Time Valid One-Time Signature for Time-Critical Multicast Data Authentication

Qiyan Wang, Himanshu Khurana, Ying Huang, Klara Nahrstedt
University of Illinois at Urbana-Champaign
Champaign, IL, 61801, United States

***Abstract*—It is challenging to provide authentication to time-critical multicast data, where low end-to-end delay is of crucial importance. Consequently, it requires not only efficient authentication algorithms to minimize computational cost, but also avoidance of buffering packets so that the data can be immediately processed once being presented. Desirable properties for a multicast authentication scheme also include small communication overhead, tolerance to packet loss, and resistance against malicious attacks. In this paper, we propose a novel signature model – Time Valid One-Time Signature (TV-OTS) – to boost the efficiency of regular one-time signature schemes. Based on the TV-OTS model, we design an efficient multicast authentication scheme "TV-HORS" to meet the above needs. TV-HORS combines one-way hash chains with TV-OTS to avoid frequent public key distribution. It provides fast signing/verification and buffering-free data processing, which make it one of the fastest multicast authentication schemes to date in terms of end-to-end computational latency (on the order of microseconds). In addition, TV-HORS has perfect tolerance to packet loss and strong robustness against malicious attacks. The communication overhead of TV-HORS is much smaller than regular OTS schemes, and even smaller than RSA signature. The only drawback of TV-HORS is a relatively large public key of size 8KB to 10KB, depending on parameters.**

I. Introduction

Multicast authentication is a security primitive that enables each receiver in the multicast group to verify if received data originates from the claimed sender and was not altered on the way. In this paper, we focus on multicast authentication of time-critical messages. Our work is motivated by the need for authenticating time-critical multicast data in the power grid, which is one of the largest cyber-physical critical infrastructures and is being transformed today with the design and development of advanced real-time control applications [1]. These applications aim to allow timely control of power flow over physical power networks based on data from monitoring and control devices such as PMUs (Phasor Measurement Units) and relays. One of the goals of such systems is to prevent cascading failures that lead to blackouts [3]. For example, in the NASPI (North American SynchroPhasor Initiative) [2], each PMU senses and multicasts system environment data to control centers at a frequency of 30 samples per second. To provide exact control and in-time abnormality detection, such data often needs to be processed fast with low processing delay (several milliseconds at most). Given the nature of critical decisions based on this data, authentication is essential to prevent adversaries from forcing catastrophic decisions by modifying or forging data. Another example is found in current substation communication systems where critical messages related to transient faults (e.g., caused by lightning) need to be shared in a multicast manner across LANs within 4 ms [4]. Hence, efficient multicast authentication of time-critical data is crucially important to the power grid as well as other similar critical infrastructures. We believe time-critical multicast authentication has wide applicability in many delay sensitive applications.

The nature of time-critical messages implies two basic requirements on the authentication scheme: 1) efficient algorithms to minimize computational cost, and 2) the ability to avoid packet buffering so that the data can be processed instantly upon being available. Additional desirable properties for multicast authentication are 3) small communication overhead, 4) tolerance to packet loss, and 5) resistance to malicious attacks.

In the past decade, researchers proposed multiple approaches to address the multicast authentication problem. However, none of existing schemes can meet all the five requirements simultaneously. The simple method of using public key signatures (like RSA [5]) to sign each message is too computationally expensive to authenticate time-critical messages. Signature amortization based approaches [20]–[27] achieve higher computational efficiency, but suffer from long buffering delay. Although online/offline signature [6] can mitigate the online signature generation cost, it results in a larger signature size and expensive verification. Authentication delay based scheme TESLA [9], [20] features high computational efficiency and low communication overhead, but it requires packet buffering either at the sender or at receivers. Another type of signature, which is much more computationally efficient than regular public key signatures, is the *one-time signature* (OTS) [11]–[13]. OTS is based on nothing more than one-way hash functions and supports buffer-free signing and authentication. However, there exist two drawbacks that prevent OTS from being widely employed in practice: first, the signature size is much larger than common public key signatures; second, the "one-timedness" requires repeated distributions of public keys.

In this paper, we propose a novel signature model "Time Valid One-Time Signature (TV-OTS)" for time-critical multicast data authentication. TV-OTS inherits OTS's advantages of fast message signing/verification and no need for buffering. Furthermore, TV-OTS overcomes the shortcomings of regular OTS schemes, since it has a much smaller signature size and can be extended in an efficient manner to authenticate a large volume of messages. We achieve this by signing only the first l bits of the hash of the message and use time-bounded signature period to prevent signature forgery.

