

Efficient One-time Signature Schemes for Stream Authentication*

YONGSU PARK AND YOONKUN CHO[†]

College of Information and Communications

Hanyang University

Seoul, 133-791 Korea

E-mail: yspark@ssrnet.snu.ac.kr

[†]*School of Computer Science and Engineering*

Seoul National University

Seoul, 151-742 Korea

E-mail: cho@ssrnet.snu.ac.kr

When one-time signatures are used for stream authentication, one of the most serious drawbacks is that their large signature size yields high communication overhead. In this paper, we present two efficient one-time signature schemes for stream authentication. Compared with the previous schemes, these schemes have the smallest signature sizes. Moreover, their verification overheads are low. The signature size of Scheme 1 is smaller than that of Scheme 2 whereas Scheme 2 has much smaller signing cost: it requires only 2 hash operations in the majority of cases. Although Scheme 1's signing cost is relatively high, it can be parallelized without any additional risk because sharing the private key among distributed servers is not required.

Keywords: information theory, security, cryptography, authentication, digital signature, stream distribution

1. INTRODUCTION

Recently the Internet is widely used for distributing streamed data. To enable widespread commercial broadcast services, it is important to provide data integrity and source authentication [11, 13]. For example, a listener may feel the need to be assured that stock quotes or news streams have not been altered and were made by the original broadcast station.

One-time digital signature schemes are digital signature mechanisms that sign the message at most once. Because one-time signature schemes have much lower signing cost than ordinary signature schemes, they can be used for signing each live stream chunk in order to support fast packet rates [9, 12]. One of the serious drawbacks in using these schemes, however, is that their large signature size yields high communication overhead. For example, if we use SHA-1 [6], the size of Lamport signature [6] is 3200 bytes, which is much larger than the size of a single stream chunk that usually does not exceed 512 bytes¹ [11, 13].

Received May 7, 2004; revised April 26, 2005; accepted November 2, 2005.

Communicated by Shiuhyng Shieh.

* A preliminary version of this paper has appeared on the technical/industrial track of ACNS 2004.

¹ In the Internet, streamed media is transmitted in the unit of IP packet. If a stream chunk (or its signature) is partially transmitted to the receivers by IP packet loss, it will be unverifiable. The larger the size of each chunk, the more frequently partial transmission occurs and the more chunks will be unverifiable. Hence, the size of a chunk should be much smaller than that of IP packet which is usually 4096 bytes.

In [15] Rohatgi proposed a method for efficient use of k -time signature to reduce communication overhead. Recently, new one-time signature schemes that have small signature sizes have been proposed [9, 12, 14]. Although Powerball [9] is claimed to have the smallest signature size among them, its on-line signing cost is very high. HORS (Hash to Obtain Random Subset) [14] requires only a single hash operation for signature generation but its signature size is larger than that of BiBa [12] or Powerball.

In this paper, we propose two efficient one-time signature schemes for stream authentication. The proposed schemes can be viewed as the improvements of HORS by using double hash preimages and by inserting a restricted condition on signature generation/verification.

Compared with [9, 12, 14], under the same security level, our schemes have the smallest signature sizes. Moreover, their verification costs are low. The signature size of Scheme 1 is smaller than that of Scheme 2 whereas Scheme 2 has much smaller signing cost: it requires only 2 hash operations in the majority of cases. Although Scheme 1's signing cost is relatively high, it can be parallelized without any additional risk because sharing the private key among distributed servers is not required.

This paper is organized as follows. In section 2 we give some preliminaries and definitions to understand our schemes. In section 3 we describe related work and in section 4 we propose our schemes. In section 5, we analyze the computation cost of our schemes, perform security analysis and show the comparison results between the previous schemes and the proposed schemes. Finally, conclusions are made in section 6.

2. PRELIMINARIES

In this section we briefly describe definition of one-time/ k -time signatures and then define some notations. One-time digital signature schemes are digital signature mechanisms that sign the message at most once; if we use it more than once, the signatures can be forged [6]. In k -time digital signature schemes, we can use the signing operation at most k times.

We use the following notations in the rest of the paper. Let $f: \{0, 1\}^l \rightarrow \{0, 1\}^l$ be a one-way function. Let $h: \{0, 1\}^* \rightarrow \{0, 1\}^{k \log_2 t}$ and $g_h: \{0, 1\}^* \rightarrow \{0, 1\}^{k \log_2 n}$ be one-way hash functions (l, k, t and n are security parameters). Note that $f()$, $h()$ and $g_h()$ can be implemented by using standard hash functions (SHA-1 or RIPEMD-160) [14]. $A \parallel B$ denotes the concatenation of strings A and B . If an integer f is divisible by an integer e , we write $e \mid f$. $|C|$ denotes the bit size of string C . The signer and the verifier are denoted as S and V , respectively.

3. RELATED WORK

Research on broadcast/multicast stream authentication can be classified into two categories: first, research for designing faster signature schemes and second, research for amortizing each signing operation by making use of a single signature to authenticate several packets [5]. For lack of space, we explain the previous work of only the first approach that is related to this paper (for the latter approach, refer to [4, 5, 8, 10, 11, 13, 16, 17]).

In [17], Wong and Lam proposed some methods for speeding up FFS (Feige-Fiat-Shamir) signature scheme [6] by using CRT (Chinese Remainder Theorem) [6], reducing the verification key size and using precomputation with large memory. They showed that the verification in the scheme was as fast as that of RSA with a small exponent and the signing operation was much faster than those of other schemes (RSA, DSA, ElGamal and Rabin). Moreover, they extended FFS to allow “adjustable and incremental” verification, in which V can verify the signature at different levels: V can verify it at a lower security level with a small computation cost, and later increase the security level with longer computation time.

