

An Efficient Broadcast Authentication Scheme in Wireless Sensor Networks

Shang-Ming Chang Shihpyng Shieh Warren W. Lin
{changsm, ssp, warren}@csie.nctu.edu.tw

National Chiao Tung University / University of California, Berkeley*
1001 Ta Hsueh Road
Hsinchu, Taiwan 300

Chih-Ming Hsieh
polaris@iii.org.tw

Institute for Information Industry
11th Floor, 106 Ho-ping East Road, Section 2
Taipei, Taiwan 106

ABSTRACT

A broadcast authentication mechanism is important in wireless sensor networks, assuring receivers of a packet's validity. To provide authentication, some researchers utilize one way key chains and delayed disclosure of keys; however, such an approach requires time synchronization and delayed authentication. Another technique uses one-time signature schemes. Unfortunately, such schemes suffer from large key sizes and a limited number of uses per key. To cope with these problems, we propose an efficient, one-time signature-based broadcast authentication scheme for wireless sensor networks that reduces storage usage and includes a re-keying mechanism.

Keywords

Wireless sensor networks, broadcast, authentication, one time signature, key renewal.

1. INTRODUCTION

A wireless sensor network (WSN) can cheaply monitor an environment for diverse industries, such as healthcare, military, or home [23], [24], [25], [26]. A WSN typically consists of several base stations and thousands of sensor nodes, which are resource limited devices with low processing, energy, and storage capabilities. Distributing data through wireless communication is also bandwidth limited.

Broadcast authentication is a basic and important security mechanism in a WSN because broadcast is a natural communication method in a wireless environment. When base stations want to send commands to thousands of sensor nodes, broadcasting is a much more efficient method than unicasting to

This work is supported in part by the National Science Council (NSC), the Institute for Information Industry (III), the Taiwan Information Security Center at NCTU (TWISC@NCTU), and the Team for Research in Ubiquitous Secure Technology at UC Berkeley (TRUST).

* Shihpyng Shieh is currently a visiting professor of the University of California, Berkeley.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ASLACCS'06, March 21–24, 2006, Taipei, Taiwan.

Copyright 2006 ACM 1-59593-272-0/06/0003...\$5.00.

each node individually.

A message authentication code (MAC) is an authentication tag derived by applying an authentication scheme and 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 modification. Because the sender and its receivers share the same secret key, any one of the receivers can impersonate the sender and forge messages to other receivers. That is, both sender and receivers can sign messages. This problem stems from the symmetric property of MAC.

Therefore, to achieve authenticated broadcasts, it is necessary to establish an asymmetric mechanism in which only the sender can sign messages, and the receivers can only verify messages. However, asymmetric cryptographic mechanisms like RSA digital signatures are significantly more computationally expensive than symmetric ones. It is impractical to use them in a resource limited sensor network. A possible approach is to use efficient symmetric primitives as a tool to design a scheme with asymmetric properties.

1.1 Requirements

In addition to the asymmetric mechanism 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; therefore, the scheme should be able to cope with the loss of packets during transmission.
2. Short authentication latency. Many WSN applications are real time applications, e.g. the tracking of enemy movements on a battlefield. 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 other packets; otherwise, the failure to verify a packet prevents the verification of subsequent packets.
4. Low computation cost. Receivers have limited computation power. Thus, they should only perform a small number of operations to verify a packet.
5. Low communication overhead. Because a WSN is restricted in bandwidth, the number of bytes per packet used for authentication should be small.
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 recovers from any loss of packets, has no authentication latency, can individually authenticate packets, has negligible overhead, and has a computation cost similar to what is found in symmetric cryptographic primitives. In practice, such a perfect scheme is difficult to achieve, and a compromise needs to be found between these requirements.

1.2 Related Work

Previously proposed broadcast authentication schemes roughly divide into two categories by the cryptographic primitives they use. The first type is a signature amortization scheme that utilizes asymmetric primitives and distributes the cost of the signature over a block of packets. The second type is a MAC-based scheme which uses symmetric primitives to design an elaborate scheme that achieves asymmetry.

EMSS [20], hash trees [22], hash chains [11], and expander graphs [21] are examples of proposed signature amortization schemes whose main impediment is packet loss. Recently, some researchers [1], [4], [5], [14] suggest using erasure codes [15], [16], [17] to deal with packet loss. Unfortunately, these schemes suffer pollution attacks [5], a type of Denial-of-Service attack aimed at the erasure decoding procedure. Distillation codes [5] solve this problem at the expense of higher communication overhead per packet. Furthermore, schemes employing distillation codes require receivers to buffer the packets before verifying signatures. As a consequence, receivers cannot authenticate each packet individually and require larger storage space. Because of this buffering problem, we will not use a signature amortization scheme. Instead, we focus on using an efficient symmetric cryptographic primitive to achieve asymmetry.