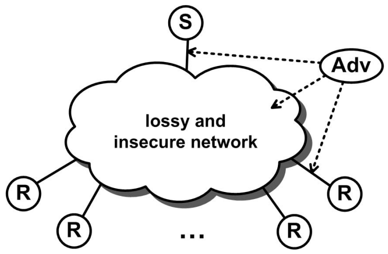


Fig. 1: Network model.

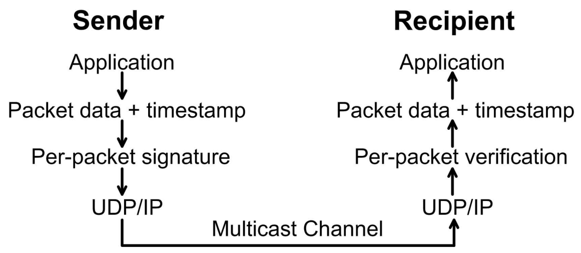


Fig. 2: Data flow of time-critical multicast authentication.

We need loose time synchronization to manage the signature period, but our requirement on synchronization is very weak in that the synchronization error could be as large as several minutes. In application environments such as the power grid such synchronization is relatively easy to achieve.

Based on the TV-OTS model, we design an efficient multicast authentication scheme "TV-HORS (Time Valid Hash to Obtain Random Subsets)" to achieve the five properties stated before. TV-HORS combines one-way hash chains with TV-OTS to allow the signing of large numbers of streaming packets. This approach avoids the need for frequent public key distribution while retaining the properties of TV-OTS. In addition, TV-HORS incurs very short end-to-end computational latency[1] that is on the order of microseconds. To the best of our knowledge, TV-HORS is one of the fastest multicast authentication schemes to date in terms of end-to-end computational latency. Besides, TV-HORS has perfect resistance to packet loss: it can tolerate arbitrary-length burst loss and selective packet loss. At the downside, TV-HORS has a relatively large public key that is 8KB to 10KB depending on parameters.

This paper is organized as follows: Section II introduces the background and models. Section III presents the general TV-OTS model. Section IV gives some security primitives that are important to our scheme construction. Section V describes the TV-HORS scheme. Section VI presents the security analysis of TV-HORS. Section VII gives experimental results and compares our scheme against various classes of multicast authentication schemes. Section IX concludes the paper.

II. Background and Models

A. Network Model

We consider a multicast group involving one sender (S) and a potentially large number of receivers (R). Each message is delivered from S to each R through a lossy and insecure network, such as the Internet (see Figure 1). The intermediate nodes in the network only forward the packets and do not provide any security guarantee (such as integrity and authenticity checks). These nodes may also be malicious and drop or modify S's packets or even inject fake packets. We assume that a multicast tree has been established before the multicast protocol starts. How to establish a multicast tree is orthogonal to this work.

[1]We define end-to-end computational latency as the time duration from when an outgoing message is present at the sender till the receiver has verified the message, *excluding* the transmission latency and queuing delay at intermediate nodes on the way.

We consider a class of applications where 1) each generated message is unknown to S until it is ready to send; 2) the desired end-to-end delay on each packet is short and upper bounded by σ; 3) a sender timestamp is embedded into each packet so that R can check if the received packet has expired; 4) S (resp. R) signs (resp. verifies) the message once it appears; 5) the sending rate at S is dynamic but has an upper bound denoted as λ_m; 6) packets are transmitted using the UDP/IP protocol, and there is no feedback or retransmission mechanism provided. The data flow of the time-critical multicast authentication protocol is shown in Figure 2.

B. Security Goals

Our goal is to achieve the security property that R can verify the fact that received data originates from S and was not altered on the way. In particular, if the received packet was injected or modified by a malicious adversary (Adv), R can recognize and discard it; if the packet is really sent by S and arrives intactly, it can be authenticated by R.

C. Attack Model

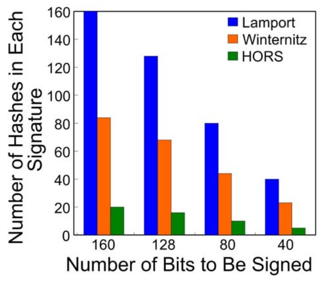
The goal of Adv is to inject a malicious message and convince R that the message was sent by S, or to modify the packet sent by S without being detected. We consider a strong threat model (Figure 1):

- Adv has full control over the network. Adv can selectively eavesdrop, capture, drop, resend, delay, and alter arbitrary packets.
- Adv has access to a fast network with negligible delay.
- The computation power of Adv is limited, but not bounded to that of S or R. Adv can use more powerful devices.
- Adv can compromise arbitrary number of Rs and learn any secrets that R knows.

