# How to Authenticate Real Time Streams Using Improved Online/Offline Signatures

Chong-zhi Gao[1] and Zheng-an Yao[2]

[1] Information, Machinery and Electronics College,
Guangzhou University,
GuangZhou 510000, China
[2] College of Mathematic and Computational Science,
Sun Yat-Sen University,
GuangZhou 510275, China

**Abstract.** Providing authentication protocols for real time streams is a challenging task. This is because the authentication rate is very important for real time streams, whereas it is usually a bottleneck. Using improved online/offline signatures and hash chain techniques as tools, our proposed protocol greatly reduces the online computational and communicational cost and thus is more applicable to authenticate real time streams.

**Keywords:** Stream Authentication, Real Time Streams, Online/Offline Signatures.

## 1 Introduction

A digital stream is a (potentially infinite) sequence of bits that a sender transmits to a receiver. With the growth of the Internet and the popularization of electronic commerce, there are more and more applications that need data transmissions such as live video/radio broadcastings and real time stock quote systems. In the data transmission, people are usually concerned with the following issues:

1) *Privacy:* the sender keeps information secret from those who are unauthorized to see it.
2) *Integrity:* the receiver can ensure that information has not been altered by any unauthorized parties.
3) *Authenticity:* the receiver can corroborate that the received information is sent by a certain party.
4) *Non-repudiation:* the receiver can prove to a third party that the information is sent by a certain party.

Over the years, researchers have proposed various techniques to achieve these objects. For example, public encryption schemes[21, 3, 13] were proposed to ensure the privacy of data and signature schemes[2, 21, 20] were proposed to ensure the authenticity.

Y.G. Desmedt et al. (Eds.): CANS 2005, LNCS 3810, pp. 134–146, 2005.

### 1.1 Authentication for Real Time Streams

In this paper, our goal is to provide authenticity as well as integrity and non-repudiation for real time streams. A real time stream is quite different from a non-real time stream since the sender cannot be expected to obtain the entire stream before or on sending the stream. Thus, the sender can only buffer few packets while transmitting these streams. In addition, the authentication rate must be higher than the stream generation rate while this is not required for non-real time streams.

### 1.2 Related Work

A trivial authentication method is to sign each packet[7]. The sender first splits the stream into packets and signs each packet one by one. The receiver then verifies these signatures after he/she receives the packets and their corresponding signatures. However, this method has its disadvantage since every packet requires a sign/verify computation and thus the computational cost is quite heavy. In addition, adding a signature to each packet will greatly increase the communication overhead.

In 1997, Gennaro and Rohatgi [7] proposed two paradigms for stream authentications. In the paradigm for streams that can be known in advance by the sender, they use hash chain techniques and signature techniques to authenticate streams. In this paradigm, although a signature is amortized over several packets, the computational cost is still high since a signature operation is very inefficient. In the paradigm for streams that can not be known in advance, they employ one time signatures introduced in [12, 14]. This paradigm results in a large communication overhead since the signature size and the key size of one-time signatures are very large.

In 1998, Wong and Lam [23] proposed a tree chaining technique to authenticate streams. Their construction is robust to any number of losses in streams. However, the communication overhead per packet is quite large (even larger than the size of a digital signature) and thus is not practical.

Miner and Staddon [15] proposed a graph-based authentication protocol in 2001. In their transmission model, each packet is assumed to be lost independently with the same probability and the protocol is designed based on this probability.

There are other authentication protocols for streams such as Perrig et al.'s EMSS and TESLA scheme[18], Wu et al.'s object-based scheme[24] and Pannetract and Molva's EC scheme[17]. Some of them, e.g., the TESLA scheme in [18] and the objected based scheme in [24], do not offer non-repudiation.

### 1.3 Contribution

In previous works, using ordinary signature schemes such as RSA[21], DSA[16], FFS[6, 5] or eFFS[23] will result in heavy computational cost[7, 15, 24, 17] or a large communication overhead[23]. However, the so-called online/offline

signatures[4, 22] can avoid this shortcoming by dividing the authentication procedure into two phases. The first phase is offline phase: this phase's work can be carried out any time before the stream to be transmitted is generated. The second phase is online phase: this phase starts once the stream begins to be generated. Dividing authentication into two phases can avoid the computational and communicational bottlenecks that usually occur at the time of transmitting streams.