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

BiBa [2] and HORS [3] are one-time signature schemes using one way functions. They are more efficient signature schemes than public key signature schemes and can compare favorably with symmetric primitives because the main computations consist of fast, one-way hash function evaluations. One-time signature schemes have some drawbacks, though, such as the limited number of signatures that one key pair can generate and the large size of the public key. Our proposed scheme improves upon the problem of large storage usage.

In the next section, we describe the system architecture and examine various cryptographic tools that our scheme utilizes. We detail our proposed scheme in section 3. In section 4, we provide a security analysis of our scheme and also compare it against other authentication schemes for WSNs. Finally, we conclude our findings in section 5.

2. PRELIMINARIES

In this section, we first outline the system architecture. We then review some cryptographic primitives used for authentication and a one-time signature called HORS, which is thus 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

A WSN may contain one or more base stations and hundreds or thousands of sensor nodes. Figure 2-1 is an example of a WSN. Compared to a base station, a sensor node is very limited in resources. For simplification, we assume that each broadcast message originates from the base station. A sensor node can broadcast messages by first unicasting the message to the base station, which then broadcasts the messages on the sensor node's behalf. In addition, messages transmitted in a sensor network may reach the destination directly or may be forwarded by some intermediate nodes; however, we do not distinguish between them in our scheme. Furthermore, we assume a base station shares a pairwise secret key with each sensor node, allowing each sensor node to securely receive the base station's public key.

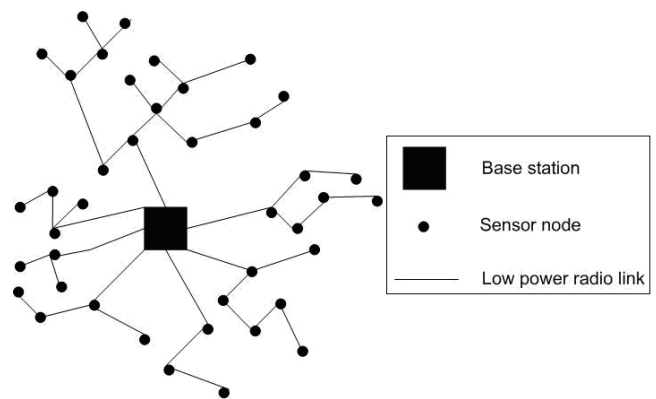


Figure 2-1. Architecture of a WSN

Table 2-1. SmartDust characteristics

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

Sensor nodes are resource limited devices. Our sensor node capabilities are modeled after SmartDust, a Berkeley proposed sensor node prototype whose characteristics are shown in Table 2-1.

2.2 Cryptographic Primitives

In this section, we introduce various cryptographic primitives and schemes for authentication.

2.2.1 Message Authentication Code

A message authentication code (MAC) is a symmetric cryptographic mechanism that takes as input a k -bit secret key and a message, and outputs an l -bit authentication tag. To exchange authentic messages, a sender and receiver must share the same secret key. Using the secret key, the sender computes the message's authentication tag (or MAC) and appends it to the message. To verify the authenticity of a message, the receiver computes the message's MAC with the secret key and compares it to the original MAC appended with the message.

For any message, a secure MAC function prevents an attacker without prior knowledge of the secret key from computing the correct MAC. A MAC achieves authenticity for point-to-point communications because a receiver knows that a message with the correct MAC must have been generated either by itself or by the sender.

2.2.2 Collision Resilient Hash Function

A collision-resilient hash function H [10] is a function that maps an arbitrary length message M to a fixed length message digest MD and exhibits the following properties. (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 computationally infeasible to find an x such that $H(x) = y$. (4) It is computationally infeasible to find two distinct messages (M, M') that hash to the same result $H(M) = H(M')$.

The collision-resilient hash function is very efficient. It only costs a few microseconds to compute on a Pentium III 800 Hz PC, which is a thousand times cheaper than asymmetric primitives. We suggest using a collision-resilient hash function, like SHA-1 or RIPEMD-160, to construct our signature scheme.

2.2.3 Message Authentication Code

A Merkle hash tree [7], [8], [9] is a mechanism for calculating a message digest over a group of data items. Figure 2-2 depicts a Merkle hash tree. It is constructed from a binary tree by using the hash of each data item as a leaf in the tree. Each internal node is then computed by taking the hash of the concatenation of its two children. Let H be a collision resilient hash function. Then, $parent = H(child_{left} | child_{right})$.

A Merkle hash tree can reduce the authentication overhead needed for a large group of data items. For example, a sender signs the root of the tree instead of individual data items. The receiver can then verify the authenticity of every data item by reconstructing the tree and comparing the computed hash value of the tree, which we call *treehash*, with the authenticated root value.