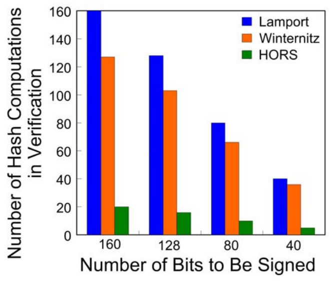
D. Assumption

We assume there exists a trusted key server from which R can securely obtain S's public key. We also assume there exists a secured time synchronization system, which enables R to get loosely synchronized with S [2].

[2]Any time synchronization protocol can be used for our scheme. Refer to [9] for a simple method to securely setup initial synchronization.

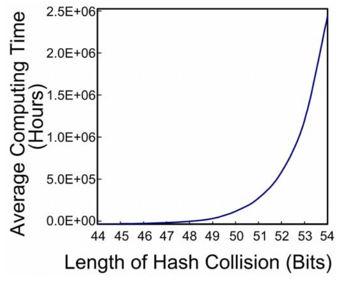


(a) Signature size

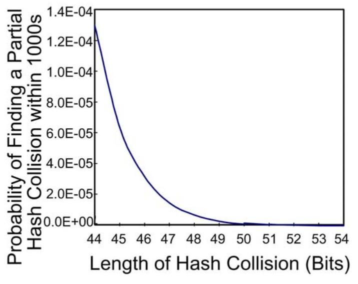


(b) Computation cost

Fig. 3: Signature size and computational cost with different numbers of bits encoded. In Winternitz OTS [12], each chain encodes 2 bits. The public key of HORS [15] is 1024 hashes.



(a) Estimated computing time



(b) Probability

Fig. 4: Average computing time and probability to find a second-preimage partial hash collision with different collision lengths.

III. Time Valid Signature (TV-OTS) Scheme

In this section, we first introduce the basic idea of TV-OTS, and then present the general TV-OTS model. Finally, we briefly explain how to circumvent the one-timed-ness of OTS to sign large numbers of packets.

A. Introduction to TV-OTS

Block-based and authentication delay based multicast authentication schemes [7], [9], [20]–[27] inevitably require packet buffering, which consequently leads to large end-to-end computational delay. In contrast, signing each packet individually using public key signatures [5], [6] can eliminate buffering delay, but suffers from expensive computational cost. The OTS schemes [11]–[13] feature substantially fast signing and verification, but incur a large signature size when signing long messages (such as 160-bit hashes).

We are inspired by the observation that the signature size and computational cost of most OTS schemes (such as [11]–[13], [15]) are mainly determined by the number of bits to be signed (See Figure 3). For example, according to Figure 3(a) the resultant signature size of signing 40 bits is only $\frac{1}{4}$ of that of encoding 160 bits. Intuitively, we improve the efficiency of OTS by signing the first l-bit of the hash of the message. However, this gives rise to the following security problem: Adv can make use of the signature generated by S over some message msg as a valid signature for other (malicious) message msg', as long as msg' is a l-bit partial collision hash of msg, i.e. $H_l(msg') = H_l(msg)$, where $H_l(\cdot) = trunc_l(H(\cdot))$, $trunc_l$ denotes a function that truncates the input to the first l bits, and H is a secure hash function. As a result, Adv can "legally" sign msg' even if he does not know the private key and cannot generate the signature by himself.

So far, the only known way to find a (second-preimage) partial hash collision[3] is brute force. The number of hash computations needed to find a l-bit partial collision is 2^l on average. We can roughly control Adv's calculation time on finding the collision by adjusting the collision length l. Figure 4(a) shows the estimated calculation time with different collision lengths using a workstation level machine [4]. Figure 4(b) shows the probability that a l-bit hash collision can be found within $1000s$.

[3]Finding a l-bit second-preimage hash collision can be formally defined as: given x, find a proper y, s.t. $H_l(y) = H_l(x)$.

The straightforward solution to prevent signature forgery is to carry TV-OTS over reliable transmission channels with bounded end-to-end delay σ, and ensure that the lower bound on Adv's calculation time to find a l-bit second-preimage collision (denoted as T_{Adv}) is larger than σ, i.e. $T_{Adv} > \sigma$. However, this is inapplicable to lossy multicast transmission. We adopt an alternative approach using loose time synchronization, in which each R knows an upper bound on the synchronization error with S. In this model, S specifies a *signature period* in advance, and signs the message *only within* the signature period. To be practical, such a signature period can be made long enough (such as several hours) by adjusting the collision length l. Upon receiving a signed message, R can estimate how long the signature has been exposed at most. If the exposure duration is smaller than T_{Adv}, R is assured that Adv has not obtained a valid signature yet, and R will continue the verification; otherwise, R will discard the message. Now we present this TV-OTS model in detail.

