication without additional elaborate design. If we use MAC in broadcast communication, the sender and the receivers share the same secret key. Anyone of the receivers knows the MAC key and could impersonate the sender and forge messages to other receivers. This problem comes from the symmetric property of MAC. That is, both sender and receivers can sign messages. Hence, we need an asymmetric mechanism that sender can sign messages and the receivers can only verify messages to achieve authenticated broadcast.

We know that authenticated broadcast need an asymmetric mechanism like public key signature otherwise any compromised receiver could forge messages from the sender. However, asymmetric cryptographic mechanisms like RSA digital signatures cost expensive thousand times than symmetric ones. It's impractical to use them in resource-limited sensor network. A suggested method is use efficient symmetric primitives as a tool to design a scheme with asymmetric property.

1.1 Requirements

Besides the asymmetric property that is needed for broadcast authentication, designing an efficient broadcast authentication scheme for wireless sensor networks still faces many challenges:

1. **Robust to packet loss.** The wireless communication environment is not reliable. Some packets may be loss during the transmission. The scheme must deal with packet loss problem and should be robust to it.
2. **Short authentication latency.** Many applications of sensor network are real time applications such as real time collection information about current battlefield conditions. To authenticate real time data, the maximum number of additional packets that need to be received before a packet can be authenticated should be small.
3. **Individual authentication.** The receiver should verify the received packets individually without depending on some other packets. Otherwise, the failure of verification causes the related packets cannot be verified too.
4. **Low computation cost.** The computation of scheme should be small, since a large number of receivers need to verify the authentication information, and receivers are sensor nodes which have restricted computation power.
5. **Low communication overhead.** The number of bytes per packets which describe the embed authentication information should be small, since the bandwidth of sensor network is restricted.

6. **Low storage requirement.** Since the storage space of sensor nodes is limited, some data for authentication like key material and signatures stored in memory cannot be too large.

Ideally we would like a scheme that has perfect robustness, has no latency, can be individual authenticated and has an overhead as well as a cost similar to what is found in symmetric cryptographic primitives. In practice however, such a perfect scheme is hard to achieve and a compromise needs to be found between these requirements.

1.2 Related work

There has been many proposed broadcast authentication scheme in the literature. They could be roughly divided into two categories by the cryptographic primitives they use. The first one is signature amortization scheme which use asymmetric primitives like RSA digital signature and distribute the cost of signature over the block of packets. The second one is MAC-based scheme which use symmetric primitives like MAC and design an elaborate way to achieve asymmetric property that needed in broadcast communication setting.

The idea of signature amortization is to sign a whole block of packets for amortization purpose. EMSS [20], hash tree [22], hash chain [11], and expander graph [21] are some proposed schemes, whose main challenges come from packet loss problem. Recently, some researchers [1, 4, 5, 14] propose using erasure code [15, 16, 17] to deal with packet loss. But, these schemes with erasure code will suffer pollution attack [5], a Denial-of-Service attack to erasure code; Distillation code [5] has solved this problem with more communication overhead in each packet. However, these schemes have one limitation that the sender or the receivers must buffer the packets before verifying signatures. So, receivers can not authenticated each packet individually and need larger storage requirement. Because the storage of a sensor node is limited, the buffering problem causes signature amortization scheme not to fit in WSN. Therefore, we prefer using an efficient symmetric cryptographic primitive to achieve asymmetric property that needed in broadcast authentication scheme.

Perrig et al. proposed a very efficient time based stream authentication scheme, called TESLA [19], and provided a tiny version for WSN, called μTESLA [18]. They use pure symmetric primitives to achieve asymmetric property by one way key chain and delay disclosure. However, it has some constraints including time synchronization of whole network, inefficiently unicast the initial trust, and delay authentication.

BiBa [2], HORS [3] are one time signature schemes using one way function. They are more efficient signature schemes than public key signature schemes. The efficiency of one time signature can compare favorably with the symmetric primitives', because the main computations are one way hash function evaluations. This advantage is desirable for designing efficient broadcast authentication schemes. However, they have some drawbacks, including the limited number of signature that one key pair can generate and the large size of public key which could not store in sensor node's memory.

We propose an efficient one time signature scheme for broadcast authentication and improve the large storage problem which is not fit in wireless sensor networks.

Chapter 2 Preliminaries

In this chapter, we first propose the system architecture. Then, we review some cryptographic primitives for authentication and a one time signature called HORS, which is so far the fastest one time signature scheme for signing and verifying. Our scheme can be viewed as an improvement of HORS.

2.1 System architecture

There are several base stations and thousands of sensor nodes in a wireless sensor network. Base station is resourceful while sensor node is resource limited. For simplification, we assume each broadcast message is from the base station to the sensor nodes. Broadcast messages from the sensor nodes can first unicast to base station, which then broadcast the messages to the other sensor nodes. The messages transmitted in a sensor network may reach the destination directly or may have to be forwarded by some intermediate nodes; however, we do not distinguish between them in our scheme.

We assume the base station share pairwise secret key with each sensor nodes, so the public key of base station can securely transmit to each sensor nodes by shared pairwise key.

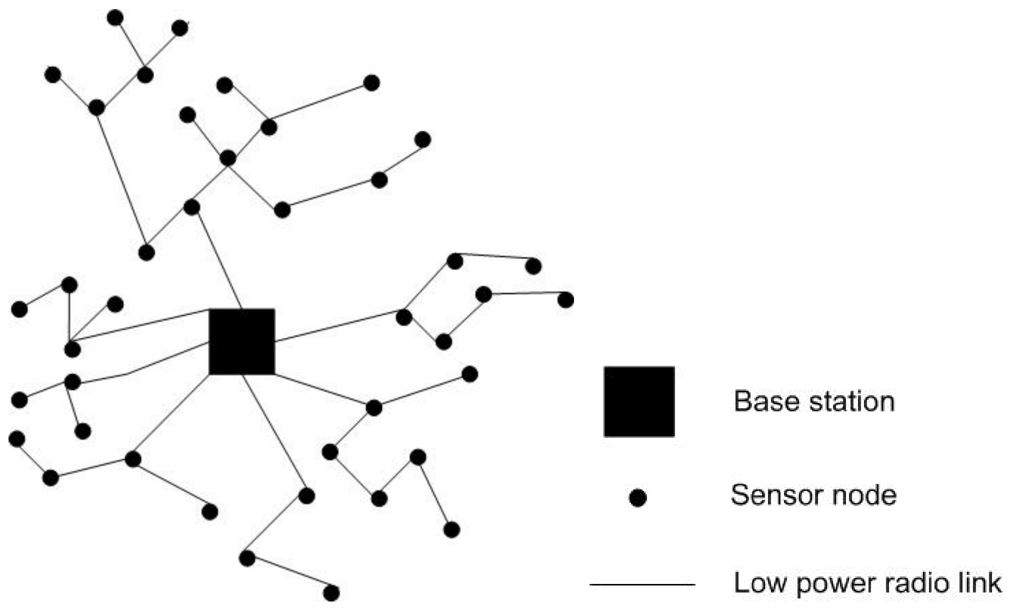

Figure 2-1. Architecture of wireless sensor network

Sensor nodes are resource limited devices. Figure 2-1 shows the characteristics of Berkeley proposed sensor prototype. Our sensor node capacities follow this prototype.