To reconstruct the tree, the receiver needs all of the data items. An alternative is for the receiver to verify a data item individually by computing the treehash using the data item and its authentication path. Illustrated in Figure 2-3, the *authentication*

path of the leaf is the value of all nodes that are siblings of nodes on the path between the leaf and the root.

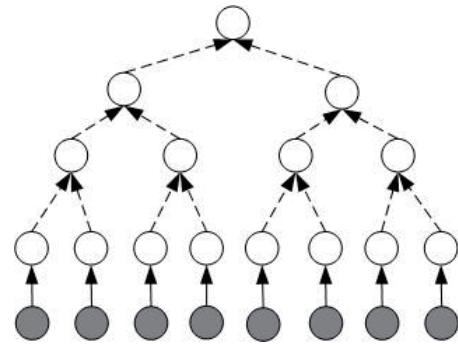


Figure 2-2. Merkle hash tree

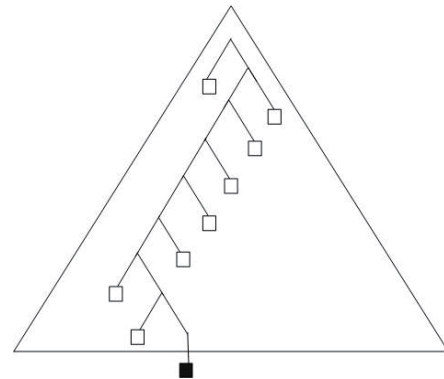


Figure 2-3. Authentication path of Merkle hash tree

2.2.4 Efficiency of Cryptographic Primitives

As exhibited in Table 2-2, symmetric cryptographic primitives like DES and MD5 are much more efficient than asymmetric primitives like RSA. The one way hash function is almost as efficient as a symmetric cipher. All experiments are performed on an 8-byte size input, using the OpenSSL libraries on an 800 MHz Pentium III Linux station.

2.3 One-time Signatures

First proposed by Lamport [6] and Rabin [13], one-time signature schemes are efficient signature schemes based on one way functions. These signature schemes differ from public key signature schemes by the number of messages each can sign. Using a single key pair, the former can only sign several messages, while the latter can sign an unlimited number of messages. This is due to the disclosure of the private key of a one-time signature scheme shortly after signing a few messages. Public key signature schemes never disclose a private key.

Table 2-2. Efficiency of cryptographic primitives

Algorithm	Time per operation	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	0.5 μ s	1.9*10 ⁶
RC5-32-12	0.2 μ s	4.9*10 ⁶
Rijndael 128	0.7 μ s	1.5*10 ⁶
MD5	1.0 μ s	1.0*10 ⁶
SHA-1	1.5 μ s	0.66*10 ⁶
HMAC-MD5	2.5 μ s	0.40*10 ⁶

Despite this limitation, one-time signature schemes are advantageous because of their speed. Since they are based on one way functions, their computation cost is quite low when compared with that of asymmetric primitives.

In general, one-time signature schemes use the secret key as input to a sequence of one-way functions which generate a number of intermediate results that eventually lead to the public key. The one-way property of the function implies that it is infeasible to compute the secret key, or any intermediate result, from the public key. The private key is a self-authenticating value. Motivated by the application of signatures to stream and broadcast authentication, Perrig proposed a one-time signature called BiBa [2], which features fast verification and a short signature. The disadvantage of BiBa is a longer signing time and larger public key size than previous schemes.

2.3.1 Subsubsections

Reyzin and Reyzin proposed a new one-time signature scheme called HORS [3], which improves upon the BiBa scheme by decreasing the time needed to sign and verify messages while also reducing the key and signature sizes, making HORS the fastest one-time signature scheme available thus far. The security of BiBa depends upon 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.

HORS is computationally efficient, requiring a single hash evaluation to generate the signature and a few hash evaluations for verification. Due to the large public key size (usually 10K bytes), HORS is not suitable in a WSN environment without additional modifications. Figure 2-4 outlines the HORS algorithm while Figure 2-5 depicts its architecture.

In a typical example of HORS, we take parameters $l=80$, $t=1024$, and $ft=160$. The private key size is $1024 * 80 \text{ bits} = 10\text{K}$ bytes, and the public key size is $1024 * 160 \text{ bits} = 20\text{K}$ bytes. Since we assume the base station is the sender, the size of the private key is manageable; however, the size of the public key is too large for a sensor node.