In this paper, using our improved online/offline signature schemes and the hash chain techniques[7, 19, 10] as building blocks, we construct an efficient authentication protocol for real time streams. Compared to previous protocols, our new construction has the following advantages:

1. By using our improved online/offline technique, we have greatly reduced the computational and communicational cost of authenticating streams.
2. By using the hash chain techniques, we amortize a single signing/verification operation over many packets.
3. Our protocol can tolerate 1 bursty loss of packet in a block. In addition, the ability of tolerating packet loss can be strengthened by using more complex hash chain constructions[18, 10].

### 1.4 Organization

The rest of this paper is organized as follows. In section 2, we give preliminaries. Section 3 reviews online/offline signatures and proposes a new scheme to improve the performance. Using the improved online/offline signature as a crucial building block, section 4 proposes an efficient authentication protocol for real time streams. Section 5 concludes this paper.

## 2 Preliminaries

### 2.1 Notations

The most of the following notations are borrowed from [8]. We denote by $\mathbb{N}$ the set of natural numbers. If $k \in \mathbb{N}$, we denote by $1^k$ the concatenation of $k$ ones and by $\{0,1\}^k$ the set of bitstrings of bitlength $k$. By $\{0,1\}^*$, we denote the set of bitstrings of arbitrary bitlength.

If $S$ is a finite set, then the notation $x \xleftarrow{R} S$ denotes that $x$ is selected randomly from the set $S$. If $\mathcal{A}$ is a algorithm, by $\mathcal{A}(\cdot)$ we denote that $\mathcal{A}$ receives only one input. If $\mathcal{A}$ receives two inputs we write $\mathcal{A}(\cdot,\cdot)$ and so on. If $\mathcal{A}(\cdot)$ is a probabilistic algorithm, $y \leftarrow \mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2,\cdots}(x_1, x_2, \ldots)$ means that on input $x_1, x_2, \ldots$ and with access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$, $\mathcal{A}$'s output is $y$. If $p(\cdot,\cdot,\ldots)$ is a predicate, the notation $Pr[p(x,y,\ldots) : x \xleftarrow{R} S; y \xleftarrow{R} T; \ldots]$ denotes the probability that $p(x,y,\ldots)$ will be true after the ordered execution of the algorithms $x \xleftarrow{R} S, y \xleftarrow{R} T, \ldots$. "PPT" is an abbreviation for "probabilistic polynomial-time" and "$\|$" represents the concatenation operation.

## 3 Online/Offline Signatures and Our Improvement

The main building blocks of our authentication protocol are our improved online/offline signature scheme (which was first introduced in[4]) and the hash chain techniques. In this section, we first review the definition and the state of art of online/offline signatures. Then, In order to improve the performance, we propose an improved online/offline signature scheme as well as a security proof.

An online/offline signature scheme is a signature scheme used in a particular scenario where the signer must response quickly once the message to be signed is presented. This notion was first introduced by Even, Goldreich and Micali in 1990[4]. The idea of online/offline signatures is to split the signing procedure into two phases. The first phase is offline: in this phase, the signer does some preparatory work before the message to be signed is presented. The second phase is online: once the message to be signed is known, the signer utilizes the result of the pre-computation and use a very short time to accomplish the signing procedure.

Even et al. utilized one-time signature schemes to construct online/offline signatures. Their method can convert any signature schemes into online/offline signature schemes. Due to the use of one-time signatures, the resulting length of signatures is very long. Therefore, the method is not practical.

In 2001, Shamir and Tauman[22] proposed a new online/offline signature scheme which is based on trapdoor hash functions[11]. A trapdoor hash function is a special type of hash function which is associated with a public (hashing) key $pk$ and a secret key. It has two inputs and is written as $H_{pk}(\cdot;\cdot)$. Given $pk$ and an input pair $(m, r)$, everyone can compute the value of $H_{pk}(m; r)$. But only the person who holds the secret key can find collisions of the hash function. Shamir and Tauman's improved scheme highly enhanced the efficiency of signing, especially the efficiency in the online phase.

We call the schemes proposed by Even et al.[4] $\mathcal{OT}$-$\mathcal{OS}$ scheme (online/offline signatures based on one-time signatures) and the schemes proposed by Shamir et al.[22] $\mathcal{HSS}$-$\mathcal{OS}$ scheme (online/offline signatures using the hash-sign-switch paradigm). Both $\mathcal{OT}$-$\mathcal{OS}$ and $\mathcal{HSS}$-$\mathcal{OS}$ utilize a standard signature and another type of complex computation ($\mathcal{OT}$-$\mathcal{OS}$ should compute one-time signatures and $\mathcal{HSS}$-$\mathcal{OS}$ should evaluate trapdoor hash functions). This increases the complexity of online/offline signatures. Although $\mathcal{HSS}$-$\mathcal{OS}$'s efficiency in the online phase is very high, the computational cost of the offline phase and verification procedure is heavy. Furthermore, $\mathcal{HSS}$-$\mathcal{OS}$ requires the signer hold two private keys: one is for standard signatures, the other is for trapdoor hash functions. Exposure of any of the keys could lead to a total break of the scheme.