B. TV-OTS Model Description

We suppose R is loosely synchronized with S with maximal synchronization error ϵ. A private/public key pair is associated with a particular signature period $[t_0^S, t_1^S]$, where the superscript indicates that t_0^S (or t_1^S) is S's local starting (or end) time of the signature period. The selection of the signature period is predetermined by S and embedded into S's public key before the protocol starts. S can sign and send a message only within this specific period. S uses some OTS scheme to sign the first l bits of the message digest.

Upon receiving a signed message, R first records the local receiving time t^R and estimates the upper bound on S's local time as $t^R + \epsilon$. Then the longest exposure time of the signature could be $t^R + \epsilon - t_0^S$, provided that S sends the signed message at

[4]Linux system running on 8-core processor (2.80GHZ, 2GB cache, 4GB memory). In the rest of this paper, when we mention a workstation level computer, we are referring to this machine.

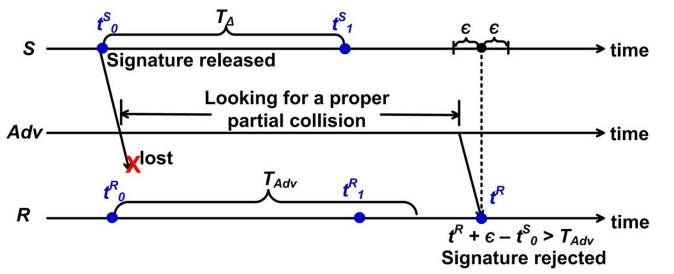


Fig. 5: An example of the general TV-OTS model. S sends the signed message at the very beginning of the signature period (at t_0^S). Adv drops the packet sent by S. The protocol is secure if Adv cannot find a proper message that matches the released signature before the end of the signature period.

the very beginning of the signature period. If $t^R+\epsilon-t_0^S < T_{Adv}$ (recall that T_{Adv} is the lower bound on Adv's calculation time in finding second-preimage partial collisions), R is assured that Adv has not found a matching partial collision, and thus it is safe to continue verifying the signature. Otherwise, R discards it because the message may be injected by Adv (see Figure 5).

The time by which R can accept the message is $t^R = T_{Adv} + t_0^S - \epsilon$. After that time R will stop verifying any message. Therefore, S must sign the message before $T_{Adv} + t_0^S - \epsilon - \sigma$ (recall that σ is the desired upper bound on the end-to-end delay between S and R), i.e. $t_1^S < T_{Adv} + t_0^S - \epsilon - \sigma$. Hence, the duration of the signature period will be

$$T_\triangle = t_1^S - t_0^S < T_{Adv} - \epsilon - \sigma \qquad (1)$$

Because T_{Adv} increases exponentially with l, we can make $T_\triangle$ long enough by simply choosing a relatively large l (refer to Figure 4(a)). Since $T_{Adv} \gg \epsilon$, our TV-OTS schemes can tolerate a very large synchronization error ϵ (e.g. several minutes).

C. Remarks on TV-OTS

The standard OTS schemes sign the whole hash of the message (160 bits when using SHA1), and hence suffer from a large signature size and a large number of hash computations (Figure 3). In contrast, if no bits are signed (i.e. only computing MAC to ensure integrity, but not signing to support authentication), it can achieve extremely high communication and computational efficiency at the cost of security. TV-OTS is in the middle as it signs part of the message digest.

It is important to note that TV-OTS does not compromise any security (which will be shown in Section VI). However, TV-OTS *does* trade off some application flexibility, in that S must sign the message within a predetermined period (although the duration of the period could be very long).

TV-OTS requires very loose time synchronization. Note that in many time-critical applications, communicating peers are well synchronized in order to check the validity of messages and deadlines. Hence, we do not introduce extra burden to such applications.

D. Building a Multicast Authentication Scheme

As we have presented, TV-OTS provides an effective way to reduce the signature size of OTS schemes. Now we introduce the second merit of TV-OTS. That is, under the TV-OTS model we can circumvent the "one-timed-ness" of OTS in a more efficient way, so that TV-OTS can be applied to authenticate a large volume of streaming packets.

Researchers have proposed some advanced techniques [16], [17] to extend standard OTS into multiple-time signatures. These approaches are valuable in a theoretical sense but hard to use in practice due to their substantial overheads and potential security problems. For example, in the case of HORS OTS [15], Pieprzyk et al. [16] constructed a multiple-time signature HORS++ from cover-free families, which introduces large communication and computational overheads. Neumann [17] designed a v-time signature HORSE based on the assumption of reliable transmission. However, this scheme is vulnerable to delay packet attack. A simpler and more efficient approach is reusing the same key to sign v different messages as suggested in [15]. Nevertheless, the multiple exposed signatures leak some information of S's private key and consequently reduce the hardness of forging a signature. The more messages are signed, the more easily Adv can fabricate the signature. Hence, in the standard OTS paradigm, reusing a single key can hardly achieve a reasonably large v while holding a sound security.