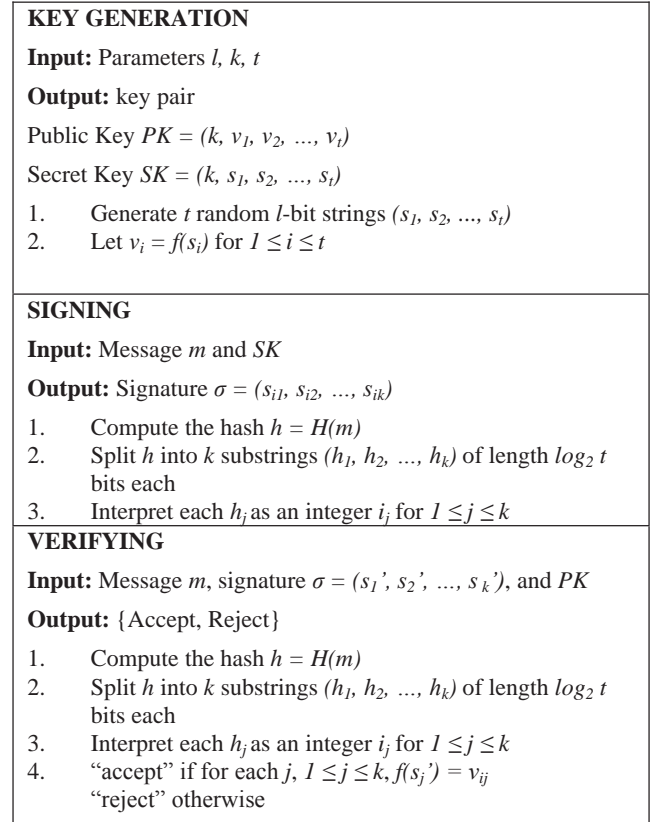


Figure 2-4. HORS algorithm

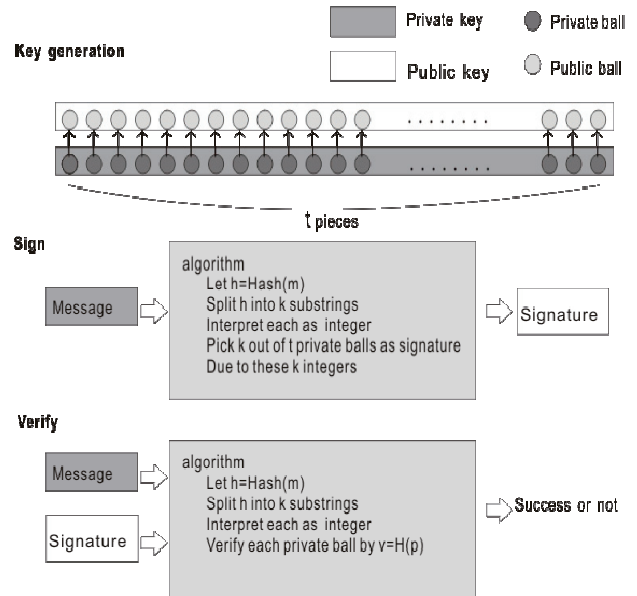


Figure 2-5. HORS architecture

3. Proposed Scheme

We propose a computationally lightweight one-time signature scheme that allows sensor nodes to authenticate broadcast messages from a base station in a wireless sensor network. To attain the asymmetric property necessary for broadcast

authentication, we utilize symmetric cryptographic primitives. Moreover, we mitigate the general drawbacks of one-time signature schemes: the use of an extremely large key size and the limitation to authenticate only a few messages. The proposed scheme efficiently reduces the storage requirement and includes a re-keying mechanism to sign additional messages.

3.1 Signature Scheme

Compared to HORS, our scheme consumes less storage and communication overhead at the expense of a higher computation cost. This is a worthwhile tradeoff because storage is a more precious resource than computation power in a sensor node, especially since our scheme only uses a few additional hash computations.

In the remainder of this section, we first explain the basic idea of our scheme, followed by our generalized scheme. Finally, we propose a re-keying mechanism for our scheme.

3.1.1 The Basic Idea

First, the signer must 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 random numbers *private balls* and their hash values *public balls*.

A verifier can efficiently authenticate the private balls based on the public balls but cannot feasibly compute a valid private ball given a public ball. HORS uses the one-way hash function F as a commitment scheme. Given a set of private balls, the public ball is $p_i = F(r_i)$. The verifier can easily authenticate r_i by verifying $p_i = F(r_i)$.

As previously mentioned, the public key is composed of t public balls. To reduce the size of the public key, we can either reduce the ball size or the number of balls. Because the length of public ball is related to the security strength of the hash function, we cannot reduce the ball size. Thus, we reduce the number of public balls needed by using a Merkle hash tree instead of one-way hash function to authenticate the private balls, as illustrated in Figure 3-1. We place the original private 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 becomes the new public key.

A change in the public key generation also effects signature generation and verification. The signer generates the signature as before, by picking k private balls out of t private balls. However, the signer must affix additional public balls to the signature. The additional public balls correspond to the authentication path of each picked ball. Using the authentication path, the verifier can validate each picked ball by reconstructing the path from picked ball to the root of the Merkle hash tree.