In the following, we give a formal syntax of online/offline signatures.

### 3.1 The Syntax of Online/Offline Signatures

An online/offline signature scheme ($\mathcal{OS}$) is a triple of algorithms $(\mathcal{G}, \mathsf{Sign}, \mathsf{Ver})$.

– $(pk, sk) \leftarrow \mathcal{G}(1^k)$ is a PPT algorithm which on input a security parameter $k \in \mathbb{N}$, outputs a public/private key pair $(pk, sk)$.

- $\sigma \leftarrow \mathsf{Sign}(sk, M)$ is a PPT algorithm which on input a private key $sk$ and a message $M$, outputs a signature $\sigma$. The signing algorithm consists of two sub-algorithms:
  - $St \leftarrow \mathsf{Sign\_off}(sk)$ is a PPT algorithm which on input the private key $sk$, outputs a state information $St$.
  - $\sigma \leftarrow \mathsf{Sign\_on}(St, M)$ is a PPT algorithm which on input a state $St$ and a message $M$, outputs a signature $\sigma$.

  The signing algorithm $\mathsf{Sign}$ first runs $\mathsf{Sign\_off}(sk)$ to get $St$, then transfers $St$ and the message $M$ to $\mathsf{Sign\_on}(\cdot, \cdot)$, finally it returns the signature $\sigma$ which is the output of $\mathsf{Sign\_on}(St, M)$ .
- $0/1 \leftarrow \mathsf{Ver}(pk, M, \sigma)$ is a PPT algorithm which on input the public key $pk$, a message $M$ and a signature $\sigma$, outputs 0 or 1 for $\mathsf{reject}$ or $\mathsf{accept}$ respectively.

***Completeness:*** It is required that if $\mathsf{Sign}(sk, M) = \sigma$ then $\mathsf{Ver}(pk, M, \sigma) = 1$ for all $(pk, sk)$ generated by $\mathcal{G}(1^k)$.

### 3.2 Security Notion

On defining the security notion of an online/offline signature scheme, we view an $\mathcal{OS}$ scheme as a standard signature scheme and use the security notion called existential unforgeability under chosen message attacks [9] in the random oracle model [1].

***Existential Unforgeability:*** Existential unforgeability for $\mathcal{OS}$ under chosen message attacks in the random oracle model is defined in the following game. This game is carried out between a simulator $\mathcal{S}$ and an adversary $\mathcal{A}$. The adversary $\mathcal{A}$ is allowed to make queries to a sign-oracle $\mathsf{Sign}(sk, \cdot)$ and a hash oracle $h(\cdot)$. The attacking game is as follows:

1. The simulator runs $\mathcal{G}$ on input $1^k$ to get $(pk, sk)$. $pk$ is sent to $\mathcal{A}$.
2. On input $(1^k, pk)$, $\mathcal{A}$ is allowed to query the sign-oracle $\mathsf{Sign}(sk, \cdot)$ and the hash oracle $h(\cdot)$ polynomial times.
3. $\mathcal{A}$ outputs a pair $(M, \sigma)$.

The adversary wins the game if the message $M$ has never been queried to the oracle $\mathsf{Sign}(sk, \cdot)$ and $\mathsf{Ver}(pk, M, \sigma) = 1$ holds. Let $\mathrm{Adv}_{\mathcal{A},\mathcal{OS}}$ be the *advantage* of the adversary $\mathcal{A}$ in breaking the signature scheme, i.e.

$$\mathrm{Adv}_{\mathcal{A},\mathcal{OS}} = \Pr[\mathsf{Ver}(pk, M, \sigma) = 1 : (pk, sk) \leftarrow \mathcal{G}(1^k); (M, \sigma) \leftarrow \mathcal{A}^{h(\cdot), \mathsf{Sign}^{h(\cdot)}(sk, \cdot)}]$$

where $\mathcal{A}$ has never requested $M$ to the signing oracle and the probability is taken over the internal coin tosses of the algorithm $\mathcal{G}$ and $\mathcal{A}$.