Reusing the same key to sign multiple messages is inapplicable for regular OTS schemes, but interestingly it works with our TV-OTS model. In particular, since re-usage of the same key results in shorter calculation time of Adv in finding a proper second-preimage collision, we adjust the parameters of collision length l and signature period's duration $T_\triangle$ to ensure that Adv is still unable to obtain a valid signature within the signature period. That is, Equation (1) still holds and thus the security of TV-OTS can be guaranteed.

Note that although TV-OTS can achieve a larger v than regular OTS schemes, it is not enough to cover the packets transmitted in most applications. For example, in real-time stream transmissions the sender may propagate several millions of packets in an hour. Obviously, re-distributing a new public key for every v packets is too expensive to be realistic. To bridge the gap between v-time signature and streaming packet authentication, we use multiple key pairs but adopt one-way hash chains to link these keys together, thereby avoiding the need for key re-distribution. Each key is an element of the hash chain and assigned to a particular *epoch* (conceptually the same as signature period in the TV-OTS model) to sign v messages at most. There are two merits of combining TV-OTS and hash chains to construct the multicast authentication scheme: 1) any later used keys can be automatically authenticated by earlier verified keys and consequently distribution of a single key is capable of authenticating a potentially large volume of packets; 2) any packet loss can be tolerated since the element of the hash chain can be verified by any previous elements. This combination approach will be elaborated in Section V.

IV. Cryptographic Primitives

Before going into the design of our multicast authentication scheme, we present two security primitives that are basic construction components of our scheme.

A. *One-way Hash Chains*

One-way chains (or hash chains) are an important cryptographic primitive used in many security applications [9], [14]. Given a hash function H and a randomly selected *seed* k_P, one can form a hash chain of length $P+1$ by iteratively computing $k_i = H(k_{i+1})$, $0 \leq i \leq P$. The hash chain is used in the reverse order of generation. Based on a trusted element k_i, one can verify any later used element $k_j (j > i)$ by checking $H^{j-i}(k_j) = k_i$, where H^x denotes iteratively applying H for x times.

As for the storage of a hash chain, one can either create it at once and store each element of the chain, or just store the initial seed s_P and compute other elements on demand. Coppersmith and Jakobson [29] propose a hybrid approach to reduce the storage with a small recomputation penalty: an one-way chain with P elements only requires $\log(P)$ storage and $\log(P)$ computations to access an arbitrary element. If precomputation is allowed, we can avoid online recomputation penalty by precomputing those shortly needed elements offline.

To lower the communication overhead in the transmission of hash chain values, researchers introduce the notion of *light (one-way) chain* [28], which is similar to standard hash chains (named *heavy chains*) but has compact elements.

In practice, light chains are usually used in conjunction with heavy chains. For example, one generator first constructs a heavy chain $\{k_i\}_{0 \leq i \leq P}$ (also called *salt chain* in this case), and then based on the salt chain, derives a light chain as follows: given a random w-bit seed s_P, compute $s_i = h_{k_i}(s_{i+1})$, $0 \leq i < P$, where $h_{k_i}(\cdot) = trunc_w(H(\cdot || k_i))$. The elements at the same location of the heavy chain and the light chain are used in pairs, i.e. (k_i, s_i). To verify (k_j, s_j) based on $(k_i, s_i)(i < j)$, one can recursively compute $H(k_q) = k_{q-1}$ and $h_{k_q}(s_q) = s_{q-1}$ for $q = j$ down to i, and match the last calculated values with (k_i, s_i). The security of the light chain depends on the heavy chain. To forge or invert the light chain is tantamount to breaking the heavy chain.

B. *HORS One-Time Signature*

HORS (Hash to Obtain Random Subsets) [15] is based on a generalization of an earlier OTS scheme by Bos and Chaum [10], which uses a bijection Q to map each message M to a unique t-element subset of a N-element set, T. The message length is bounded to $\log_2 \binom{N}{t}$, T is the private key, the public key is the set created by applying a one-way function to each element of T, and the t-element subset forms the signature.

Reyzin and Reyzin [15] improve this scheme by 1) allowing that the subset of T corresponding to M contains *at most* k elements, and 2) replacing Q with such a function H that it is computationally infeasible to find M_1 and M_2, s.t. $H(M_1) \subseteq H(M_2)$. This relaxation on the mapping function not only promotes the protocol's efficiency, but also suggests a way to sign v messages, provided that it is infeasible to find $M_1, ..., M_v$, s.t. $H(M_v) \subseteq \bigcup_{i=1}^{v-1} H(M_i)$. Reyzin and Reyzin recommend using a strong hash function, SHA-1 for example, to implement H. The detailed protocol is as follows.