One security flaw occurs when an attacker replaces a disclosed private ball i with private ball j . We cannot distinguish between two private balls in the same tree. To resolve this problem, we use the uniqueness of each leaf's authentication path. For each private ball, we concatenate the public balls along its authentication path. Then, we obtain the identity of the private ball by applying the hash function to this concatenated value. When validating a signature, the verifier first checks the identity of each private ball.

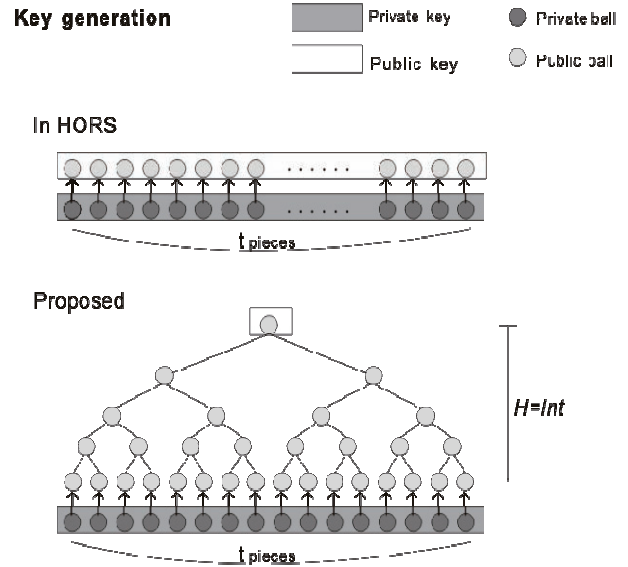


Figure 3-1. Key generation procedure

3.1.2 The Generalized Scheme

We generalize our scheme by first constructing many small Merkle hash trees of height h that hold 2^h private balls. The public key contains the root nodes of all the Merkle hash trees, and hence reduces the key size by a factor of 2^h . However, to authenticate each private ball, the signer adds the authentication path of each private ball, which requires h verification nodes. Thus, the signature size increases by a factor of h .

By constructing many Merkle trees, we can lower the overall storage requirement. As public key size decreases, the signature size increases. If we only build one Merkle tree, the combined size of the public key and signature would not be minimal. Therefore, we need to find an optimal balance between the public key size and signature size.

We use the parameters in Table 3-1 when describing our scheme.

Table 3-1. System Parameters

t	number of private balls
k	number of signature balls
d	number of public balls
l	size of ball (bits)
r	r -subset resilient

Our proposed scheme comprises of three phases: initial phase, signing phase, and verification phase. In the initial phase, the sender generates a private key and its corresponding public key, as shown in Figure 3-2. A pseudorandom generator produces a private key, which is made of t l -bit random numbers. As outlined in Figure 3-3, the PUBLIC_KEY_GENERATION algorithm derives the public key, which consists of d hash values. Next, the

sender uses the private key in the signing phase to sign a message, as Figure 3-4 shows. Finally, the receivers use the sender's public key in the verification phase to validate the signature of the message, as illustrated in Figure 3-6. We elaborate upon each phase in the remainder of this section.

3.1.2.1 Key Generation

A key pair consists of a private key and a public key. The private key is composed of t l -bit random numbers created by a pseudorandom generator, and the public key is derived from these t random numbers. First, we input t random numbers into the one-way hash function and output t hash values. Then, we separate t hash values into d groups so that there are t/d values in each group. Finally, we use these t/d values as the leaves of the binary tree and compute each intermediate node as the hash of the concatenation of the two child values. Thus, we attain d Merkle trees, whose roots comprise 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 roots. In a typical case, $t = 1024$ and $d = 32$.

KEY_GENERATION

Input: parameters t, k, d, l

Output: key pair

Private Key $K_{pri} = (k, s_1, s_2, \dots, s_t)$

Public Key $K_{pub} = (k, v_1, v_2, \dots, v_{ln d})$

1. Randomly generate t l -bit random numbers (s_1, s_2, \dots, s_t) as private key
2. $K_{pub} = PUBLIC_KEY_GENERATION(k, d, K_{pri})$
3. Distribute public key

Figure 3-2. Key generation algorithm

PUBLIC_KEY_GENERATION

Input: parameters k, d , and K_{pri}

Output: K_{pub}

1. Use t balls as pre-image of leaves to build c Merkle trees with height $ln t$.
2. Take $ln d$ tree root as public key, with each public key corresponding to a sequence period.

Figure 3-3. Public key generation algorithm

3.1.2.2 Broadcasting Authenticated Messages

A sender must sign all messages it broadcasts. To sign a message m , we first compute $h = H(m)$. Then, we separate the hash value h into k pieces and regard these pieces as integers, ending up with (i_1, i_2, \dots, i_k) between 0 and $t-1$. Next, we combine these integers to form a subset of $\{0, 1, 2, \dots, t-1\}$ of size at most k . Each integer is an index of private balls (r_1, r_2, \dots, r_t) . We pick k private balls $(r_{i_1}, r_{i_2}, \dots, r_{i_k})$ and use them, along with their associated authentication paths, as the signature of this message m . Compared to the original HORS, these authentication paths are added communication overhead. For better performance, the duplicate

path is sent only once. We discuss the duplicate authentication path below.