In [4], Gennaro and Rohatgi proposed an efficient broadcast authentication scheme by using one-time signatures. Compared with the ordinary signature schemes, one-time (or k -time) signature scheme shows much faster sign/verification rates. However, the size of one-time (or k -time) signature is much larger than those of RSA or ElGamal, e.g., the size of Lamport one-time signature of the SHA-1 hashed message is 3200bytes. In [15], Rohatgi used TCR (Target Collision Resistant) function to reduce the signature size and the public key size of k -time signature. Another method used in [15] is the optimization of the size of the certificate, which further reduced communication overhead. If these methods are combined, size overhead per stream chunk is about 300bytes when on-line scheme of [4] is used.

Recently, Perrig *et al.* devised an efficient one-time signature scheme for broadcast authentication, BiBa (Bins and Balls) [12]. BiBa uses the Birthday Paradox as follows. Assume that there exists a family of hash functions $G = \{g_0, \dots, g_{n-1}\}$. First, S generates random secret values s_1, \dots, s_t that are called balls. The public key consists of the outputs of a one-way function that takes each ball as an input: $v_i = f(s_i)$ ($1 \leq i \leq t$). Given a message m to be signed, S computes $h = h(m)$ and selects g_h from G . Then, by using g_h , each ball is mapped to one of n bins as follows: s_i is related to the bin having index $g_h(s_i)$. If there exists a bin that contains k balls, these k balls are the signature of m . The signature size of BiBa is much smaller and verification cost is much lower than those of the previous one-time signature schemes. However, the signing cost is high and the public key size is large. In [12], Perrig presented efficient broadcast authentication methods that permit a large public key size of one-time signatures.

Mitzenmacher and Perrig proposed the Powerball signature scheme [9], which is an improvement of BiBa. First, the key generation in Powerball is as follows. For input parameters n , k and t , S generates t random $l (= k \log n)$ -bit strings s_1, \dots, s_t and then computes $p_i = f(s_i)$, $v_i = f(p_i)$ for $1 \leq i \leq t$. The private key and public key are $SK = (s_1, \dots, s_t)$ and $PK = (v_1, \dots, v_t)$, respectively. Second, the signature generation is as follows. Given a message m and SK , S computes $h = h(m \parallel c)$ to select g_h from the family of hash functions G , where c is a counter that S increments if he is unable to find a signature. Each ball p_i ($1 \leq i \leq t$) is mapped to one of n bins that has index $g_h(p_i)$. Then, each s_i ($1 \leq i \leq t$) is interpreted as a sequence of k bins, $b_{w_1} \parallel \dots \parallel b_{w_j} \parallel \dots \parallel b_{w_k}$ ($0 \leq b_{w_j} < n$). If S finds s_i such that each corresponding bins b_{w_j} ($1 \leq j \leq k$) is mapped to at least one ball p_i , the signature of m is $SIG = (p_{i_1}, \dots, p_{i_k}, s_i, c)$. Third, the signature verification is as follows. Given m , $SIG = (p'_1, \dots, p'_k, s', c')$ and PK , V computes $h = h(m \parallel c')$ and selects g_h from G . If $f(p'_i)$ ($1 \leq i \leq k$) and $f(f(s'))$ is in PK and all the corresponding k bins for s' contain at least one ball, V accepts the signature. As will be shown in section 5.4, even though key

generation cost of Powerball is twice larger than that of BiBa, the signature size and verification cost are smaller than those of BiBa.

Recently, Reyzin and Reyzin proposed an efficient one-time signature scheme, HORS [14]. First, the key generation in HORS is as follows. For input parameters l , k and t , S generates t random l -bit strings s_1, \dots, s_t and computes $v_i = f(s_i)$ for $1 \leq i \leq t$. The private key and public key are $SK = (s_1, \dots, s_t)$ and $PK = (v_1, \dots, v_t)$, respectively. Second, the signature generation is as follows. Given a message m and SK , S splits $h(m)$ into h_1, \dots, h_k , each of which is $\log_2 t$ bits long. h_j is interpreted as an integer i_j ($1 \leq i_j \leq t$, $1 \leq j \leq k$). The signature of m is $SIG = (s_{i_1}, \dots, s_{i_k})$. Third, the signature verification is as follows. Given m , $SIG = (s'_{i_1}, \dots, s'_{i_k})$ and PK , V splits $h(m)$ into h'_1, \dots, h'_k . V interprets h'_j as an integer i'_j ($1 \leq j \leq k$). If $f(s'_{i'_j}) = v_{i'_j}$ for all j , V accepts the signature. HORS has very low signing cost since it requires only 1 hash operation. However, the signature size of HORS is larger than those of Powerball and BiBa, which will be shown in section 5.4.

4. THE PROPOSED SCHEME

In HORS, assume that an attacker A has $SIG = (s_{i_1}, \dots, s_{i_k})$ for m . Then, A can forge a signature for another message by changing the positions of elements in SIG , e.g., if A can find m' or m'' ($m', m'' \neq m$) such that $h(m') = h_2 \parallel h_1 \parallel h_3 \parallel \dots \parallel h_k$ or $h(m'') = h_3 \parallel h_2 \parallel h_1 \parallel \dots \parallel h_k$, he can create a signature $(s_{i_2}, s_{i_1}, s_{i_3}, \dots, s_{i_k})$ for m' or $(s_{i_3}, s_{i_2}, s_{i_1}, \dots, s_{i_k})$ for m'' . We improved HORS by minimizing the success probability of this attack. Thus, under the same security level, our schemes have smaller signature sizes than that of HORS. More specifically, we use double hash preimages and insert a restricted condition on signature generation/verification to minimize the above attack. Our schemes have the constraint in that $2 \mid k$.