**Definition 1.** *An adversary $\mathcal{A}$ $(t, q_s, q_h, \epsilon)$-breaks an online/offline signature scheme $\mathcal{OS}$ if $\mathcal{A}$ runs in time at most $t$, makes at most $q_s$ queries to the signing oracle and at most $q_h$ queries to the hash oracle, and $\mathrm{Adv}_{\mathcal{A},\mathcal{OS}}$ is at least $\epsilon$.*

*A signature scheme $\mathcal{OS}$ is existentially unforgeable under chosen message attacks if for every PPT adversary $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A},\mathcal{OS}}$ is negligible.*

### 3.3 Our New Construction of Online/Offline Signatures

To improve Shamir and Tauman's online/offline signature scheme $\mathcal{HSS}$-$\mathcal{OS}$, we propose a new construction and prove its security. Compared to the scheme $\mathcal{HSS}$-$\mathcal{OS}$, there is an almost 50% reduction in signature size and overall computational cost.

The new online/offline signature scheme $\mathcal{TH}$-$\mathcal{OS}$ is based on a trapdoor hash family $\mathcal{TH} = (\mathcal{G}, \mathcal{H}, \mathcal{F})$ (A formal definition of a trapdoor hash family including a security definition is given in Appendix A). A standard collision free hash function $h : \{0,1\}^* \rightarrow \{0,1\}^{\alpha(k)}$ (where $\alpha()$ is a polynomial function of the security parameter $k$) is also needed, which will be treated as a random oracle in the proof of security.

The scheme $\mathcal{TH}$-$\mathcal{OS} = (\mathcal{G}', \mathsf{Sign}, \mathsf{Ver})$ is constructed as follows:

- $\boldsymbol{\mathcal{G}'}(1^k)$. The key generation algorithm is set to be $\mathcal{G}$, i.e., $\mathcal{G}' = \mathcal{G}$. It takes as input a security parameter $k$, outputs a key pair $(pk, sk)$. Let $\mathcal{M}_{pk}, \mathcal{R}_{pk}$ and $\mathcal{Q}_{pk}$ be $\mathcal{TH}$'s message space, tag space and range set resp. It is required that $\{0,1\}^{\alpha(k)} \subset \mathcal{M}_{pk}$ holds.
- **Sign_off**$(sk)$. Given a secret key $sk$ , proceeds as follows:
  1. Select at random $(m, r) \in_R \mathcal{M}_{pk} \times \mathcal{R}_{pk}$, and compute $\theta = H_{pk}(m; r)$.
  2. Let $St = (sk, \theta, m, r)$.

  *Remark 1.* Assigning $(sk, \theta, m, r)$ instead of $(sk, m, r)$ to $St$ is to avoid recomputing the value $\theta = H_{pk}(m; r)$ in the online phase.

- **Sign_on**$(St, M)$. Given $St = (sk, \theta, m, r)$ and a message $M \in \{0,1\}^*$, computes the signature as follows:
  1. Compute $m' = h(M \| \theta)$.
  2. Run the collision-finding algorithm $\mathcal{F}$ of $\mathcal{TH}$ with the input $(1^k, sk, m, r, m')$ to obtain $r'$ such that $H_{pk}(m'; r') = H_{pk}(m; r)$.
  3. Output the signature as $\sigma = (\theta, r')$.
- **Ver**$(pk, M, \sigma)$. Given a public key $pk$, a message $M$, and a signature $\sigma = (\theta, r')$, checks that $H_{pk}(h(M \| \theta); r') \stackrel{?}{=} \theta$. Output 1 if this check succeeds and output 0 otherwise.

**Completeness:** it is straightforward.

### 3.4 Security and Efficiency

**Theorem 1.** *Let $\mathcal{TH}$ be a uniform trapdoor hash family with a super-logarithmic min-entropy $\beta(\cdot)$. Let $\mathcal{TH}$-$\mathcal{OS}$ be the associated online/offline signature scheme as constructed in Section 3.3. Then $\mathcal{TH}$-$\mathcal{OS}$ is existentially unforgeable under chosen message attacks in the random oracle model if $\mathcal{TH}$ is collision resistant.*

The proof of the above is based on relatively standard ideas, but is complicated by details of the simulations and models. Due to the page limitation, we present it in the full paper.

**Efficiency.** In the new scheme $\mathcal{TH}$-$\mathcal{OS}$, the offline phase involves only one evaluation of an trapdoor hash function. The online phase involves one operation of finding a collision and one evaluation of a standard hash function. Using the trapdoor hash family proposed by [22], the operation of finding a collision requires about 0.1 modular multiplication of two 1024 bit numbers. Our scheme's verification algorithm involves one evaluation of an trapdoor hash function and one evaluation of a standard hash function. We compare the efficiency of the Hash-Sign-Switch paradigm ($\mathcal{HSS}$-$\mathcal{OS}$) [22] and our new scheme in Table 1.