- **Key generation**. Generate N random w-bit strings $\{x_1, ..., x_N\}$, which form the private key K_{pri}. The public key is then computed as $K_{pub} = \{y_1, ..., y_N\}$, where $y_i = F(x_i)$ and F is an one-way function.
- **Signing**. To sign a b-bit message M, compute $m = H(M)$, where H is a collision resistant hash function. Split m into t substrings $\{b_1, ..., b_t\}$ of $\log_2 N$ bits each. Interpret each b_j as an integer i_j between 0 and N. The signature of M is formed as $\{x_{i_1}, ..., x_{i_t}\}$.
- **Verification**. To verify a signature $\{x'_{i_1}, ..., x'_{i_t}\}$ over the message M, compute $m' = H(M)$. Split m' into t $(\log_2 N)$-bit substrings $\{b'_1, ..., b'_t\}$. Interpret each b'_j as an integer i'_j between 0 and N. Check if $H(x'_{i_1}) = y_{i_1}, ..., H(x'_{i_t}) = y_{i_t}$ holds.

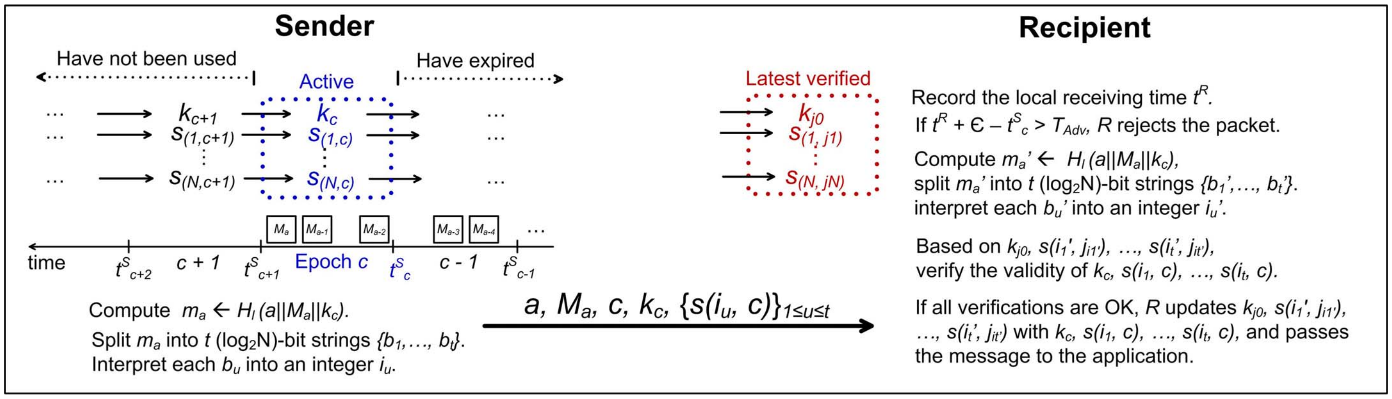
V. TV-HORS Multicast Authentication Scheme

This section presents how to achieve time-critical multicast authentication based on the TV-OTS model. In fact, we can convert most OTS protocols (such as [11]–[13], [15]) into a TV-OTS scheme. Since among these OTS candidates HORS OTS [15] has the smallest signature size, in this work we focus on HORS to construct the multicast authentication scheme.

The basic idea is first applying the TV-OTS model to the HORS OTS to convert it into a time valid v-time signature scheme, and then using one-way hash chains to link multiple key pairs together to enable authentication of a large number of streaming packets. Since the public key (resp. private key) of HORS consists of N values, we need a set of N hash chains to form the keys. In addition, we divide the transmission session (of duration T_ϕ) into a number of epoches (of duration $T_\triangle$). Each epoch is assigned a key pair, and in each epoch S can sign v packets at most using the specific private key. Table I lists the notations used in this scheme.

TABLE I: Notation Table

M_a	The a-th message to be sent
$T_\triangle$	Duration of each epoch
T_ϕ	Duration of the transmission session
λ_m	Upper bound on sending rate (# of packets sent per second)
ϵ	Upper bound on synchronization error
σ	Desired upper bound on end-to-end delay
T_{Adv}	Lower bound on Adv's calculation time at finding a proper second-preimage partial hash collision
t_i^S	S's local starting time of the i-th epoch
k_j	The j-th element in the salt chain
$s_{(i,j)}$	The j-th element in the i-th SAGE chain
P	The length of SAGE/salt chains (also the number of epoches)
L	Security level
l	The number of bits signed in each message digest
t	The number of SAGEs contained in each signature
N	The total number of SAGE chains
v	Maximum number of packets that can be signed in each epoch



Compute $m_a \leftarrow H_l(a||M_a||k_c)$.
Split m_a into t ($\log_2$N)-bit strings $\{b_1, \ldots, b_t\}$.
Interpret each b_u into an integer i_u.