4.1 Scheme 1

Like HORS, Scheme 1 consists of 3 parts: key generation, signature generation and signature verification. First, key generation is as follows. For input parameters l , k and t , S generates t random l -bit strings s_1, \dots, s_t and then computes $p_i = f(s_i)$, $v_i = f(p_i)$ for $1 \leq i \leq t$. In the proposed scheme, all s_a, p_b, v_c ($1 \leq a, b, c \leq t$) should be different². The private key and public key are $SK = ((s_1, \dots, s_t), (p_1, \dots, p_t))$ and $PK = (v_1, \dots, v_t)$, respectively. Second, signature generation is as follows. Given a message m and SK , S selects a random value c and computes $h(m \parallel c) = h_1 \parallel \dots \parallel h_k$. h_j is interpreted as an integer i_j ($1 \leq j \leq k$). i_j ($1 \leq j \leq k$) must meet the following equations. If not, S repeats the above procedure for another c .

$$i_1 < \dots < i_{k/2}, i_{k/2+1} < \dots < i_k, \{i_1, \dots, i_{k/2}\} \cap \{i_{k/2+1}, \dots, i_k\} = \phi. \quad (1)$$

The signature is $SIG = (c, (s_{i_1}, \dots, s_{i_{k/2}}), (p_{i_{k/2+1}}, \dots, p_{i_k}))$. Third, signature verification is as follows. Given a message m , $SIG = (c', (u_1, \dots, u_{k/2}), (u_{k/2+1}, \dots, u_k))$ and PK , V com-

² s_a, p_b, v_c ($1 \leq a, b, c \leq t$) are l -bits strings, where usually $l \geq 80$. Considering that t is on the order of 1000 (see section 5.4), the probability that all s_a, p_b, v_c are different is extremely high.

computes $h(m \parallel c') = h'_1 \parallel \dots \parallel h'_k$. V interprets h'_j as an integer i'_j ($1 \leq j \leq k$). i'_j ($1 \leq j \leq k$) must meet Eq. (1). If $f(f(u_j)) = v_{i'_j}$ and $f(u_{j+k/2}) = v_{i'_{j+k/2}}$ for $1 \leq j \leq k/2$, V accepts the signature.

Example 1: Assume that $l = 80$, $k = 4$ and $t = 8$. First, S generates $SK = ((s_1, \dots, s_8), (p_1, \dots, p_8))$ where s_i is 80-bit string and $p_i = f(s_i)$ ($1 \leq i \leq 8$). Then, S computes $PK = (v_1, \dots, v_8)$ where $v_i = f(p_i)$ ($1 \leq i \leq 8$). For a message m , S selects a random value c and computes $h(m \parallel c) = h_1 \parallel h_2 \parallel h_3 \parallel h_4$. S interprets each h_j as i_j ($1 \leq j \leq 4$). If $i_1 = 2, i_2 = 7, i_3 = 4, i_4 = 8$ then $i_1 < i_2, i_3 < i_4$, and $\{i_1, i_2\} \cap \{i_3, i_4\} = \emptyset$. The signature of m is $SIG = (c, (s_2, s_7), (p_4, p_8))$. Given m , $SIG = (c', (u_1 = s_2, u_2 = s_7), (u_3 = p_4, u_4 = p_8))$ and PK , V computes $h(m \parallel c') = h'_1 \parallel h'_2 \parallel h'_3 \parallel h'_4$. V interprets each h'_j as i'_j ($1 \leq j \leq 4$). If m and SIG have not been modified, $i'_1 = 2, i'_2 = 7, i'_3 = 4, i'_4 = 8$, which meets Eq. (1). If $f(f(u_1)) = v_2, f(f(u_2)) = v_7, f(u_3) = v_4$ and $f(u_4) = v_8$, V accepts the signature.

Recall that in HORS, an attacker A can forge a signature for another message by changing the positions of elements in $SIG = (s_{i_1}, \dots, s_{i_k})$. In Scheme 1, all such attacks are impossible, which results in significant reduction of the probability of forgery. We will show this in sections 5.3 and 5.4.

Note that we can prevent this forgery attack if we simply modify HORS s.t. in generating a signature all i_1, \dots, i_k should meet the condition, $i_1 < \dots < i_k$. However, this naive method requires a large amount of $h(m)$ operations to find such i_1, \dots, i_k . In our scheme, the signing cost is smaller than that of this naive method because we use the hash function twice for each s_i , divide the signature into two part, and use Eq. (1) that are less restrictive than the condition of the naive method. (For further dividing the signature into multiple parts, see Appendix.)

4.2 Scheme 2

Although Scheme 1 has a small signature, the signing cost is quite large, which will be seen in section 5.4. In this section we present Scheme 2, which has a less restrictive condition in signature generation compared with Eq. (1) of Scheme 1. Thus, the signing cost of Scheme 2 is much smaller than that of Scheme 1.

Scheme 2 also consists of 3 parts: key generation, signature generation and signature verification. First, key generation is as follows. For input parameters $l, k(2|k)$ and t , S generates t random l -bit strings s_1, \dots, s_t and then computes $p_i = f(s_i), v_i = f(p_i)$ for $1 \leq i \leq t$. In the proposed scheme, all s_a, p_b, v_c ($1 \leq a, b, c \leq t$) should be different. The private key and public key are $SK = ((s_1, \dots, s_t), (p_1, \dots, p_t))$ and $PK = (v_1, \dots, v_t)$, respectively.

Second, signature generation is as follows. Given a message m and SK , S selects a random value c and computes $h(m \parallel c) = h_1 \parallel \dots \parallel h_k$. h_j is interpreted as an integer i_j ($1 \leq j \leq k, 1 \leq i_j \leq t$). i_j ($1 \leq j \leq k$) must meet the following condition: i_1, \dots, i_k should be different. If not, S repeats the above procedure for another c . The signature is $SIG = (c, (s_{i_1}, \dots, s_{i_{k/2}}), (p_{i_{k/2+1}}, \dots, p_{i_k}))$.

Third, signature verification is as follows. Given a message m , $SIG = (c', (u_1, \dots, u_{k/2}), (u_{k/2+1}, \dots, u_k))$ and PK , V computes $h(m \parallel c') = h'_1 \parallel \dots \parallel h'_k$. After V interprets h'_j as an integer i'_j ($1 \leq j \leq k, 1 \leq i'_j \leq t$), he verifies that all i'_j are different. If $f(f(u_j)) = v_{i'_j}$ and $f(u_{j+k/2}) = v_{i'_{j+k/2}}$ for $1 \leq j \leq k/2$, V accepts the signature.