**Table 1.** The cost of two online/offline signature schemes. Abbreviations used are: "eva-TH" for an evaluation of a trapdoor hash function; "eva-SH" for an evaluation of a standard hash function; "sign-SS" and "ver-SS" for the signing and verification algorithm of a standard signature scheme respectively. Note that the evaluation of a standard hash function is very efficient, therefore we use an asterisk to remark it.

| Schemes | Sign_off | Sign_on | Ver | Signature size |
|---|---|---|---|---|
| $\mathcal{HSS}$-$\mathcal{OS}$ (The Hash-Sign-Switch Paradigm) | 1 eva-TH<br>1 sign-SS | 1 finding collision | 1 eva-TH<br>1 ver-SS | 1 standard sig;<br>1 point in $\mathcal{Q} \times \mathcal{R}$ |
| The New Scheme $\mathcal{TH}$-$\mathcal{OS}$ | 1 eva-TH | 1 eva-SH *<br>1 finding collision | 1 eva-SH *<br>1 eva-TH | 1 point in $\mathcal{Q} \times \mathcal{R}$ |

We can see that operations of the standard signature are eliminated in the new scheme, at the cost of additional single hash evaluation in the online phase. Thus, the new scheme need only one private key instead of two private keys in the $\mathcal{HSS}$-$\mathcal{OS}$ scheme and there is an almost 50% reduction in the signature size and the overall computational cost.

## 4 Authenticating Real Time Streams

### 4.1 The New Protocol

Using improved online/offline signatures and the hash chain techniques as building blocks, we can construct an efficient protocol to authenticate real time streams. The basic idea is to split the authentication procedure into two phases just like in online/offline signature schemes. In the first phase(offline phase) , the sender does some preparatory work before the streams to be send are known. In the second phase(online phase), on obtaining the streams, the sender attaches authentication information on streams and send these data to the receiver. The following is the details of this protocol.

Suppose the sender is capable of buffering $n$ packets. Suppose $\mathcal{TH} = (\mathcal{G}, \mathcal{H}, \mathcal{F})$ is a trapdoor hash family and $h : \{0,1\}^* \rightarrow \{0,1\}^{\alpha(k)}$ is a standard collision free hash function. Let $pk$ be the sender's public key and $sk$ be the corresponding private key. Consider $n$ packets that constitute a block.

**Offline phase:**

1. The sender randomly selects $s$ pairs:

$$(\widetilde{m_1}, \widetilde{r_1}); (\widetilde{m_2}, \widetilde{r_2}); \ldots; (\widetilde{m_s}, \widetilde{r_s}) \qquad (\ (\widetilde{m}_j, \widetilde{r}_j) \in \mathcal{M} \times \mathcal{R}\ , 1 \leq j \leq s)$$

   and computes $\theta_j = H_{pk}(\widetilde{m}_j; \widetilde{r}_j)\ \ (1 \leq j \leq s)$.
2. The sender stores $\widetilde{m}_j, \widetilde{r}_j, \theta_j\ \ (j = 1, 2, \ldots)$ and sends $\theta_j\ \ (j = 1, 2, \ldots)$ to the receiver.

*Remark 2.* The number $s$ depends on the number of blocks that the sender want to send.

*Remark 3.* The offline phase can be carried out any time before the streams to be send are known. Thus, we can greatly reduce the computational cost and communication overhead while sending the streams. This is the most important contribution of our scheme.

**Online phase:**

Suppose the $j$-th block's packets are $m_1, m_2, \ldots, m_n$. For the $j$-th block, the sender

1. Computes

$$D_n = h(m_n \| 00 \ldots 0)$$
$$D_{n-1} = h(m_{n-1} \| D_n \| 00 \ldots 0)$$
$$D_i = h(m_i \| D_{i+1} \| D_{i+2})\ \ (1 \leq i \leq n-2)$$

2. uses the private key $sk$ and $(\widetilde{m}_j, \widetilde{r}_j)$ to compute $r_j$(see section 3.3 for details) such that
$$H_{pk}(h(D_1 \| \theta_j); r_j) = H_{pk}(\widetilde{m}_j; \widetilde{r}_j).$$
3. sends $< (r_j, D_1); (m_1, D_2, D_3); (m_2, D_3, D_4); \ldots; (m_{n-2}, D_{n-1}, D_n); (m_{n-1}, D_n); m_n >$ to the receiver where $(r_j, D_1)$ is the header and $(m_i, D_{i+1}, D_{i+2})$ is the $i$-th packet.