CPU	8-bit 4MHz
storage	8KB instruction flash 512 bytes RAM 512 bytes EEPROM
Communication	916 MHz radio
Bandwidth	10Kbps
Operating system	TinyOS
OS code space	3500 bytes
Available code space	4500 bytes

Table 2-1.SmartDust characteristics

2.2 Cryptographic primitives

We introduce some cryptographic primitives for authentication and some tools for our scheme in this section.

Message Authentication Code (MAC)

A message authentication code (MAC) takes as input a *k*-bit key and a message, and outputs an *l*-bit authentication tag. A receiver who wants to ensure that messages originate from the claimed sender, can verify message authenticity by 1) sharing a secret key with the sender; 2) the sender adds an authentication tag (or MAC) computed with the shared key to every message it sends; 3) the receiver computes the MAC function using the shared key to verify that the authentication tag is correct. Because the same key is shared between the sender and the receiver, this is also a type of symmetric cryptography. A secure MAC function prevents an attacker (without knowledge of the secret key) from computing the correct MAC for a new message. A

MAC achieves authenticity for point-to-point communication, because a receiver knows that a message with the correct MAC must have been generated either by itself or by the sender. So when the receiver gets a fresh message with a correct MAC that it has not generated itself, the message must originate from the sender.

Collision resilient hash functions

A function *H* that maps an arbitrary length message *M* to a fixed length message digest MD is a *collision-resilient hash* function [10] if (1) The description of *H* is publicly known and does not require any secret information for its operation. (2) Given *x*, it is easy to compute *H*(*x*). (3) Given *y*, in the range of *H*, it is hard to find an *x* such that H(x) = y. (4) It is hard to find two distinct messages (M, M')that hash to the same result H(M) = H(M'). More precisely, any efficient algorithm (solving a P-problem) succeeds in finding such a collision with negligible probability.

The collision-resilient hash function is very efficient. It only costs a few micro-second to compute for a Pentium III 800 Hz PC. It costs 1000 times cheaper than asymmetric primitives do.

We propose using collision-resilient hash function, for example SHA-1 and RIPEMD-160, to construct our signature scheme.

Merkle hash tree

Merkle hash tree [7, 8, 9] is a mechanism for calculating a message digest over a group of data items. We construct a binary tree using the hashes of the data items as tree leaves. Then, we compute each internal node value by taking the hash of the concatenation of its two children as figure 2-2 shows.

$$parent = Hash(child_{left} \mid child_{right})$$

Let $Hash(\cdot)$ be a collision resilient hash function.

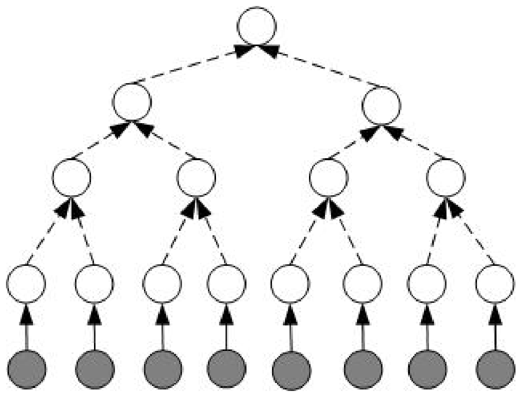
Figure 2-2.Merkle hash tree

One can use Merkle tree as a tool to reduce the authentication overhead needed for a large group of data items. For example, we can sign the root of tree only instead of sign each data item. And then, the verifier can verify the authenticity of every data item by reconstructing the tree and comparing the computed hash value of tree, we called *treehash* here, with the authenticated root value.

However, the verifier cannot reconstruct the tree without all of the data items. If the verifier wants to verify each data item individually, he may compute the *treehash* with the data item and its a*uthentication path*. The *authentication path* of the leaf is the values of all nodes that are siblings of nodes on the path between the leaf and the root. This is illustrated in Figure 2-3.

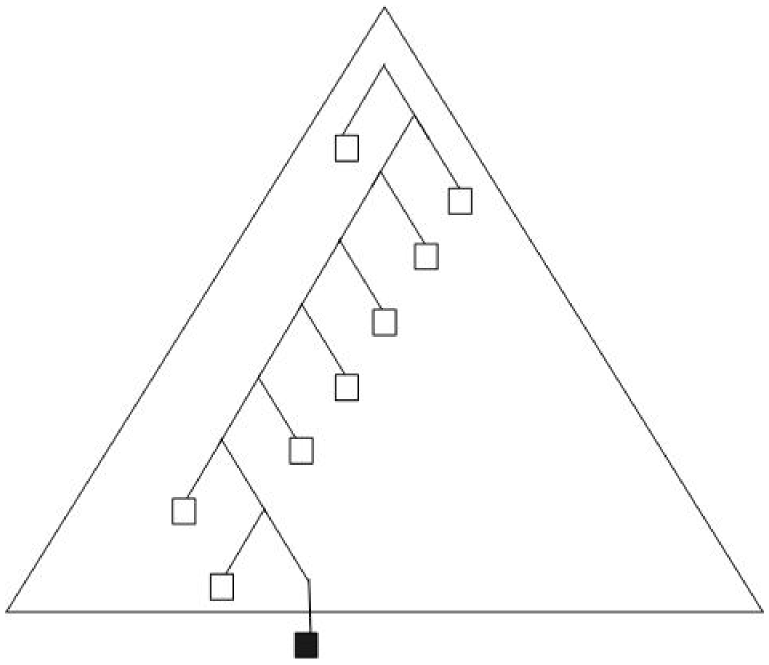
Figure 2-3.Authentication path of Merkle tree

Efficiency of cryptographic primitives

Table 1 shows the efficiency of cryptographic primitives. We can know that symmetric cryptographic primitives like DES and MD5 are more efficient than asymmetric primitives like RSA. The one way hash function is almost as efficient as symmetric cipher.

Algorithm	Time per opration	Operations per second
RSA 1024 sign	8.7 ms	115
RSA 1024 verify	0.48ms	2094
RSA 2048 sign	53.9ms	19
RSA 2048 verify	1.7ms	605
DES	0.5μs	$1.9*10^6$
RC5-32-12	0.2μs	$4.9*10^6$
Rijndael 128	0.7μs	$1.5*10^6$
MD5	1.0μs	$1.0*10^6$
SHA-1	1.5μs	$0.66*10^6$
HMAC-MD5	2.5μs	$0.40*10^6$

All experiments are performed on an 8-byte input size, using the OpenSSL libraries on a 800 MHz Pentium III Linux station

Table 2-2.The efficiency of widely used cryptographic primitives

2.3 One time signature

One-time signature scheme first proposed by Lamport [6] and Rabin [13] were efficient signature schemes based on one way function. One difference between one time signature scheme and public key signature scheme is the number of messages they can sign. One time signature schemes can be used to sign only several messages with a key pairs. While public key cryptography like RSA signatures can be used to sign unlimited number of messages. This is due to the disclosure of private key. The private key of one time signature will be disclosed after signing messages while the private key of RSA digital signature will never be disclosed.