Example 2: Assume that $l = 80$, $k = 4$ and $t = 8$. First, S generates $SK = ((s_1, \dots, s_8), (p_1, \dots, p_8))$ where s_i is a 80-bit string and $p_i = f(s_i)$ ($1 \leq i \leq 8$). Then, S computes $PK = (v_1, \dots, v_8)$ where $v_i = f(p_i)$ ($1 \leq i \leq 8$). For a message m , S selects a random value c and computes $h(m \parallel c) = h_1 \parallel h_2 \parallel h_3 \parallel h_4$. S interprets each h_j as i_j ($1 \leq j \leq 4$). If $i_1 = 4$, $i_2 = 2$, $i_3 = 3$, $i_4 = 5$ then it meets the condition that all i_j ($1 \leq j \leq 4$) should be different. The signature of m is $SIG = (c, (s_4, s_2), (p_3, p_5))$. Given m , $SIG = (c', (u_1 = s_4, u_2 = s_2), (u_3 = p_3, u_4 = p_5))$ and PK , V computes $h(m \parallel c') = h'_1 \parallel h'_2 \parallel h'_3 \parallel h'_4$. V interprets each h'_j as i'_j ($1 \leq j \leq 4$). If m and SIG have not been modified, $i'_1 = 4$, $i'_2 = 2$, $i'_3 = 3$, $i'_4 = 5$, which meets the above condition. If $f(f(u_1)) = v_4$, $f(f(u_2)) = v_2$, $f(u_3) = v_3$ and $f(u_4) = v_5$, V accepts the signature.

Recall that in HORS, an attacker A can forge a signature for another message by changing the positions of elements in $SIG = (s_{i_1}, \dots, s_{i_k})$. In Scheme 2, all the valid signatures that A can forge have a form of $SIG' = (c', (a \text{ permutation of } (s_{i_1}, \dots, s_{i_{k/2}})), (a \text{ permutation of } (p_{i_{k/2+1}}, \dots, p_{i_k})))$, which results in significant reduction of the probability of forgery. We will show this in sections 5.3 and 5.4.

5. ANALYSIS

We first calculate the computation cost of the proposed schemes in section 5.1. In section 5.2, we analyze the private key size, the public key size and the signature size. In section 5.3, we perform security analysis of our schemes. Finally, we explain the comparison results between the previous schemes and our schemes in section 5.4.

In this section, we assume that $f()$ and $h()$ are modelled as a random oracle [1]. This simplifies analysis on security of the proposed schemes. Moreover, under this assumption, the output of $h()$ shows a uniform distribution, which simplifies analysis on the computation cost of our schemes. This assumption is from the convention of the previous schemes [9, 12, 14].

5.1 Computation Cost

We analyze the computation cost of our schemes in terms of the number of one-way (hash) functions computed. In Scheme 1, key generation requires $2t$ evaluations of $f()$. For signature verification, V must compute $h()$ once and $f()$ $3k/2$ times. For signature generation, S evaluates $h()$ $\frac{t^k (\frac{k}{2})^2 (t-k)!}{t!}$ times on average by the following theorem. Note that the output of $h()$ shows a uniform distribution over $[0, 2^{k \log_2 t} - 1]$ since we assume that $h()$ is modelled as a random oracle [1]. Therefore, i_1, \dots, i_k have uniform distributions over $[1, t]$.

Theorem 1 Assume that i_1, \dots, i_k are random variables that have uniform distributions over $[1, t]$. In Scheme 1, the probability that Eq. (1) hold is $\frac{t!}{t^k (\frac{k}{2})^2 (t-k)!}$.

Proof: First the probability that $i_1 < \dots < i_{k/2}$ is $\binom{t}{\frac{k}{2}} / t^{\frac{k}{2}}$. When $i_1, \dots, i_{k/2}$ ($i_1 < \dots < i_{k/2}$) have already been selected, the probability that $i_{k/2+1} < \dots < i_k$ and $\{i_1, \dots, i_{k/2}\} \cap \{i_{k/2+1},$

..., $i_k\} = \phi$ is $\frac{\binom{t-\frac{k}{2}}{\frac{k}{2}}}{t!} / t^{\frac{k}{2}}$. Hence, the probability that Eq. (1) hold is $\frac{\binom{t}{\frac{k}{2}} \binom{t-\frac{k}{2}}{\frac{k}{2}}}{t^{\frac{k}{2}} t^{\frac{k}{2}}} = \frac{t!}{t^k (\frac{k}{2})^2 (t-k)!}$. \square

As will be shown in section 5.4, the signing cost of Scheme 1 is relatively large. For example, if $t = 1024$ and $k = 8$, the probability that Eq. (1) hold is 0.00169. This means that S can obtain a valid signature after $1/0.00169 = 592$ trials of $h(m \parallel c)$ on average. However, the signature size is smaller than that of any other scheme, which will be shown in section 5.4.

In Scheme 2, key generation requires $2t$ evaluations of $f()$. For signature verification, V must compute $h()$ once and $f()$ $3k/2$ times. For signature generation, S evaluates $h()$ $\frac{t^k}{t(t-1)\dots(t-k+1)}$ times on average by the following theorem.

Theorem 2 Assume that i_1, \dots, i_k are random variables that have uniform distributions over $[1, t]$. In Scheme 2, the probability that all i_1, \dots, i_k are different is $\frac{t(t-1)\dots(t-k+1)}{t^k}$.

Proof: Let a sample space S be $\{i_1, i_2, \dots, i_k \mid (1 \leq i_j \leq t), (1 \leq j \leq k)\}$, where each member of S means the values of i_1, i_2, \dots, i_k . Obviously, $|S| = t^k$. Let us consider the event E where all i_j are different. $|E| = t(t-1) \dots (t-k+1)$. Therefore, the probability that all i_1, \dots, i_k are different is $|E|/|S| = \frac{t(t-1)\dots(t-k+1)}{t^k}$. \square

If k is much smaller than t , $\frac{t(t-1)\dots(t-k+1)}{t^k}$ approaches 1. For example, if $t = 1024$ and $k = 10$ (the case mentioned in section 5.4), the probability that all i_1, \dots, i_k are different is 0.9569. This means that S can obtain a valid signature after $1/0.9569 = 1.045$ trials of $h(m \parallel c)$ on average. In all the cases described [9, 12, 14], the average number of the required hash operations in Scheme 2 is far less than 2.