*Remark 4.* If we use the trapdoor hash family in Appendix A, then the second step of this phase (computing $r_j$) has a complexity of approximate 0.1 modular multiplication. This is very efficient compared to a standard signature evaluation.

*Remark 5.* The reader can see that every $D_i$ is repeated twice. Doing it in this fashion is to avoid packet loss. We will explain this issue later.

**Verification of streams:**

On receiving the $j$-th block $< (r_j, D_1); (m_1, D_2, D_3); (m_2, D_3, D_4); \ldots; (m_{n-2}, D_{n-1}, D_n); (m_{n-1}, D_n); m_n >$, the receiver carries out the following steps.

– On receiving $(r_j, D_1)$, check that

$$H_{pk}(h(D_1 \| \theta_j); r_j) \stackrel{?}{=} \theta_j$$

– On receiving the $i$-th packet $(m_i, D_{i+1}, D_{i+2})$ $(1 \leq i \leq n-2)$, check that

$$h(m_i \| D_{i+1} \| D_{i+2}) \stackrel{?}{=} D_i$$

– On receiving $(m_{n-1}, D_n)$ , check that

$$h(m_{n-1} \| D_n \| 00 \ldots 0) \stackrel{?}{=} D_{n-1}$$

– On receiving $m_n$ , check that

$$h(m_n \| 00 \ldots 0) \stackrel{?}{=} D_n$$

### 4.2 Performance Analysis

We analyze the performance of our protocol from four aspects:

– **buffering of sender.** The maximum number of packets that need to be stored by the sender in order to compute the authentication information is $n$. Depending on the buffering capability of the sender and the situation of the networks, $n$'s value can be flexible.
– **Computational cost.** This is one of the most important measurements of the performance. We place our emphasis on the online phase since the bottleneck of authenticating real time streams always occurs in transmitting the streams (the computation of offline phase can be carried out any time before the stream transportation). For a block ($n$ packets), the sender needs to do $n$ evaluations of a standard hash function plus 0.1 modular multiplication of two 1024 bit numbers (this is the computational cost of finding collisions of a trapdoor hash function) if we use the trapdoor hash family in Appendix A. For the receiver, $n$ evaluations of a standard hash function and one evaluation of a trapdoor hash function are needed.

**Table 2.** the performance of several authentication protocols. Abbreviations used are: "eva-TH" for an evaluation of a trapdoor hash function; "eva-SH" for an evaluation of a standard hash function; "sign-SS" and "ver-SS" for the signing and verification algorithm of a standard signature scheme respectively. Note that the evaluation of a standard hash function is very efficient, therefore we use an asterisk to remark it. We also note that 1 sign-SS $\gg$ 0.1 modular multiplication.

| Protocols | Online computational cost of the sender | Online communicational cost | Computational cost of the receiver |
|---|---|---|---|
| Wong98[23] Star chaining | 1 sign-SS $n+1$ eva-SH * | $n$ standard sig; $n(n-1)$ hash lengths | 1 ver-SS $n+1$ eva-SH * |
| Wong98[23] Tree chaining | 1 sign-SS $2n-1$ eva-SH * | $n$ standard sig; $n\log_2 n$ hash lengths | 1 ver-SS $2n-1$ eva-SH * |
| Pannetract03[17] EC scheme | 1 sign-SS $n$ eva-SH * 2 coding operations | $\approx 2n$ hash lengths (while $p = 1/4$) | 1 ver-SS $n$ eva-SH * |
| Our protocol | 0.1 multiplication $n$ eva-SH * | $\approx 2n-1$ hash lengths | 1 ver-SS $n$ eva-SH * |

– **Communication overhead.** For the same reason of analyzing the computational cost, we place our emphasis on the online phase. For a block ($n$ packets), the communication overhead (i.e., the additional authentication information embedded in the stream) is $2n - 1$ hash lengths.
– **Tolerance of bursty loss.** Our protocol can tolerate 1 bursty loss of packet, i.e., the rest of packets can also be authenticated even if one packet is lost. In addition, the ability of tolerating packet loss can be strengthened by using more complex hash chain constructions[18, 10].

We compare the performance of several authentication protocols in Table 2 where our protocol uses the trapdoor hash family of Appendix A.

## 5 Conclusion

In this paper, we consider the authentication of digital streams over an insecure network. Using our improved online/offline signature schemes and the hash chain techniques as building blocks, we construct an efficient authentication protocol for digital streams. Compared to previous protocols, our protocol greatly reduces the online computational and communicational cost and thus is more applicable to authenticate real time streams.