Record the local receiving time t^R.
If $t^R + \epsilon - t^S_c > T_{Adv}$, R rejects the packet.

Compute $m_a' \leftarrow H_l(a||M_a||k_c)$,
split m_a' into t ($\log_2$N)-bit strings $\{b_1', \ldots, b_t'\}$.
interpret each b_u' into an integer i_u'.

Based on k_{j0}, $s(i_1', j_{i1'})$, …, $s(i_t', j_{it'})$,
verify the validity of k_c, $s(i_1, c)$, …, $s(i_t, c)$.

If all verifications are OK, R updates k_{j0}, $s(i_1', j_{i1'})$, …, $s(i_t', j_{it'})$ with k_c, $s(i_1, c)$, …, $s(i_t, c)$, and passes the message to the application.

Fig. 6: TV-HORS multicast authentication scheme.

A. Scheme Construction

*1) **Initialization**:* Suppose a multicast tree has been established. R initializes time synchronization with S, and learns an upper bound ϵ on the synchronization error.

For a transmission with maximal sending rate λ_m, S chooses the length of each epoch $T_\triangle$ satisfying $T_\triangle \le \frac{v}{\lambda_m}$. S estimates the duration of the transmission session T_ϕ, and evenly divides the time line into epoches. Then the total number of epoches is $P = \lceil \frac{T_\phi}{T_\triangle} \rceil$. S constructs a salt chain $\{k_j\}_{0 \le j \le P}$ of length $P+1$. Based on the salt chain, S further constructs N light chains $\{s_{(i,j)}\}_{1 \le i \le N, 0 \le j \le P}$ by computing $s_{(i,j)} = h_{k_j}(s_{i,j+1})$. The elements of light chains are referred to as SAGE (*Signature Authentic Generation Element*). In each epoch j, the SAGEs $s_{(-,j)}$ and the salt k_j are active. As the time advances, the whole row of SAGEs and the salt expire and the next row becomes active (see Figure 6). S only uses active SAGEs to sign messages. Finally, S sends the following initialization information $\langle k_0, \{s_{(u,0)}\}_{1 \le u \le N}, T_\triangle, t^S_0, P \rangle$ to R through an authenticated channel.

If S wants to launch a new transmission session after the current one ends, S needs to construct a new set of SAGE/salt chains and re-distribute the commitment values to each R.

*2) **Signing messages**:* To sign a message M_a, S first computes $m_a = H_l(a||M_a||k_c)$, where $l = \lceil \log_2 \binom{N}{t} \rceil$ and c is the index of the current epoch. Then S splits m_a into t substrings $\{b_1, ..., b_t\}$ of $\log_2 N$ bits each, and interprets each b_u as an integer i_u between 0 and N. The propagated packet is $\langle a, M_a, c, k_c, \{s_{(i_u,c)}\}_{1 \le u \le t} \rangle$.

*3) **Verifying messages**:* Upon receiving a packet, R first records the local receiving time t^R and estimates the upper bound on S's local time as $t^R + \epsilon$. Then R computes $t^S_c = t^S_0 + c \cdot T_\triangle$. If $t^R + \epsilon - t^S_c \ge T_{Adv}$, R discards the packet directly. Otherwise, R computes $m_a' = H_l(a||M_a||k_c)$, splits m_a' into t ($\log_2 N$)-bit substrings $\{b_1', ..., b_t'\}$, and interprets each b_u' as an integer i_u' between 0 and N.

Let $s_{(i_u', j_u)}$, $u \in [1, t]$, denote S's latest verified SAGEs, and k_{j_0} denotes the latest verified salt. Then R checks if the current salt and SAGEs contained in the packet are valid predecessors of k_{j_0} and $s_{(i_u', j_u)}$, $u \in [1, t]$, respectively. If not, R discards the packet. Otherwise, R updates k_{j_0}, $\{s_{(i_u', j_u)}\}_{1 \le u \le t}$ with newly verified k_c, $\{s_{(i_u', c)}\}_{1 \le u \le t}$, and passes the message to the application.

*4) **Bootstrapping a new receiver**:* The new receiver R initializes time synchronization with S. Then S sends to R the following bootstrapping information $\langle i, k_i, \{s_{(u,i)}\}_{1 \le u \le N}, T_\triangle, t^S_i, P - i \rangle$ through an authenticated channel, where i is the index of the *last* epoch.