Up till now, we have dealt with the average number of trials for the proposed schemes. Now, in Schemes 1 and 2, consider the number of trials N s.t. S can find a valid signature with (overwhelming) rate T ($0 < T < 1$). Let P denote the probability that a valid signature is found for a single trial of $h()$. In Scheme 1, $P = \frac{t!}{t^k (\frac{k}{2})^2 (t-k)!}$ and in Scheme 2, $P = \frac{t(t-1)\dots(t-k+1)}{t^k}$. Since $T = 1 - (1 - P)^N$, $N = \log_{1-P}(1 - T)$. In the above example where $t = 1024$ and $k = 10$, for $T = 0.99$, N of Schemes 1 and 2 are 2723 and 1.46, respectively and for $T = 0.5$ (where this condition is adopted from BiBa and Powerball papers [9, 12]), N of Schemes 1 and 2 are 409.7 and 0.22, respectively.

5.2 Key Size and Signature Size

We calculate the key size and signature size of the two schemes. First, the size of PK is tl bits, where l denotes the bit size of a node and it may be on the order of $96 \sim 128$ [9]. Second, the size of SK is $2tl$ bits. However, (p_1, \dots, p_t) can be obtained from (s_1, \dots, s_t) by the off-line computation, where the size of (s_1, \dots, s_t) is tl bits. Third, the size of

SIG is $kl + |c|$ bits, where $|c|$ is related to the average number of the required hash operations in signature generation. For example, if $t = 1024$ and $k = 8$ in Scheme 1, S can obtain a valid signature after 592 trials of $h(m \parallel c)$ on average as mentioned in the previous section. In this case, $|c| = 10$ is sufficient. In Scheme 2, since the average number of $h(m \parallel c)$ operations is less than 2 for most cases, $|c| = 1 \sim 2$ is sufficient.

5.3 Security Analysis

Recall the assumption that $f()$ and $h()$ are modelled as a random oracle [1]. Then, the output of $h()$ shows a uniform distribution. Moreover, given a and $b = h(a)$, the probability that anyone finds $a' \neq a$ such that $b = h(a')$ is $1/2^{|h()|} = 1/2^{k \log_2 t} = 1/t^k$.

Under this assumption, just like [12, 14] we analyze security of the proposed schemes against the r -non-adaptive-message attack. In the r -non-adaptive-message attack, an adversary A is assumed to have r messages of his choice and their signatures. Then, A selects a new message m' and tries to forge a signature of m' . To simplify analysis, just like [12, 14] we assume that A does not attempt to invert or find a collision of one-way functions $f()$. Since our schemes are one-time signature algorithms, we analyze only the case when $r = 1$.

5.3.1 Security analysis on Scheme 1

Assume that an adversary A has a valid signature $SIG = (c, (s_{i_1}, \dots, s_{i_{k/2}}), (p_{i_{k/2+1}}, \dots, p_{i_k}))$ for a message m . Because we assumed that A does not try to invert or find a collision of one-way function $f()$, a valid forged signature SIG' for m' should consist of the elements of SIG except for c , i.e., $SIG' = (c', S')$ s.t. S' consists of only the elements of a set $\{s_{i_1}, \dots, s_{i_{k/2}}, p_{i_{k/2+1}}, \dots, p_{i_k}\}$. Among all possible candidates of S' , $S' = ((s_{i_1}, \dots, s_{i_{k/2}}), (p_{i_{k/2+1}}, \dots, p_{i_k}))$ is the only one to meet Eq. (1) and to be accepted in the verification procedure. Hence, $SIG' = (c', (s_{i_1}, \dots, s_{i_{k/2}}), (p_{i_{k/2+1}}, \dots, p_{i_k}))$.

Moreover, in order to be accepted as a valid signature, $h(m' \parallel c') = i_1 \parallel \dots \parallel i_k$. Because we assumed that the output of $h()$ shows a uniform distribution, the probability that $h(m' \parallel c') = i_1 \parallel \dots \parallel i_k$ is $\frac{1}{2^{|h()|}} = \frac{1}{2^{k \log_2 t}} = \frac{1}{t^k}$.

From above observation, in Scheme 1, the probability for any attacker to find a valid signature after a single trial of $h()$ (we call this a *probability of forgery*) is $1/t^k$.

5.3.2 Security analysis on Scheme 2

Assume that an adversary A has a valid signature $SIG = (c, (s_{i_1}, \dots, s_{i_{k/2}}), (p_{i_{k/2+1}}, \dots, p_{i_k}))$ for a message m . Because we assumed that A does not try to invert or find a collision of one-way function $f()$, a valid forged signature SIG' for m' should consist of the elements of SIG except for c , i.e., $SIG' = (c', S')$ s.t. S' consists of only the elements of a set $\{s_{i_1}, \dots, s_{i_{k/2}}, p_{i_{k/2+1}}, \dots, p_{i_k}\}$. Among all possible candidates of S' , $S' =$ (a permutation of $(s_{i_1}, \dots, s_{i_{k/2}})$, a permutation of $(p_{i_{k/2+1}}, \dots, p_{i_k}))$ is the only one to meet the condition that all i_1, \dots, i_k should be different and that SIG' can be accepted in the verification procedure. Hence, $SIG' = (c',$ (a permutation of $(s_{i_1}, \dots, s_{i_{k/2}})$, a permutation of $(p_{i_{k/2+1}}, \dots, p_{i_k}))$).

Moreover, in order to be accepted as a valid signature, $(i'_1, \dots, i'_{k/2})$ is a permutation

of $(i_1, \dots, i_{k/2})$ and $(i'_{k/2+1}, \dots, i'_k)$ is a permutation of $(i_{k/2+1}, \dots, i_k)$ where $h(m' \| c') = i'_1 \| \dots \| i'_k$. Since we assumed that $h()$ is modelled as a random oracle, its output shows a uniform distribution. Hence, for A 's each trial of $h' = h(m' \| c')$ for different c' , when $h' = h'_1 \| \dots \| h'_k$ is interpreted as k integers i'_1, \dots, i'_k , all i'_1, \dots, i'_k show uniform distributions among $[0, t - 1]$.