SIGNATURE_GENERATION

Input: message m and K_{pri}

Output: signature $\sigma = \{a_{i_1}, a_{i_2}, \dots, a_{i_k}, bs\}$, where $a_i = (s_i, ap_i)$ (ap is the authentication path of the ball)

1. Compute the hash $h = H(m)$
2. Split h into k pieces (h_1, h_2, \dots, h_k) of length $ln t$ bits each
3. Interpret each h_j as an integer i_j , with $1 \leq j \leq k$

Figure 3-4. Signature generation algorithm

3.1.3 The Duplicated Authentication Path

To allow a receiver to verify each private ball, the sender must include in the signature nodes that comprise the authentication path. Unfortunately, the sender may inadvertently send these additional nodes multiple times. For example, consider Figure 3-5. A sender first transmits ball s_0 and its authentication path $\{v_1, m_{23}, m_{47}\}$. It then sends s_1 and its authentication path $\{v_0, m_{23}, m_{47}\}$. The sender should send nodes m_{23} and m_{47} once since they are duplicates. Moreover, a receiver can compute $v_0 = H(s_0)$. Thus, if a sender selects a direct neighbor of a disclosed private ball, then no additional nodes are required to be sent. In general, if a node at height e is a common parent, then all nodes higher than e need not be resent. Therefore, selected private balls that are close together can reduce transmission overhead. We note that the upper bound of the sum of the authentication paths is $min(r*k*h, \text{the whole tree})$.

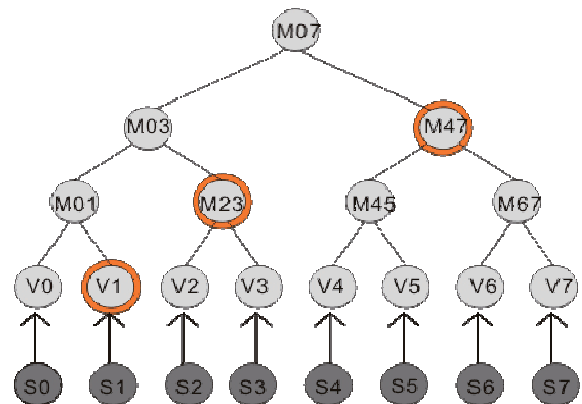


Figure 3-5. Duplicate authentication path

3.1.3.1 Authenticating Broadcast Messages

When a receiver obtains a broadcast message, it must ensure that the message originated from an authentic sender by verifying the signature of message m with the following procedure. Since the sender will periodically re-key, the receiver must first decide which of the sender's public keys should be used to verify the message's signature. Second, the receiver checks which sequence period of that public key the sequence number falls into. Third, it computes the hash value $h = H(m)$. Fourth, the receiver separates the hash value h into k pieces and regards these pieces as integers

(i_1, i_2, \dots, i_k) between 0 and $t-1$, with each integer as an index to a private ball (r_1, r_2, \dots, r_t) . Next, the receiver checks the identities of the balls by uniqueness of authentication path as discussed at the end of section 3.1.1. Finally, the receiver verifies each private ball by computing the treeshash of the private ball with its authentication path and checking whether this treeshash is equivalent to the public key. A complete match assures the receiver that the private ball belongs to the authenticated sender. The verification algorithm is shown as Figure 3-6.

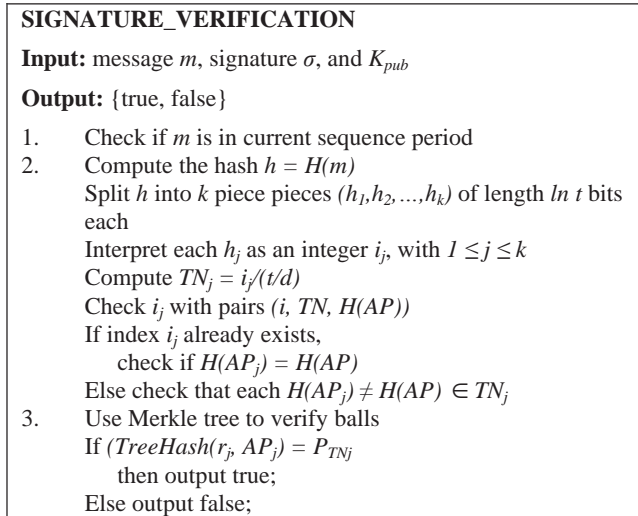


Figure 3-6. Signature verification algorithm

3.2 Re-keying Mechanism

Because a single key pair can only sign r messages, a sender should use a new key pair when signing more than r messages. Therefore, we offer a re-keying scheme as a solution.