Despite the limit imposed on the number of messages signed, one advantage of such a scheme is that it is generally quite fast. Because one time signature scheme is construct based on one way function and the computation cost of one way function is quite low when comparing with the computation of public key cryptography.

All the previous multiple-time signature schemes follow the general idea that the secret key is used as the input to a sequence of one-way functions which generate a sequence of intermediate results and finally the public key. Onewayness of the function implies that it is infeasible to compute the secret key, or any intermediate result of the computation, from the public key. The private key is self- authenticating value. Motivated by the applications of signatures to stream authentication and broadcast authentication, Perrig in [2] proposes a one-time signature called BiBa, which has the advantages of fast verification and being short signature (perhaps, BiBa has the fastest verification of all previously known one-time signature schemes). The disadvantage of BiBa is, however, the signing time that is longer than in other previous schemes and the public key size is quite larger.

HORS- Hash to Obtain Random Subset

Reyzin and Reyzin in [3] proposed a new one-time (*r*-time) signature, called *HORS* (for Hash to Obtain Random Subset), which algorithm is shown as figure 2-4 and figure 2-5. HORS improves the BiBa scheme with respect to the time overhead necessary for verifying and signing, and reduces the key and signature sizes. This makes HORS the fastest one-time signature scheme available so far. Besides, we note that the security of BiBa can be proved in the random-oracle model while the security of HORS relies on the assumption of the existence of one-way functions and the *subset-resilience* as defined in Appendix A.

The efficiency of HORS is great. It only needs one way hash function evaluation to generate signature. And several one way hash function to verify signature. However, because of the large public key size of HORS (generally 10Kbytes), it is not suitable when we use HORS in sensor network.

```
Key generation
    Input: Parameters l, k, t
        Generate t random l-bit strings $s_1, s_2, ..., s_t$
        Let $v_i=f(s_i)$ for $1 \le i \le t$
    Output: $PK = (k, v_1, v_2, ..., v_t)$ and $SK = (k, s_1, s_2, ..., s_t)$

Signing
    Input: Message m and secret key $SK = (k, s_1, s_2, ..., s_t)$
        Let $h=Hash(m)$
        Split h into k substrings $h_1, h_2, ..., h_k$, of length $\log_2 t$ bits each
        Interpret each $h_j$ as an integer $i_j$ for $1 \le j \le k$
    Output: $\sigma = (s_{i1}, s_{i2}, ..., s_{ik})$

Verifying
    Input: Message m, signature $\sigma = (s_{i1}, s_{i2}, ..., s_{ik})$
            and public key $PK = (k, v_1, v_2, ..., v_t)$
        Let $h=Hash(m)$
        Split h into k substrings $h_1, h_2, ..., h_k$, of length $\log_2 t$ bits each
        Interpret each $h_j$ as an integer $i_j$ for $1 \le j \le k$
    Output: "accept" if for each j, $1 \le j \le k$, $f(s'_j) = v_{ij}$;
            "reject" otherwise
```

Figure 2-4.HORS algorithm

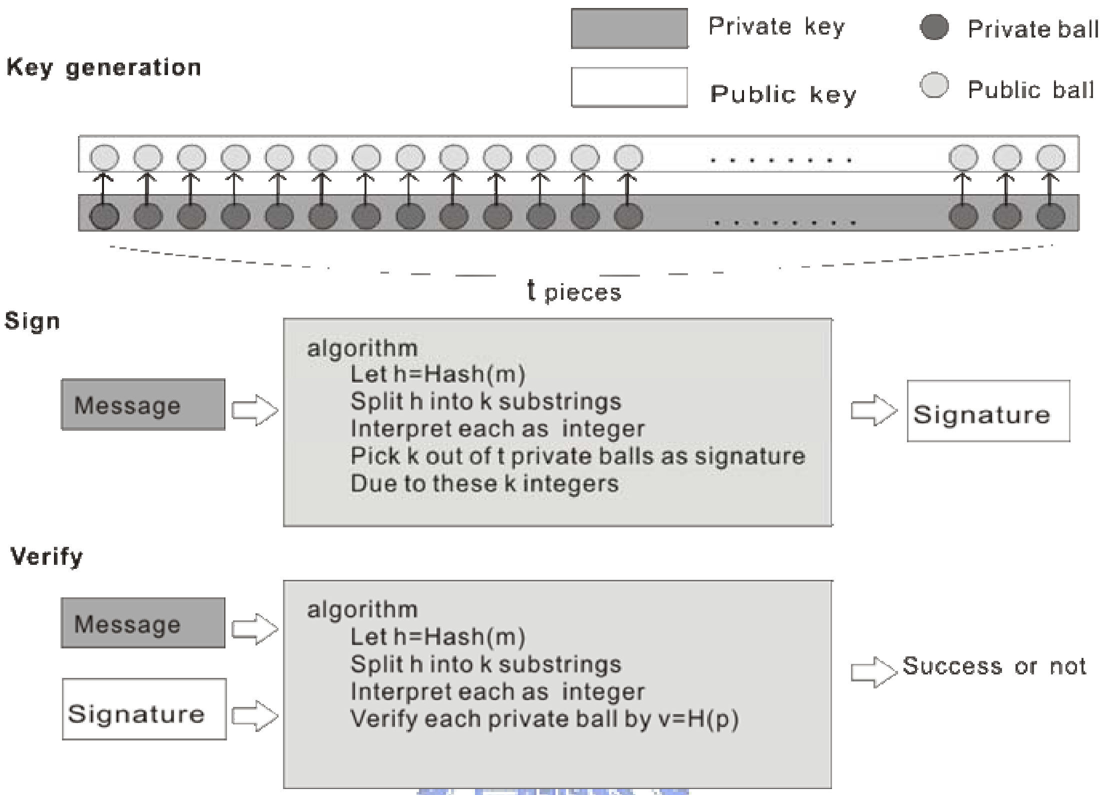

Figure 2-5.HORS algorithm illustration

In a typical example of HORS, we take parameters *l=80, t=1024*, and *f*$_l$*t*=160. The private key is equal to 10Kbytes, which is computed from 1024*80bits and the public key size is equal to 20Kbytes, which is computed from 1024*160bits. Because we assume the sender is base station and it is resourceful, the private key size is not large for sender. But the public key stored in sensor node is too large for sensor nodes' memory.

Chapter 3 Proposed scheme

We design a one-time signature scheme so that receivers (sensor nodes) can authenticate the source of broadcast messages from sender (base station) in wireless sensor networks. The computation cost of our scheme is very lightweight for sensor nodes and its asymmetric property is what we need to achieve broadcast authentication (discussed in chapter 1). Moreover, we mitigate the general drawback of one time signature, which is very large key size than other signature schemes. Large key size requires large storage space, which the sensor nodes can not afford. The proposed scheme reduces the storage requirement efficiently Another drawback of one time signature is that sender can only sign several message with one key pair. We proposed a rekeying mechanism for this.