## References

1. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Proceedings of the 1st ACM Conference on Computer and Communications Security, pages 62–73, Fairfax, Virginia, November 1993. ACM Press.
2. W. Diffie and M. E. Hellman. Multiuser cryptographic techniques. In Proc. AFIPS 1976 National Computer Conference, pages 109–112, Montvale, N.J., 1976. AFIPS.
3. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inform. Theory, 31:469–472, 1985.
4. Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. In Proc. CRYPTO 89, volume 435 of Lecture Notes in Computer Science, pages 263–277. Springer-Verlag, 1990.
5. Uriel Feige, Amos Fiat, and Adi Shamir. Zero knowledge proofs of identity. In Proc. 19th ACM Symp. on Theory of Computing, pages 210–217, May 1987.
6. A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In Proc. CRYPTO 86, volume 263 of Lecture Notes in Computer Science, pages 186–194. Springer, 1987.
7. R. Gennaro and P. Rohatgi. How to sign digital streams. Lecture Notes in Computer Science, 1294:180–197, 1997.
8. Shafi Goldwasser, Silvio Micali, and Ron L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing, 17(2):281–308, April 1988.
9. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing, 17(2):281–308, April 1988. Special issue on cryptography.

10. 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), pages 13–22, San Diego, CA, February 2001. Internet Society.
11. Hugo Krawczyk and Tal Rabin. Chameleon signatures. In Proceedings of the Symposium on Network and Distributed Systems Security (NDSS '00), pages 143–154, San Diego, CA, February 2000. Internet Society.
12. L. Lamport. Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International, October 1979.
13. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. DSN Progress Report, pages 42–44, 1987.
14. Ralph C. Merkle. A digital signature based on a conventional encryption function. In Proc. CRYPTO 87, volume 293 of Lecture Notes in Computer Science, pages 369–378. Springer-Verlag, 1988.
15. Sara Miner and Jessica Staddon. Graph-based authentication of digital streams. In Proceedings of the IEEE Symposium on Research in Security and Privacy, pages 232–246, Oakland, CA, May 2001. IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Society Press.
16. National Bureau of Standards. Digital signature standard. Technical Report FIPS Publication 186, National Bureau of Standards, 1994.
17. Alain Pannetrat and Refik Molva. Efficient multicast packet authentication. In NDSS, 2003.
18. Adrian Perrig, Ran Canetti, Doug Tygar, and Dawn Song. Efficient authentication and signature of multicast streams over lossy channels. In Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, May 2000. IEEE Computer Society Press.
19. Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Xiaodong Song. Efficient authentication and signing of multicast streams over lossy channels. In IEEE Symposium on Security and Privacy, pages 56–73, 2000.
20. M. O. Rabin. Digitalized signatures. In Foundations of Secure Computation, pages 155–168. Academic Press, 1978.
21. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120–126, 1978.
22. Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In Advances in Cryptology – CRYPTO '2001, volume 2139 of Lecture Notes in Computer Science, pages 355–367. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 2001.
23. Chung Kei Wong and Simon S. Lam. Digital signatures for flows and multicasts. In IEEE ICNP '98, 1998.
24. Yongdong Wu, Di Ma, and Changsheng Xu. Efficient object-based stream authentication. In INDOCRYPT, pages 354–367, 2002.

## A Trapdoor Hash Families

**Definition 2 (Trapdoor hash families).** *A trapdoor hash family is a triple* $(\mathcal{G}, \mathcal{H}, \mathcal{F})$ *such that:*

– $\mathcal{G}$*: key generation algorithm, a PPT algorithm, which, on input* $1^k$ *produces a pair* $(pk, sk)$ *where* $pk$ *is called the public hash key, and* $sk$ *is the corresponding private key.*

- *$\mathcal{H}$ is a family of randomized hash functions. Every hash function in $\mathcal{H}$ is associated with a public hash key pk, takes input of a message from the message space $\mathcal{M}_{pk}$ and a random element from the tag space $\mathcal{R}_{pk}$, and outputs a value in the range $\mathcal{Q}_{pk}$. This hash function is written as $H_{pk}(\cdot;\cdot)$ and has the following properties:*
  1. *Efficiency: Given a public hash key pk and a pair $(m, r) \in \mathcal{M}_{pk} \times \mathcal{R}_{pk}$, $H_{pk}(m; r)$ is computable in polynomial time.*
  2. *Collision resistance: Without private key, any PPT algorithm can not find $(m, r)$and $(m', r')$ such that $m' \neq m$ and $H_{pk}(m'; r') = H_{pk}(m; r)$. A formal description is given in Definition 4.*
- *$\mathcal{F}$: collision-finding algorithm, a PPT algorithm satisfies:*