Let us consider the probability that $(i'_1, \dots, i'_{k/2})$ is a permutation of $(i_1, \dots, i_{k/2})$ and $(i'_{k/2+1}, \dots, i'_k)$ is a permutation of $(i_{k/2+1}, \dots, i_k)$ for a single trial of $h(m' \| c')$, where i_1, \dots, i_k are given. Let a sample space S be $\{i'_1, \dots, i'_k \mid (1 \leq i'_j \leq t), (1 \leq j \leq k)\}$, where each member of S means the values of i'_1, \dots, i'_k . Obviously, $|S| = t^k$. Let us consider the event C where $(i'_1, \dots, i'_{k/2})$ is a permutation of $(i_1, \dots, i_{k/2})$ and $(i'_{k/2+1}, \dots, i'_k)$ is a permutation of $(i_{k/2+1}, \dots, i_k)$. $|C| = ((k/2)!)^2$. Therefore, the probability that the event C occurs is $|C|/|S| = \frac{((k/2)!)^2}{t^k}$.

Hence, in Scheme 2, the probability for any attacker to find a valid signature after a single trial of $h()$ (probability of forgery) is $\frac{((k/2)!)^2}{t^k}$.

5.4 Comparison Results

Compared schemes are BiBa [12], Powerball [9] and HORS [14]. Table 1 shows the comparison results under the condition that all schemes have the same signature size ($= kl$) and the same public key size ($= tl$), where l denotes the bit size of a node and it may be on the order of $96 \sim 128$ [9]. Under this condition, the probability of forgery P_f of Scheme 1 is the lowest. P_f of Scheme 2 is lower than those of BiBa and HORS for all the possible values of (k, t, l) and P_f of Powerball is higher than that of Scheme 2 (except for the cases $k = 2, 4$).

Table 1. Comparison results where all schemes have the same $|SIG|$ and $|PK|$.

Scheme	Signature size (bits)	Verification cost	Key generation cost	Signing cost	Public key size (bits)	Probability of forgery
BiBa [12]	kl	$2k + 1$	t	$2t$	tl	$\frac{k!^+}{2t^k}$
Powerball [9]	kl	$2k + 1$	$2t$	$2t$	tl	$\frac{(k-1)!^+}{2t^k}$
HORS [14]	kl	$k + 1$	t	1	tl	$(\frac{k}{t})^k$
Scheme 1	kl	$\frac{3k}{2} + 1$	$2t$	$\frac{t^k ((k/2)!)^2 (t-k)!}{t!}$	tl	$(\frac{1}{t})^k$
Scheme 2	kl	$\frac{3k}{2} + 1$	$2t$	$\frac{t^k}{t(t-1)\dots(t-k+1)}$	tl	$\frac{((k/2)!)^2}{t^k}$

⁺ The values are from Theorem 1 of [9] and section 6 of [9] where $P_s = 1/2$.

For all the schemes, if k becomes smaller (or larger), the probability of forgery increases (or decreases). Hence, when all the schemes have the same probability of forgery, Scheme 1 has the smallest signature size and the next is Scheme 2.

Key generation costs of our schemes are higher than that of HORS or BiBa. How-

ever, this may not be significant since key generation can be pre-processed and easily parallelized over multiple servers.

Table 2 shows the comparison results under the condition adopted from [9], where the probability of forgery, t and message size are fixed as 2^{-80} , 1024 and 80bits, respectively (k has a different value for each scheme). Under this condition, Scheme 1 has the smallest signature size. Scheme 2 and Powerball are the next ($k = 10$). Note that P_f of Scheme 2 is 2^{-86} , which is much lower than that of Powerball ($= 2^{-82}$). The signature size of HORS is larger than those of our schemes, Powerball and BiBa. Table 2 shows that signing cost of Scheme 2 is close to 1 and that verification cost is low for Schemes 1 and 2.

Table 2. Comparison results where all schemes have the same security level.

Scheme	Signature size (bits)	Signing cost	Verification cost	Key generation cost	Public key size (bits)
Lamport ⁺ [6]	80 l	1	80	160	160 l
Merkle-Winternitz ⁺ [7]	23 l	1	169	355	1 l
Bleichenbacher-Manurer ⁺ [2]	45 l	1	72	182	1 l
BiBa ⁺ [12]	11 l [¶]	2048	23	1024	1024 l
Powerball ⁺ [9]	10 l [¶]	2048	21 [‡]	2048	1024 l
HORS [14]	13 l	1	14	1024	1024 l
Scheme 1 [§]	8 l [§]	592	13	2048	1024 l
Scheme 2 [§]	10 l [§]	1.045	16	2048	1024 l

⁺ The values are adopted from Table 6 of [9].

[‡] The value ($= 20$) described in Table 6 of [9] is erroneous.

[¶] The value will be larger by 1~2 [9, 12].

[§] Because of c in *SIG*, the value will be larger by $|c|$ (see section 5.2).

Although on-line signing cost of Scheme 1 is higher than that of HORS, it is lower than those of Powerball and BiBa. A higher on-line signing cost against HORS may not be significant due to the following reasons:

- Hash operation is very cheap, e.g., for MD5 algorithm, Celeron 850Mhz can hash 805.7Mb per second [3].
- Note that computing $h(m \parallel c)$ and verifying Eq. (1) do not require SK to be accessed. Hence, when signing operation is parallelized over distributed servers, they need not share SK . Unlike our scheme, Powerball or BiBa requires SK to be shared among the servers. If one of these servers is compromised, SK would be exposed.
- The parameterized algorithm in Appendix enables the trade-off between key generation cost and signing cost. Table 3 in Appendix shows that by increasing the value d , signing cost can be reduced down to 16.4 whereas key generation cost is increased up to 4078.
- For stream authentication, the signing operation is required only for a broadcast server, whereas there are lots of receivers who may have low computing power or low bandwidth. Hence, a small signature size and low verification cost can be more important than low signing cost.