3.1 Signature scheme

The idea of our scheme is using more computation cost to trade for less storage requirement and less communication overhead than HORS. This trade-off is worth because the storage resource is more precious than computation power for sensor node. Especially, the additional computation cost of our scheme is several hash function computation, which is very lightweight computation overhead. So, the proposed scheme is desirable for WSN.

In the following, we first explain the basic idea of our scheme. Then, we propose the generalized scheme. Finally we propose a rekeying mechanism for our scheme.

3.1.1 The basic idea

For signing and verifying the messages, the signer must first generate the key pair. The key pair includes the *private key* which consists of *t* random numbers, and the *public key* which consists of *t* hash values of these *t* random numbers. For convenience, we call these t random numbers *private balls* and call their hash values *public balls*.

These private balls have a good property that the verifier can efficiently authenticate them based on the public balls, and that it is computationally infeasible for an adversary to find a valid private ball given a public key. The generation of public balls in HORS is to use the one way hash function F as a commitment scheme. Given a set of private balls, the public ball is $p_i = F(r_i)$. If the verifier learns function F in an authentic fashion, it can easily authenticate r_i by verifying $p_i = F(r_i)$.

We know that the public key is composed of *t* public balls. To reduce the public key size, we can reduce the ball size or the number of balls. Because the length of public ball is related to the security strength of hash function, we cannot reduce public ball size. We reduce the number of public balls we needed. Our solution to reduce the public key size (that is reduction of public ball number) is to use a Merkle hash tree for authenticating private balls instead of one way hash function. We place the original public balls at the leaves of a binary tree and compute each internal node as the hash of the concatenation of the two child values. The root node of the hash tree is used as the new public key, and hence the public key is small.

Because we change the way of generating public key, this also change the way of signature generation and verification. The signer generate signature as before, which is *k* picked private balls out of *t* private balls, but the signer needs to add some additional public balls to the signature. The additional public balls are *authentication* *path* of each picked balls. With authentication path, the verifier can verify each picked balls by reconstructing the path from picked private ball to the root of the hash tree. The figure 3-1 shows this method.

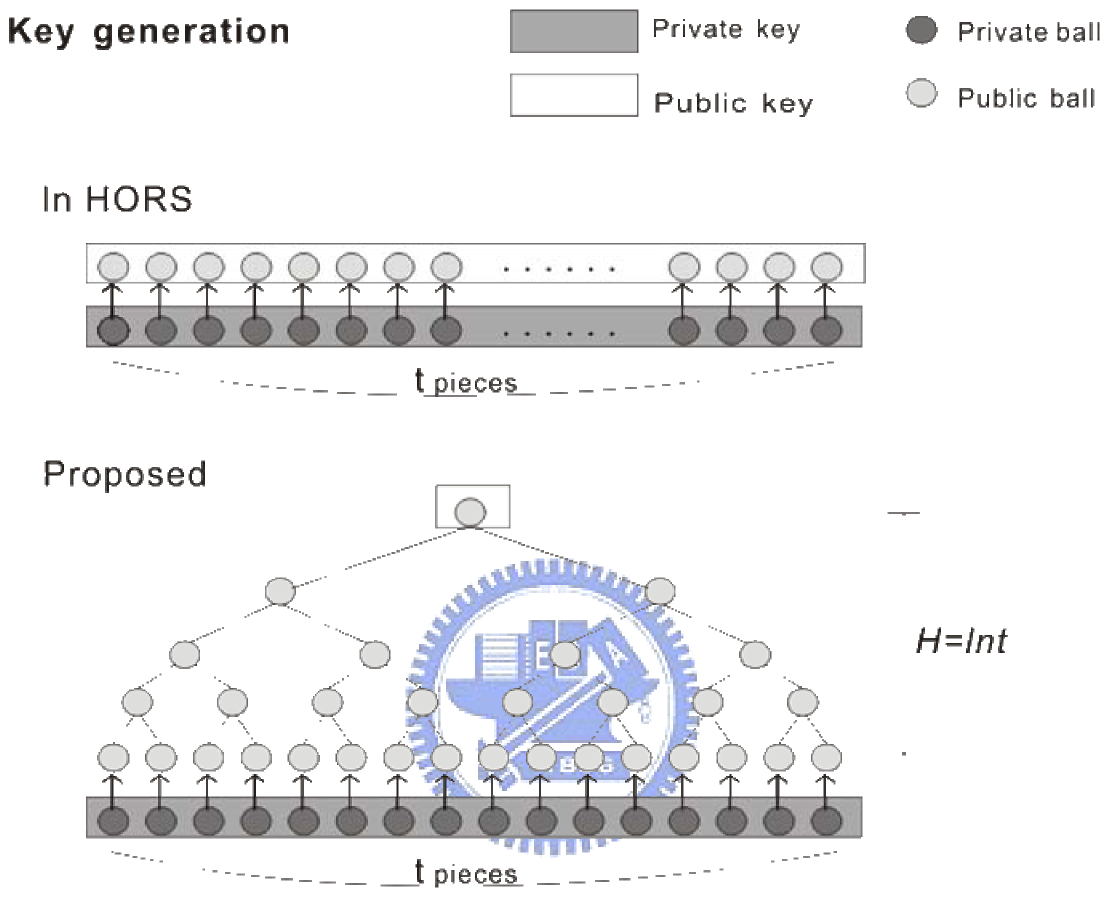

Figure 3-1.Key generation procedure

One security flaw occurs when attackers take disclosed private ball *i* to pretend to be private ball *j*. We can not distinguish two private balls in the same tree. We use the uniqueness of each leaves' authentication path to solve this problem. When receiver gets the message and its signature, he takes the following actions for distinguish different private balls. For each private ball, concatenate the public balls of authentication path. Then, apply hash function to this concatenate value to get a hash value and take this hash value as the identity of the private ball. The disclosed private ball's identity will store in receiver's memory. When verifying the coming signature, we first check the identity of each private ball.

3.1.2 The generalized scheme

We generalize our scheme as following. We construct many small hash trees of height h that contain 2^h private balls. The public key would then contain all the root nodes of all small hash trees, and hence we reduce the size by a factor of 2^h. But, for authenticating each private ball, the signer adds the authentication path of each private ball, which has h verification nodes. Hence the signature size is expanded by a factor of h.

Instead of constructing one Merkle tree only, we construct many Merkle trees owing to the lowest storage requirement. This generalized scheme comes from a fact that the public key size decreases, the signature size increases. If we only construct one Merkle tree, the size of public key plus the size of signature could not be the smallest sum. We should find an optimal balance between them due to the lowest storage requirement which are the sum of public key size and signature size.