$$\Pr[H_{pk}(m'; r') = H_{pk}(m; r) : (pk, sk) \overset{\mathrm{R}}{\leftarrow} \mathcal{G}(1^k); (m, r) \overset{\mathrm{R}}{\leftarrow} \mathcal{M}_{pk} \times \mathcal{R}_{pk};$$

$$m'(\neq m) \overset{\mathrm{R}}{\leftarrow} \mathcal{M}_{pk}; r' \overset{\mathrm{R}}{\leftarrow} \mathcal{F}(1^k, sk, m, r, m')] = 1.$$

*Every member in a trapdoor hash family is called a* ***trapdoor hash function****.*

**Definition 3 (Uniform).** *A trapdoor hash family $(\mathcal{G}, \mathcal{H}, \mathcal{F})$ is* ***uniform*** *if and only if whenever the input $(m, r)$ is uniformly distributed in $\mathcal{M}_{pk} \times \mathcal{R}_{pk}$, the output of the collision-finding algorithm $\mathcal{F}$ is uniformly distributed in $\mathcal{R}_{pk}$.*

**Definition 4 (Collision resistance).** *Let $\mathcal{TH} = (\mathcal{G}, \mathcal{H}, \mathcal{F})$ be a trapdoor hash family. The* ***advantage*** *of an adversary $\mathcal{I}$ in breaking $\mathcal{TH}$'s collision resistance is:*

$$\mathrm{Adv}_{\mathcal{I},\mathcal{TH}} = \Pr[m' \neq m \texttt{ and } H_{pk}(m'; r') = H_{pk}(m; r) : (pk, sk) \overset{\mathrm{R}}{\leftarrow} \mathcal{G}(1^k);$$

$$(m, r, m', r') \overset{\mathrm{R}}{\leftarrow} \mathcal{I}(1^k, pk)]$$

*where the probability is taken over the internal coin tosses of the algorithm $\mathcal{G}$ and $\mathcal{I}$.*

*An adversary $\mathcal{I}$ $(t, \epsilon)$-breaks $\mathcal{TH}$'s collision resistance if with running time of at most t, $\mathcal{I}$'s advantage $\mathrm{Adv}_{\mathcal{I},\mathcal{TH}}$ is at least $\epsilon$.*

*A trapdoor hash family $\mathcal{TH}$ is* ***collision resistent*** *if for every PPT adversary $\mathcal{I}$, $\mathrm{Adv}_{\mathcal{I},\mathcal{TH}}$ is negligible.*

**An instantiation of trapdoor hash families[22]:**

- **Setup:** Select at random two safe primes $p, q \in \{0,1\}^{L/2}$ (i.e., primes such that $p' \overset{\mathrm{def}}{=} \frac{p-1}{2}$ and $q' \overset{\mathrm{def}}{=} \frac{q-1}{2}$ are primes) and compute $N = pq$. Choose at random an element $g \in \mathbb{Z}_N^*$ of order $\lambda(N)$ ($\lambda(N) \overset{\mathrm{def}}{=} \mathrm{lcm}(p-1, q-1) = 2p'q'$). The public key is $(N, g)$ and the private trapdoor key is $(p, q)$.
- **The Hash Family:** For $pk = (N, g)$, $H_{pk} : \mathbb{Z}_N \times \mathbb{Z}_{\lambda(N)} \longrightarrow \mathbb{Z}_N^*$ is defined to be $H_{pk}(m; r) \overset{\mathrm{def}}{=} g^{m\|r} \pmod N$ ($m\|r$ denotes the concatenation of $m$ and $r$).

– **Finding trapdoor collisions:** Given $pk = (N, g)$, $sk = (p, q)$, a pair $(m_1, r_1) \in \mathbb{Z}_N \times \mathbb{Z}_{\lambda(N)}$ and an additional message $m_2 \in \mathbb{Z}_N$. We can find $r_2$ such that $g^{m_1 \| r_1} = g^{m_2 \| r_2} \pmod N$ as follows:

$$r_2 = 2^k (m_1 - m_2) + r_1 \pmod{\lambda(N)}.$$

Note that the computational cost of above operation is about one tenth of a single modular multiplication of two 1024 bit numbers.

We refer the reader to[4, 22] for the details of constructions of secure and efficient uniform trapdoor hash families.