If one key pair can sign r messages, we set the duration of a sequence period to r . The sequence numbers of the first public key ranges from 0 to $r-1$, the sequence numbers of the second public key ranges from r to $2r-2$, and so on, as depicted in Figure 3-7. Each key pair can only be used for the duration of its sequence numbers. A receiver must first check which sequence period a received message belongs to.

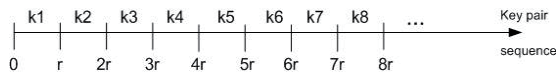


Figure 3-7. Sequence period of a public key

Since a base station shares a pair-wise secret key with each sensor node, the base station unicasts the first public key to each sensor node through the authenticated channel. For efficiency, the base station may distribute the next public key by authenticated broadcast using the old private key.

4. DISCUSSION

4.1 Security and Performance Analysis

In this section, we discuss the system parameters that influence security strength and performance of our scheme. We define these parameters in Table 4-1. Higher security strength typically results in lower performance and vice versa; therefore, we aim to provide just enough security strength. Thus, we suggest values of $t = 1024$, $k = 16$, and $r = 4$, which affords 64 bits of security against a chosen message attack. For brute force attacks, we follow HORS recommendation of $l = 80$.

Table 4-1. System parameters

h	cost of computing a hash function
l	size of a private ball
t	number of private balls
k	number of balls in the signature
h_i	size of private ball's identity
f_i	size of a public ball
d	number of public balls (or trees)
r	number of messages one key pair can sign

Theorem 1 describes the security strength of our scheme against brute force and chosen message attacks. Given a fixed amount of leaves, we can construct several trees instead of just one tree. A deeper tree produces a smaller public key size; however, the sender must transmit more public balls per signature. Since the receiver concurrently stores the public key and the signature, we wish to find an optimal balance between them. Theorem 2 determines the number of trees that will generate the smallest storage size. Although we may decrease the depth by increasing the degree of a Merkle tree, theorem 3 will show that a two degree Merkle tree is optimal.

Theorem 1: The parameter l decides the security strength against a brute-force attack, while the parameters k , t , and r determine the security strength against a chosen message attack by providing

$$k(\log t - \log k - \log r)$$

bits of security.

Proof: Let f be a one-way function operating on an l -bit input string. Then the output of the one-way function has length f_i . To perform a brute force attack against f_i , attackers derive the private ball r_i from the Merkle tree's leaves using

$$leaf_i = f(r_i)$$

Because the sender discloses the private balls with the signatures, attackers may execute a chosen message attack by collecting the private balls to forge a signature. We assume that the attacker can obtain signatures on r messages of its choice (independent of the hash H). The attacker then tries to forge a signature on 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 trivial to see that, for each invocation of H , this probability is at most

$$(rk/t)^k.$$

Lemma 1: The computation cost, communication overhead, and storage requirement are defined below.

Computation cost (of receiver):

$$k * h$$

Communication overhead:

$$d * f_l + r * (k * l + f_l * k * h)$$

Storage requirement (of receiver):

$$d * f_l + k * l + f_l * k * h + r * k * h_l$$

Proof: To verify a signature, the receiver must compute the root using the leaf and its authentication path, thus, k hash calculations are needed. The communication overhead requires the public key size, which consists of d roots, and r signature size, which each consists of k private balls and k authentication paths. The storage requirement includes the public key size, the signature size, and the private ball's identity size.

Lemma 2: The tree height is related to public key size by

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

Proof: The height of the tree depends on the number of private balls and the number of public tree roots. The number of leaves of a tree is t/d . The height of the binary tree with t/d leaves is

$$\ln(t/d) = \ln t - \ln d.$$

Theorem 2: For a given security strength, we select a value for the number of trees d that will minimize the equation

$$\min(d * f_l + k * l + f_l * k * (\ln t - \ln d)).$$

Proof: The lowest storage requirement determines the number of trees we must construct. Combining lemma 1 and lemma 2, we can find the optimal number of trees to construct.

Theorem 3: A two 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

$$(d-1) * h = (d-1) * \log_d t$$

For a fixed number of leaves t , we find that the equation

$$\min((d-1) * \log_d t)$$

is at a minimum when the degree is two.

4.2 Case Study

A municipality wishes to collect traffic information from sensors distributed in the streets. The sensors need to authenticate commands from the base station and return sensed data through a secure channel.

The WSN possesses the following characteristics. With a data rate of 10 Kbps, roughly 20 packets of 64 bytes each are sent every second. The packet drop rate is at most 5 percent, and the average length of burst drops is 5 packets. The verification latency should be less than 10 seconds.