The proposed scheme has three phases, which are initial phase, sign phase, and verify phase. The three phases will illustrate as below:

First, the sender (base station) generate key pairs (shows as figure 3-2) include private key and public key. Private keys are t l-bit random number generated by pseudorandom generator. Public key are d hash value which generated by PUBLIC_KEY_GENERATION as figure 3-3 shows. Sender use private key to sign a message as figure 3-4 shows. Receivers use public key of sender to verify the signature of message as figure 3-6 shows. We show system parameters below:

System parameters
t: private ball number
k: signature ball number
d: public ball number

l: ball size (bits)
r: r-subset resilient

Key generation

In this phase, we generate a key pair, including a private key and a public key. The private key is composed of *t* *l*-bit random numbers generated by pseudorandom generator and the public key is generated from these t random numbers. First, we take *t* random numbers as the input to one way hash function to generate *t* hash values. Then, we separate *t* hash values into *d* group. So there are *t/d* values in each group. Finally, we use these *t/d* values as the leaves of binary tree and compute each intermediate node as the hash of the concatenation of the two child values. Thus, we can get *d* Merkle trees, whose roots compose our public key. We note that the original public key of HORS is *t* hash values generated from *t* random numbers while our public key is *d* Merkle tree's root. In a typical case, *t* = 1024 and *d* = 32.

```
KEY GENERATION
Input: $parameters: t, k, d, l$

Output: key pair $K_{pri} = (k, s_1, s_2, ..., s_t)$, $K_{pub} = (k, v_1, v_2, ..., v_{\ln d})$

    1. Random generate $t$ $l$-bits random number
    $(s_1, s_2, ..., s_t)$ as private key
    2. $K_{pub} = PUBLIC\_KEY\_GENERATE(k, d, K_{pri})$
3. Public key distribution
```

Figure 3-2.Algorithm of key generation

```
PUBLIC_KEY_GENERATE
Input: $parameter\ k, d\ and\ K_{pri}$

Output: $K_{pub} = (k, v_1, v_2, ..., v_{\ln d})$
    1. Use $t$ balls as preimage of leaves to build c Merkle trees with height ($\ln t$)

    2. $\ln d$ tree root as public key $K_{pub} = (k, v_1, v_2, ..., v_{\ln d})$, and
  each public key corresponding to a sequence period.
```

Figure 3-3.Algorithm of public key generation

Broadcasting authenticated messages

When base station broadcast messages to sensor nodes, base station must sign the messages. To sign the message *m*, we first compute *h* = *H(m)*. Then, we separate the hash value *h* into *k* pieces and regard these pieces as integers, so we get $(i_1, i_2, ..., i_k)$ between zero and *t*-1. Third, we combine these integers to form the subset of $\{0,1,2,...,t-1\}$ of size at most k. Each integer is an index of private balls $(r_1, r_2, ..., r_t)$. Therefore, we can pick *k* private balls due to this message *m*. These *k* picked private balls $(r_{i1}, r_{i2}, ..., r_{ik})$ plus their authentication path are used as the signature of this message *m*. Here, the *authentication path* is the additional communication overhead compared to original HORS. The duplicate path was send only once for better performance. We will discuss the duplicate authentication path below.

```
SIGNATURE_GENERATION
Input: Message $m$ and private key $K_{pri} = (k, s_1, s_2, ..., s_t)$
Output: signature $\sigma = \{a_{i1}, a_{i2}, ..., a_{ik}, bs\}$, where $a_i = (s_i, ap_i)$
(ap: authentication path of the ball)
    1. compute $h = Hash(m)$
    2. split $h$ into $k$ pieces $(h_1, h_2, ..., h_k)$ of length $\ln t$ bits each
    3. interpret each $h_j$ as an integer $i_j$ , $1 \le j \le k$
```

Figure 3-4.Algorithm of signature generation

3.1.2.1 The duplicated authentication path

For verifying each private ball, the sender must send additional nodes called *authentication path*. These additional nodes of private balls could be sent repeatedly. Here is an example. We first send the ball s$_0$, and its authentication path $\{v_1, m_{23}, m_{47}\}$, and then send s$_1$, and its authentication path $\{v_0, m_{23}, m_{47}\}$. Two balls m_{23}, m_{47} are duplicated and should send only once. Moreover, v_0 can be computed by $v_0 = Hash(s_0)$. This shows that when we send a direct neighbor node of disclosed private ball, we send no additional node for this private ball. The generalized idea is as follow. If the nodes belong to the first common parent is at height *e*, the additional nodes at height higher than e send only once. The closer the private balls are, the more duplicated additional nodes we save. (ps: The upper bound of the sum of authentication path is min{*r***k***h*, the whole tree})

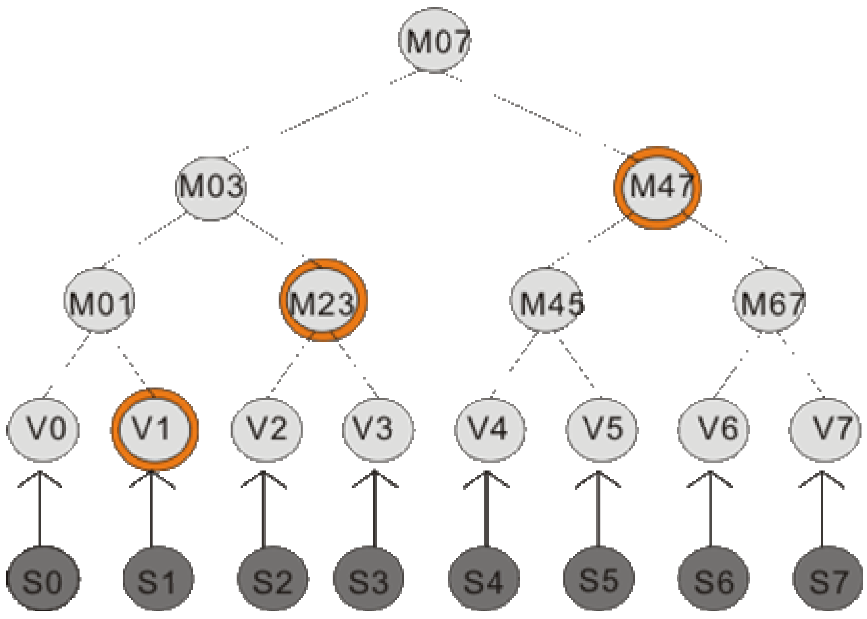

Figure 3-5.Duplicated path illustration

Authenticating broadcast messages

When the sensor nodes receive the broadcast messages, it needs to ensure that the broadcast messages come from the authenticated sender. Sensor nodes verify the signature of message m' by the following procedure. First, because the sender will rekey periodically, the verifier must decide which public key of sender should be used to verify received signatures. The verifier checks which sequence period of public key the sequence number falls into. Second, it computes $h' = H(m')$. Then, we separate the hash value *h* into *k* pieces and regard these pieces as integers $(i_1, i_2, ..., i_k)$ between zero and *t*-1. Each integer is an index of private balls $(r_1, r_2, ..., r_t)$. Third, we check the identities of balls by uniqueness of authentication path as discuss in section 3.2.1. Forth, receivers verify each ball with its *authentication path* and the public key. They compute the *treehash* with the private ball and its a*uthentication path*, then check whether this *treehash* equals to the public key. If it is true, we can assure that the private ball belongs to the authenticated sender. The verification algorithm is shown as figure 3-6.

```
SIGNATURE_VERIFICATION
Input: message $m$, signature $\sigma$, and public key $K_{pub} = (k, v_1, v_2, ..., v_{\ln d})$
Output: {true, false}
    1. check if bs in current sequence period
    2. $h = H(m)$
       Split h into k piece, $(h_1, h_2, ..., h_k)$ of length $\ln t$ bits each
       Interpret each $h_j$ as an integer $i_j$ , 1<j< k
           $TN_j = i_j / (t / d)$
           Check $i_j$ with pairs $(i, TN, Hash(AP))$
           If index $i_j$ already exists, check if $Hash(AP_j) = Hash(AP)$;
           else check each $Hash(AP_j) \neq Hash(AP) \in TN_j$
    3. Use Merkle tree to verify balls
        if ( $TreeHash(r_j, AP_j) = p_{TNj}$ )
            then output true;
        else output false;
```

Figure 3-6.Algorithm of signature verification

3.2 Rekeying mechanism

Because proposed scheme can only sign *r* messages with one key pair, when we sign more than r messages, we should sign with another key pair instead. Therefore, we propose a rekeying scheme for this situation.