- The condition that the probability of forgery is equal to 2^{-80} is very strict because [12, 14] assumed the probability to be 2^{-58} or 2^{-43} . If the probability of forgery is higher, it is possible that k and t could be smaller, which in turn would significantly reduce the on-line signing cost of Scheme 1.

6. CONCLUSIONS

In this paper, we proposed two efficient one-time signature schemes for stream authentication. Compared with BiBa, Powerball and HORS, the proposed schemes have the smallest signature sizes. Moreover, our schemes have low verification overheads. The signature size of Scheme 1 is smaller than that of Scheme 2 whereas Scheme 2 has much smaller signing cost: it requires only 2 hash operations in the majority of cases. In Scheme 1, relatively high signing cost can be parallelized without any additional risk because sharing the private key among distributed servers is not required.

REFERENCES

1. M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in *Proceedings of 1st ACM Conference on Computer and Communication Security*, 1993, pp. 62-73.
2. D. Bleichenbacher and U. Maurer, "Optimal tree-based one-time digital signature schemes," in *Proceedings of 13th Symposium on Theoretical Aspects of Computer Science*, LNCS 1046, Springer-Verlag, 1996, pp. 363-374.
3. W. Dai, "Crypto++ benchmarks," <http://www.eskimo.com/~weidai/benchmarks.html>.
4. R. Gennaro and P. Rohatgi, "How to sign digital streams," in *Proceedings of CRYPTO '97*, LNCS 1294, Springer-Verlag, 1997, pp. 180-197.
5. 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*, 2001, pp. 13-22.
6. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
7. R. C. Merkle, "A digital signature based on a conventional encryption function," in *Proceedings of CRYPTO '87*, LNCS 293, Springer-Verlag, 1987, pp. 369-378.
8. S. Miner and J. Staddon, "Graph-based authentication of digital streams," in *Proceedings of IEEE Security and Privacy Symposium*, 2001, pp. 232-246.
9. M. Mitzenmacher and A. Perrig, "Bounds and improvements for BiBa signature schemes," No. TR-02-02, Computer Science Group, Harvard University, U.S.A., 2002.
10. J. M. Park, E. K. P. Chong, and H. J. Siegel, "Efficient multicast packet authentication using signature amortization," in *Proceedings of the IEEE Security and Privacy Symposium*, 2002, pp. 227-240.
11. Y. Park, T. Chung, and Y. Cho, "An efficient stream authentication scheme using tree chaining," *Information Processing Letters*, Vol. 86, 2003, pp. 1-8.
12. A. Perrig, "The BiBa one-time signature and broadcast authentication protocol," in

- Proceedings of 8th ACM Conference on Computer and Communication Security*, 2001, pp. 28-37.
13. A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient authentication and signing of multicast streams over lossy channels," in *Proceedings of the IEEE Security and Privacy Symposium*, 2000, pp. 56-73.
 14. L. Reyzin and N. Reyzin, "Better than BiBa: Short one-time signatures with fast signing and verifying," in *Proceedings of the Australian Conference on Information Security and Privacy*, 2002, pp. 114-153.
 15. P. Rohatgi, "A compact and fast hybrid signature scheme for multicast packet authentication," in *Proceedings of 6th ACM Conference on Computer and Communication Security*, 1999, pp. 93-100.
 16. D. Song, D. Zuckerman, and J. D. Tygar, "Expander graphs for digital stream authentication and robust overlay networks," in *Proceedings of the IEEE Security and Privacy Symposium*, 2002, pp. 258-270.
 17. C. K. Wong and S. S. Lam, "Digital signatures for flows and multicasts," *IEEE/ACM Transactions on Networking*, Vol. 7, 1999, pp. 502-513.

APPENDIX

On-line computation cost of Scheme 1 is larger than that of HORS, where HORS requires that $h()$ computation be conducted only once for signing operation. However,

our algorithm requires $\frac{t!}{t^k \binom{k}{2}^2 (t-k)!} h()$ computations by Theorem 1 described in section

5. We propose a parameterized algorithm in which the trade-off between key generation cost and signing cost is possible by selecting a value d . Scheme 1 described in section 4.1 is identical to this algorithm with $d = 2$.

First, key generation is as follows. For input parameters l, k, t and d , S generates t random l -bit strings s_1^0, \dots, s_t^0 . Then, S computes $s_i^j = f(s_i^{j-1})$ for $1 \leq i \leq t$ and $1 \leq j \leq d$. All s_a^b ($1 \leq a < t, 0 \leq b \leq d$) should be different. The private key and public key are $SK = ((s_1^0, \dots, s_t^0), \dots, (s_1^{d-1}, \dots, s_t^{d-1}))$ and $PK = (s_1^d, \dots, s_t^d)$, respectively. Second, signature generation is as follows. Given a message m and SK , S selects a random value c and computes $h(m \parallel c) = h_1 \parallel \dots \parallel h_k$. h_j is interpreted as an integer i_j ($1 \leq j \leq k$). i_j ($1 \leq j \leq k$) must meet the following equations. If not, S repeats above procedure for another c .

$$i_1 < \dots < i_{k/d}, i_{k/d+1} < \dots < i_{2k/d}, \dots, i_{(d-1)k/d+1} < \dots < i_k, \\ \{i_1, \dots, i_{k/d}\}, \{i_{k/d+1}, \dots, i_{2k/d}\}, \dots, \{i_{(d-1)k/d+1}, \dots, i_k\} \text{ are pairwise disjoint.} \quad (2)$$

The signature is $SIG = (c, s_{i_1}^0, \dots, s_{i_{k/d}}^0), \dots, (s_{i_{(d-1)k/d+1}}^{d-1}, \dots, s_{i_k}^{d-1})$. Third, signature verification is as follows. Given a message m , $SIG = (c', (s'_1, \dots, s'_{k/d}), \dots, (s'_{(d-1)k/d+1}, \dots, s'_k))$ and PK , V computes $h(m \parallel c') = h'_1 \parallel \dots \parallel h'_k$. V interprets h'_j as an integer i'_j ($1 \leq j \leq k$). i'_j ($1 \leq j \leq k$) must meet Eq. (2). If $f^{d-n}(s'_j) = s_{i'_j}^d$ for $nk/d < j \leq (n+1)k/d$ and $0 \leq n \leq d-1$ (where $f^1() = f()$ and $f^n() = f(f^{n-1}())$), V accepts the signature.