According to theorem 1, a system with parameters of $l=80$, $t=1024$, $k=16$, $r=4$, $f_l=160$ yields 64-bit security. Thus, an attacker needs to perform 264 hash computations during a key pair's lifetime to forge a signature. BiBa [2] provides 58-bit security for a real-time stock quotes application; therefore, we consider 64-bit security as sufficient for our application. Using theorem 2, we calculate an optimal public key size of 640 bytes, which corresponds to 32 tree roots. Figure 4-1 depicts the optimal public key size.

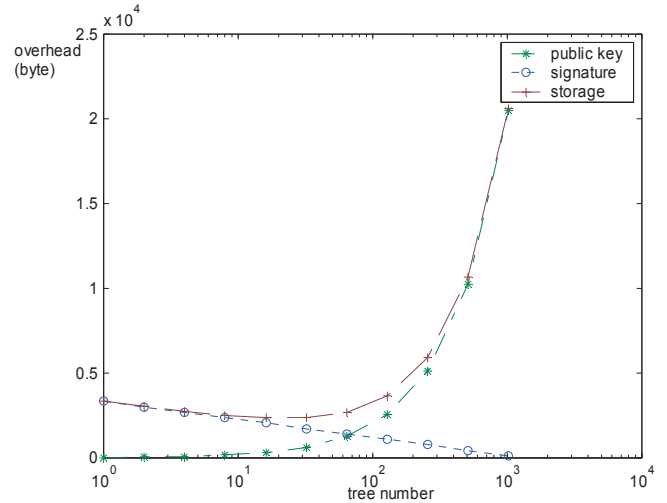


Figure 4-1. Optimal public key size

After deciding upon the optimal number of trees, we generate the key pair using the key generation algorithm of Figure 3-2. Figure 4-2 illustrates the tree construction for this example.

Private Key: 1024 80-bit random numbers

Public Key: 32 160-bit hash values

Number of trees: 32

Number of leaves: 32 per tree

Tree height: 5

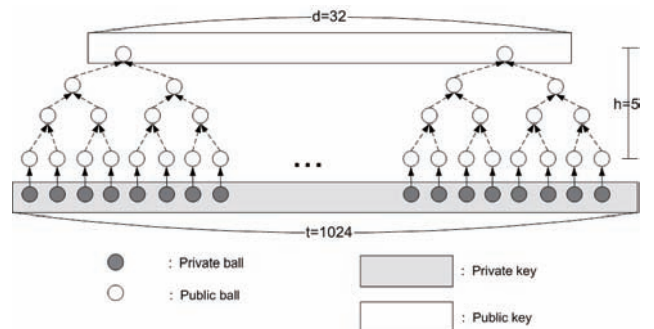


Figure 4-2. Key pair configuration

4.3 Comparison

We compare our proposed scheme with μ TESLA [18], an efficient broadcast authentication protocol for WSNs. Our scheme has four main advantages over μ TESLA, which we list below.

1. **No time synchronization requirement.** In μ TESLA, the sender and receivers utilize time synchronization and delayed disclosure to achieve asymmetry needed for broadcast authentication, while our scheme uses public-private key pairs. Thus, our scheme has no time synchronization requirement, which may be impractical in a large WSN.
2. **No buffering needed by receiver.** μ TESLA requires receivers to buffer a received packet until the corresponding MAC key is disclosed. Receivers need not buffer received messages in our scheme.
3. **Individual authentication of message.** Receivers can individually authenticate a received packet without waiting for another packet. In μ TESLA, receivers must wait for the packet containing the disclosed MAC key.
4. **Instant authentication of message.** With our scheme, receivers can instantly authenticate a received packet. In contrast, μ TESLA requires a receiver to wait one or more time periods for the packet containing the disclosed MAC key.

We also compare our scheme against two other efficient one-time signature schemes, BiBa [2] and HORS [3]. Testing at the same security level with system parameters of $t=1024$, $k=16$, $r=10$, $h=5$, our proposed scheme outperformed BiBa and HORS in three areas: computation, communication, and storage overhead. Table 4.1 summarizes the results of our experiment.

Table 4-2. Signature scheme comparison

	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	Small
Time synchronization	Yes	No	No

5. CONCLUSION

In this paper, we propose an efficient broadcast authentication scheme for wireless sensor networks that utilizes a one-time signature scheme and re-keying mechanism. Our scheme exhibits many nice properties including individual authentication, instant authentication, robustness to packet loss, and low overhead in computation, communication and storage. We improve upon the HORS one-time signature scheme, reducing the large key storage requirement of HORS by using Merkle hash trees in generating the key pair. Despite increasing computation and communication overhead, we significantly decrease the storage space usage. We

also devise a simple but efficient re-keying mechanism for our scheme.

6. REFERENCES

- [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 Hawaii 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.

7. APPENDIX

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

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

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 ,

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