If one key pair can sign r messages, we set the duration is r. the sequence period of the first public key is 0 ~ r-1, the sequence period of the second public key is r ~ 2r-2, and so on (shows as figure 3-7). Every key pair can be used in the duration of sequence numbers.

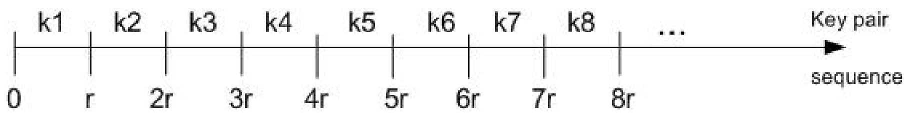

Figure 3-7.Sequence period of a public key

When sensor nodes receive the broadcast messages, they first check which sequence period the sequence number of message belongs to.

We discuss a more efficient public key distribution method here. Because each sensor node shares pair-wise key with base station, base station (sender) can unicast the public key to each sensor node using authenticated channel first time. Afterward, sender distributes the public key by authenticated broadcast where the new public key material was signed with the old private key.

Chapter 4 Discussion

4.1 Security analysis

The security strength of the proposed scheme is based on some security parameters, including the private ball size *l,* the private ball number *t,* the signature ball number *k,* and the number of messages that one key pair can be used to sign, *r* (as defined in Appendix A). We discuss these security parameters below.

Theorem 1. Security level is based on security parameters *l, k, t, r*. The parameter *l* decides the security strength against brute-force attack. The parameters *k, t, r* decide the security strength against chosen message attack. For defending chosen message attack, we provide $k(\log t - \log k - \log r)$ bits of security.

Proof:

Let *f* be a one-way function operating on *l*-bit strings, for a security parameter *l*. And f_l is the length of the one-way function output on input of length *l*. Attackers can do brute force attack against one way function by deriving private ball r_i from the Merkle tree's leaves $leaf_i$. The private ball r_i and Merkle tree's leaves $leaf_i$ has one way relationship as $leaf_i = f(r_i)$. The suggested security parameter *l* is 80bits in HORS.

Because the sender will disclose the private balls with signatures, attackers may do a chosen message attack by collecting the private balls for forging the signature. We assume that the attacker of obtains signatures on *r* messages of its choice (but the choice is independent of *Hash*), and then tries to forge a signature on an any new message *m* of its choice. We are interested in the probability that the adversary is able to do so without inverting the one-way function *f*. it is quite easy to see that this probability is at most $(rk/t)^k$ for each invocation of *Hash*, i.e., the probability that after *rk* elements of *T* are fixed, *k* elements chosen at random are a subset of them. In other words, we get $k(\log t - \log k - \log r)$ bits of security. ◊

4.2 Selection of parameters

We discuss the system parameters in this section. These parameters influence security strength and performance of our scheme. In general, higher security strength brings lower performance and vice versa. Our strategy is to provide enough security strength as we need and we assume 64 bits security is enough. We take security parameters *l*=80 against brute-force attack and t =1024, k=16, r=4 to provide 64 bits security against chosen message attack.

First, we explain the meaning of parameters in the following; *k*: the number of balls in a signature; *h*: the cost of computing a *hash* function; *d*: the number of public balls as the public key; f_l: the public ball size (bits); *r*: the number of signatures that one key pair can generate; *l*: the private ball size (bits); h_1: the size of private ball's identity. We discuss how the parameters influence the performance below.

Lemma 1. Given the parameters *k, h, d,* f_l*, r, l* h_1, the storage requirement, computation cost and communication overhead of system are related to tree height *h*.

Computation cost (of receiver): $k * h$

Communication overhead: $d * f_l + r * (kl + f_l kh)$

Storage requirement (of receiver): $d * f_l + kl + f_l kh + rk * h_1$

Proof:

The communication overhead for the whole life of a keypair includes the public key size, which consists of d b-bit root, and *r* signature size, which each consists of k private balls and k authentication path. The storage requirement equals the public key size, one signature size and rk private ball's identity. Receiver need to compute the root using leaf and its authentication path. Verifying one private ball needs *h* hash computation. Receiver need to verify *k* private balls for one signature. So, the number of hash computation is equal to $k * h$. ◊

Lemma 2. Given *t* private balls and *d* public balls as public key, the tree height is related to public key size

$$TreeHeight = \ln t - \ln d$$

Proof:

The height of tree is decided by the number of private balls and the number of public tree root. The number of leaves of a tree is *t/d*. The height of the binary tree with *t/d* leaves is $\ln t / d$, equal to $\ln t - \ln d$. ◊

With the same amount of leaves, we construct several trees instead of one tree in order to amortize the height of tree over several trees. We do so due to a fact that the higher tree is, the lower public key size is but the more additional public balls should be sent. We want to find an optimal balance between public key size and signature size. The public key and signature will store in sensor node's memory at the same time, so the sum of their size should be as small as possible. In that case, how many trees we should construct for the smallest storage requirement for sensor node? We discuss in Theorem 2.

Theorem 2. Given the parameters *l, f_l, k, t* for enough security strength, we construct *d* trees in our scheme and the parameter *d* is decided by $\min\{d * f_l + kl + f_l k(\ln t - \ln d)\}$.

Proof:

The number of trees we construct is decided by the lowest storage requirement. We know that the storage requirement for sensor node is $d * f_l + kl + f_l kh + rk * h_1$ from lemma 1 and lemma 2. Therefore, given *l, f*

min

We know that the higher tree is, the more additional nodes sent. Then, what if we increase the degree of Merkle tree for constructing a tree with low height? This is not a good idea due to theorem 3.

Theorem 3. Given t leaves, the 2-degree Merkle tree has the lowest upper bound of additional nodes needed to be sent.

Proof:

The upper bound of additional nodes we transmit in a signature is $(d-1)*h$, which is equal to $(d-1)*\log_d t$. Given t leaves, we desire $\min\{(d-1)*\log_d t\}$. The value $(d-1)*\log_d l$ is the minimum when degree is equal to 2. ◊

4.3 Case study

Real time distribution of traffic data

A municipality wishes to collect traffic information from sensors distributed over the streets. The sensors need to authenticate the command from the base station (the municipality) and transmit sensed data through secure channel (with pair-wise keys shared between sensor nodes and base station) back to base station. The system requirements are as follows:

The data rate of the stream is about 10 Kbps, about 20 packets of 64 bytes each are sent every second. The packet drop rate is at most 5% for some recipients, where the average length of burst drops is 5 packets. The verification latency should be less than 10 seconds.