The above algorithm has the constraint that $d \mid k$. A full description of the general

algorithm that does not have this constraint is in Algorithm 1, Algorithm 2 and Algorithm 3.

The computation cost can be calculated by using the methods described in section 5.1. Table 3 shows the comparison results of the computation cost under the condition described in section 5.4.

Table 3. The parameterized algorithm vs. the algorithm described in section 4.1.

Scheme	Signature size (bits)	Verification cost	Key generation cost	Signing cost	Public key size (bits)
Scheme 1 in Section 4.1	$8l$	13	2048	592	$1024l$
Parameterized algorithm ($d=2$)	$8l$	13	2048	592	$1024l$
Parameterized algorithm ($d=3$)	$8l$	16	3072	74	$1024l$
Parameterized algorithm ($d=4$)	$8l$	21	4096	16.5	$1024l$

Because the parameterized algorithm with $d = 2$ is identical to Scheme 1 described in section 4.1, each value of the second row is the same as that of the third row in Table 3. As d is larger, key generation cost and verification cost become larger whereas on-line signing cost becomes smaller.

Algorithm 1 Key generation module

1. Input: l, k, t and d .
2. Output: SK and PK .
3. S generates t random l -bit strings s_1^0, \dots, s_t^0 .
4. S computes $s_i^j = f(s_i^{j-1})$ for $1 \leq i \leq t$ and $1 \leq j \leq d$.
5. $SK = ((s_1^0, \dots, s_t^0), \dots, (s_1^{d-1}, \dots, s_t^{d-1}))$ and $PK = (s_1^d, \dots, s_t^d)$, where all s_a^b ($1 \leq a < t, 0 \leq b \leq d$) should be different.

Algorithm 2 Signature generation module

1. Input: l, k, t, d, m and SK .
2. Output: SIG .
3. $r = k/d - \lfloor k/d \rfloor$.
4. **repeat**
5. Select a random value c and compute $h(m \parallel c) = h_1 \parallel \dots \parallel h_k$.
6. h_j is interpreted as an integer i_j ($1 \leq j \leq k$).
7. **until** i_j ($1 \leq j \leq k$) must meet the following equations:
 $i_1 < \dots < i_{\lfloor k/d \rfloor + 1 - r}, i_{\lfloor k/d \rfloor + 1 - r + 1} < i_{2\lfloor k/d \rfloor + 2 - r}, \dots, i_{(d-1)\lfloor k/d \rfloor + (d-1)r + 1} < \dots < i_k$, ($\{i_1, \dots, i_{\lfloor k/d \rfloor + 1 - r}\}, \{i_{\lfloor k/d \rfloor + 1 - r + 1}, \dots, i_{2\lfloor k/d \rfloor + 2 - r}\}, \dots, \{i_{(d-1)\lfloor k/d \rfloor + (d-1)r + 1}, \dots, i_k\}$ are pairwise disjoint).
8. $SIG = (c, (s_{i_1}^0, \dots, s_{i_{\lfloor k/d \rfloor + 1 - r}}^0), \dots, (s_{i_{(d-1)\lfloor k/d \rfloor + (d-1)r + 1}}^{d-1}, \dots, s_{i_k}^{d-1}))$.

Algorithm 3 Signature verification module

1. Input: l, k, t, d, m, PK and $SIG = (c', (s'_1, \dots, s'_{\lfloor k/d \rfloor + \lfloor 1 \cdot r \rfloor}), \dots, (s'_{(d-1)\lfloor k/d \rfloor + \lfloor (d-1)r \rfloor + 1}, \dots, s'_k))$, where $r = k/d - \lfloor k/d \rfloor$.
2. Output: success or failure.
3. Select a random value c' and compute $h(m \parallel c') = h'_1 \parallel \dots \parallel h'_k$.
4. Interpret h'_j as an integer i'_j ($1 \leq j \leq k$).
5. **if** i'_j ($1 \leq j \leq k$) do not meet the equations in step 7 of Algorithm 2 **then**
6. Output "failure".
7. **end if**
8. **if** $f^{d-n}(s'_j) = s_i^d$ for $n\lfloor k/d \rfloor + \lfloor nr \rfloor < j \leq (n+1)\lfloor k/d \rfloor + \lfloor (n+1)r \rfloor$ and $0 \leq n \leq d-1$ **then**
9. Output "success".
10. **else**
11. Output "failure".
12. **end if**



Yongsu Park received the B.E. degree in Computer Science from Korea Advance Institute of Science and Technology (KAIST), South Korea, in 1996. He received the M.E. degree and the Ph.D. degree in Computer Engineering from Seoul National University in 1998 and 2003, respectively. He is currently an assistant professor in the College of Information and Communications at Hanyang University, Seoul, Korea. His main research interests include computer system security, network security, and cryptography.



Yookun Cho received the B.E. degree from Seoul National University, Seoul, Korea, in 1971, and the Ph.D. degree in Computer Science from University of Minnesota at Minneapolis, Minnesota, U.S.A., in 1978. He has been with the School of Computer Science and Engineering, Seoul National University since 1979, where currently he is a professor. He was a visiting assistant professor at the University of Minnesota during 1985, and a director of Educational and Research Computing Center at Seoul National University from 1993 to 1995. He was the member of program committee of the IPPS/SPDP'98 in 1997, and the International Conference on High Performance Computing from 1995 to 1997. He was the president of the Korea Information Science Society from 2001 to 2002. He is a member of the National Academy of Engineering of Korea. His research interests include operating systems, algorithms, system security, and fault-tolerant computing systems.