Our proposed scheme helps the sensor nodes to authentication the command from the base station. We set the system parameters $l=80$, $t=1024$, $k=16, r=4$, $f_l t=160$ for 64 bits security which can be computed from theorem 1. This means the attackers need to perform 2^{64} hash computations during a key-pair life to forge a signature. In BiBa [2], they provide 58 bits security for real time stock quotes application. So we consider 64 bits security is enough. In this case, the optimal public key size is 640 bytes which are 32 tree roots comes from theorem 2. The figure 4-1 shows the optimal public key size we choose due to the minimum storage requirement.

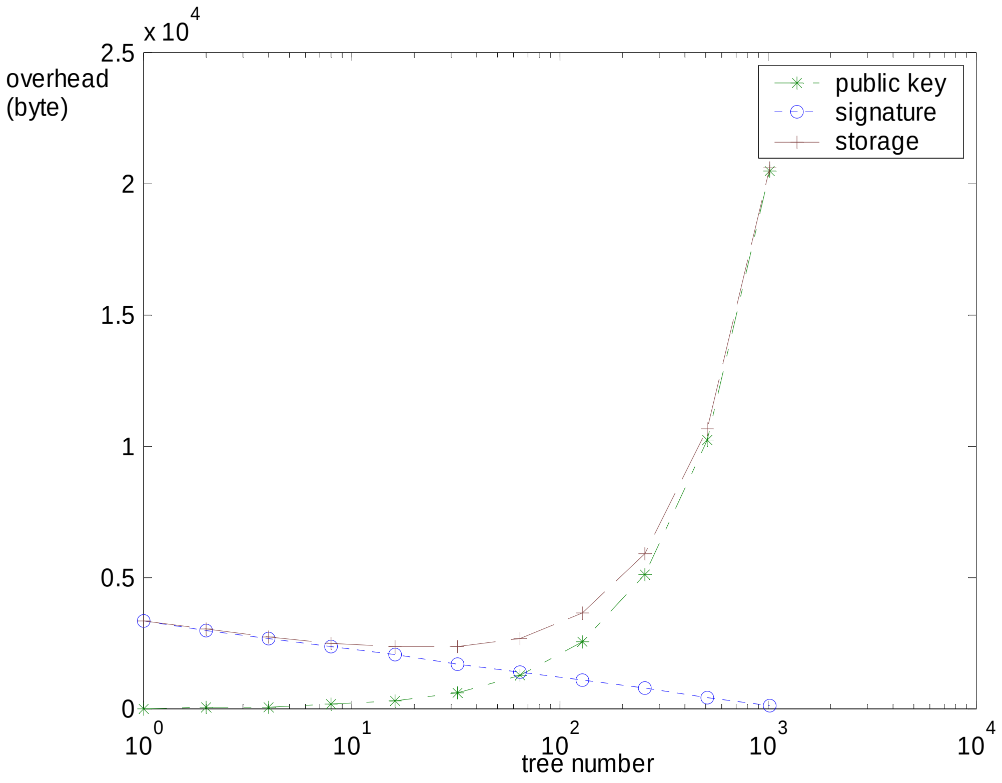

Figure 4-1.The optimal public key size

When we decide the optimal number of trees we should construct, we generate the key pair using key generation algorithm (figure 3-2). In this case, the key generation is illustrated by figure 4-2.

Private key: 1024 80-bit random numbers

Public key: 32 160-bit hash values

32 trees, each has $t/d=32$ leaves and height = $\ln t - \ln d = 5$

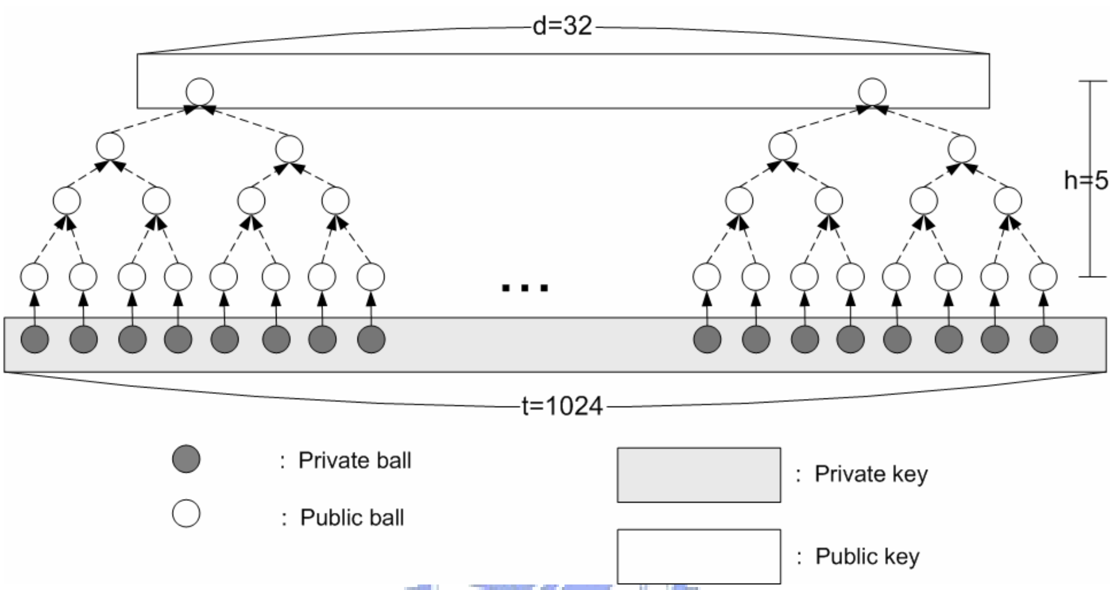

Figure 4-2. The key pair of our scheme

4.4 Comparison

μTESLA [18] is so far an efficient broadcast authentication protocol in wireless sensor networks. Therefore, we compare our scheme with it. When comparison with μTESLA, we have four advantages over μTESLA, which are listed below:

1. No time synchronization needed

In μTESLA, sender and receivers must synchronize time first and use disclosure delay to achieve asymmetric property needed in broadcast authentication. Our scheme use key pair including private key for signing messages and public key for verifying messages to achieve asymmetric property. We have no requirement for time synchronization, which is not always practical in large sensor network.

2. No receiver buffer needed

In μ TESLA, the receivers have to buffer the packets until the corresponding MAC key disclose. Instead, the receivers need no buffer in our scheme.

3. Individual authenticate

Receivers can individual authenticate the received packets without waiting for another packets. In μ TESLA, the receivers must wait the packet with disclosed MAC key.

4. Instant authenticate

Receivers can instant authenticate the received packets instantly in our scheme while receivers must wait for the packets with disclosed MAC key in In μ TESLA.

We compare our scheme with other two efficient one time signature schemes, BiBa [2] and HORS [3], here. For a general case, we take the system parameters as following for the same security level, t=1024, k=16, r=10, h=5. The proposed scheme performed better than BiBa and HORS in three criterions, which are computation overhead, communication overhead and storage requirement.

	BiBa	HORS	Proposed scheme
Generation overhead (hash computation)	2048	1	h=5
Verification overhead (hash computation)	100	1+k=17	h=5
Communication (bytes)	5250	5250	288
Storage (bytes)	5152	5152	192
Energy cost	Large	Large	Little
Time synchronization	Yes	No	No

Table 4-1.Comparison with other one time signature schemes

Chapter 5 Conclusion

In this paper, we proposed an efficient broadcast authentication scheme for wireless sensor networks. The proposed broadcast authentication scheme includes a signature scheme and a rekeying mechanism. The signature scheme which can be viewed as an improvement of one time signature scheme HORS. We reduce the large key storage requirement of HORS by using Merkle hash tree construction to generate the key pair. The idea of reducing key size is to take more computation cost to trade for less storage requirement, and the signature size is a little longer than that in HORS. We also propose a simple but efficient rekeying mechanism for our scheme.

Our scheme has many nice properties, including individual authentication, instant authentication, robust to packet loss and low overhead in computation, communication and storage.

Reference

1. A. Pannetrat and R. Molva, "Efficient multicast packet authentication," In *Proceedings of the Symposium on Network and Distributed System Security Symposium (NDSS 2003),* Internet Society, Feb. 2003.
2. A. Perrig, "The BiBa one-time signature and broadcast authentication protocol," In *Proceedings of the Eighth ACM Conference on Computer and Communications Security (CCS-8)*, pp. 28 – 37, Philadelphia PA, USA, Nov. 2001.
3. L. Reyzin and N. Reyzin, "Better than BiBa: Short onetime signatures with fast signing and verifying," In *Seventh Australasian Conference on Information Security and Privacy (ACISP 2002)*, July 2002.
4. J. M. Park, E. K. Chong, and H. J. Siegel, "Efficient multicast packet authentication using signature amortization," In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pp. 227 – 240, May 2002.
5. C. Karlof, N. Sastry, and Yaping Li, "Distillation Codes and Applications to DoS Resistant Multicast Authentication," *The 11th Annual Network and Distributed System Security Symposium*, February 2004.
6. L Lamport, "Constructing digital signatures from one-way function," *Technical Report SRI-CSL-98*, SRI International, October 1979.
7. R. Merkle, "A Digital Signature Based on a Conventional Encryption Function," *Proc. CRYPTO'87*, LNCS 293, Springer Verlag, pp 369-378, 1987.
8. R. Merkle, "A Certified Digital Signature," *Proc. CRYPTO'89*, LNCS 435, Springer Verlag, pp 218-238,1990.
9. R. Merkle, "Protocols for public key cryptosystems," *In Proceedings of the IEEE Symposium on Research in Security and Privacy*," pp. 122–134, Apr. 1980.

10. M. Bellare and P. Rogaway, “Collision-resistant hashing:Towards making UOWHFs practical,” *In Advances in Cryptology – CRYPTO ’97*, volume 1294 of Lecture Notes in Computer Science, pp. 470–484, 1997.

11. P. Golle and N. Modadugu, “Authenticating streamed data in the presence of random packet loss,” *In Proceedings of the Symposium on Network and Distributed Systems Security (NDSS 2001)*, pp. 13–22, Internet Society, Feb. 2001.

12. S. Miner and J. Staddon, “Graph-based authentication of digital streams,” *In Proceedings of the IEEE Symposium on Research in Security and Privacy*, pp. 232–246, May 2001.

13. M. O. Rabin, “Digitalized signatures,” In Richard A. Demillo, David P. Dobkin, Anita K. Jones, and Richard J. Lipton, editors, *Foundations of Secure Computation*, pp. 155-168. Academic Press, 1978.

14. J. M. Park, E. Chong, and H. J. Siegel, “Efficient multicast packet authentication using erasure codes,” *ACM Transactions on Information and System Security (TISSEC)*, pp. 258–285, May 2003.

15. J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, “A digital fountain approach to reliable distribution of bulk data,” *In proceedings of ACM SIGCOMM ’98*, September 1998.

16. L. Rizzo, “Effective erasure codes for reliable computer communication protocols,” *ACMCCR: Computer Communication Review*, 1997.

17. M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, “Practical loss-resilient codes,” *In ACM Symposium on Theory of Computing*, pp. 150–159, 1997.

18. A. Perrig, Robert Szewczyk, Victor Wen, David Culler, and J.D.Tygar, “Spins: Security protocols for sensor networks,” *Wireless Networks*, 8:521 – 534, 2002.

19. A. Perrig, R. Canetti, D. Song, and J. D. Tygar, “Efficient and secure source

authentication for multicast," *In Proceedings of the Symposium on Network and Distributed Systems Security (NDSS 2001)*, pp. 35–46, Internet Society, Feb. 2001.

20. A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "Efficient authentication and signature of multicast streams over lossy channels," *In Proceedings of the IEEE Symposium on Research in Security and Privacy*, pp. 56–73, May 2000.
21. D. Song, D. Zuckerman, and J. D. Tygar, "Expander graphs for digital stream authentication and robust overlay networks," *In Proceedings of the IEEE Symposium on Research in Security and Privacy*, pp. 258–270, May 2002.
22. C. Wong and S. Lam, "Digital signatures for flows and multicasts," *In Proceedings on the 6th International Conference on Network Protocols (ICNP '98)*, pp. 198–209, IEEE, October 1998.
23. E. Biagioni and K. Bridges, "The application of remote sensor technology to assist the recovery of rare and endangered species," *In Special issue on Distributed Sensor Networks for the International Journal of High Performance Computing Applications*, Vol. 16, N. 3, August 2002.
24. E. Biagioni and G. Sasaki, "Wireless sensor placement for reliable and efficient data collection," *In Proceedings of the Hawaiii International Conference on Systems Sciences*, January 2003.
25. D. Estrin, "Embedded networked sensing research: Emerging systems challenges," *In NSF Workshop on Distributed Communications and Signal Processing*, Northwestern University, December 2002.
26. D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," *In Proc. ACM/IEEE MobiCom*, pp. 263-270, 1999.

Appendix A

Definition 1. We say that H is *r-subset-resilient* if, for every probabilistic polynomial-time adversary A,

$$\Pr[(M_1, M_2, ..., M_{r+1}) \leftarrow A(i, 1^t, 1^k) s.t. H_{i,t,k}(M_{r+1}) \subseteq \bigcup_{j=1}^{r} H_{i,t,k}(M_j)] < negl(t,k).$$

Fix a distribution D on the space of all inputs to H (i.e., on the space of messages).

Definition 2. We say that H is *r-target-subset-resilient* if, for every probabilistic polynomial-time adversary A,

$$\Pr[M_1, M_2, ..., M_r \leftarrow D;\ M_{r+1} \leftarrow A(i, 1^t, 1^k, M_1, ..., M_r) \quad s.t.\ H_{i,t,k}(M_{r+1}) \subseteq \bigcup_{j=1}^{r} H_{i,t,k}(M_j)] < negl(t,k).$$