B. Parameter Selections

We now discuss the parameters used in this scheme. They include the maximum number of packets S can send per epoch 'v', the epoch duration '$T_\triangle$', the number of SAGEs in each signature 't', the number of SAGE chains 'N', and the protocol security level L determined by v, t, N.

(1) *'v' and 'L':* v is an important factor that determines the protocol's security in that the active SAGEs are repeatedly used to sign up to v messages in each epoch. With more exposed SAGEs Adv can have a bigger chance at finding a partial collision. To quantify the hardness for Adv to obtain a valid signature over a new message, we introduce the notion of *security level* L that is defined as follows:

Definition 1: The security level L is such a security parameter that Adv has to compute 2^L hash computations on average to obtain a valid signature for a new message (that is not sent by S) in the case that Adv is able to obtain signatures on v messages of its choice.

The best case for Adv is that any two of these v signatures do not have an overlapping SAGE. In this scenario, the probability that Adv can find a proper collision by one round hash computation is $(vt/N)^t$. Then the estimation of L can be calculated as $L = t \log_2(N/vt)$. Based on L, we can choose T_{Adv} according to Figure 4.

(2) *'$T_\triangle$':* There are two constrains on $T_\triangle$. One is the Equation (1): $T_\triangle < T_{Adv} - \epsilon - \sigma$, and the other is $T_\triangle \le \frac{v}{\lambda_m}$. Note that these two constraints can be met at the same time, when we choose enough small $T_\triangle$.

Our scheme can adapt to bounded dynamic-sending-rate transmissions. For example, $v = 8$ and $T_\triangle = 10ms$, the

TABLE II: Sample parameter selections of TV-HORS.

L	v	t	N	β		
				$P_l = 0.1$	$P_l = 0.2$	$P_l = 0.3$
44	9	11	1584	397	446	509
45	9	11	1688	423	475	542
46	8	11	1598	450	505	577
47	8	11	1702	479	538	614
48	7	11	1586	509	572	653
49	7	12	1424	458	515	588
50	6	12	1294	486	545	622
51	6	12	1370	514	577	659
52	6	12	1452	544	611	698
53	5	12	1282	576	647	739
54	5	12	1358	610	685	782

β – average number of hash computations in verification. P_l – loss rate.

sending rate could be as high as $800pkt/sec$; while for $v = 10$ and $T_\triangle = 5ms$, it can achieve sending rate up to $2000pkt/sec$.

(3) *'t' and 'N' :* Besides the security level L, t and N also determine the computational overhead in verification. To verify a SAGE in the signature, S must iteratively perform hash computations until reaching the latest verified SAGE. For analysis simplicity, we suppose that the probabilities that S sends $0, 1, 2, ..., v$ packets in each epoch are uniformly distributed, and hence S on average sends $\frac{v}{2}$ packets per epoch. Let P_l denote the packet loss rate. Then R receives $r = \frac{v}{2} \cdot P_l$ packets from S per epoch. Let P_{upd} represent the probability that a certain SAGE held by R is updated in one epoch. Then, $P_{upd} = 1 - (1 - \frac{1}{N})^{tr}$. Let D be the variable that represents the distance from the SAGE to be verified to the nearest trusted SAGE in the same SAGE chain. Then,

$$E(D) = \lim_{d' \to \infty} \sum_{d=1}^{d'} d \cdot P_{upd} \cdot (1 - P_{upd})^{d-1} = \frac{1}{1 - (1 - \frac{1}{N})^{tr}}$$

Suppose that R keeps the recently verified salts in memory and does not recompute them when verifying the SAGEs. Then the total number of hash computations needed to verify a signature is $\beta = \frac{t}{1-(1-\frac{1}{N})^{tr}} + 1$. Table II lists some sample values of these parameters.

VI. Security Analysis

In this section, we present a thorough security analysis of the TV-HORS multicast authentication scheme.

A. Brute Force Attack

The simplest attacking strategy Adv can adopt is brute force. In our scheme, Adv can inject a fake message only when it can find a proper second-preimage partial hash collision with computing time less than $T_\triangle + \epsilon + \sigma$ (Equation (1)) that is on the order of several minutes (where $T_\triangle$ is on the order of seconds or milliseconds, ϵ is on the order of minutes, and σ is no more than several seconds). According to Figure 4, we see that for $L = 44$ a workstation level machine needs to continuously compute hash functions for about 90 days on average, and the probability to find a matching second-preimage collision within $16.7min$ is only 0.013%.

One way for Adv to speed up finding a partial hash collision is having multiple machines run in parallel. We assume that Adv is able to have Z machines perfectly collaborate on searching for hash collisions so that the average calculation time could be reduced to $\frac{T_{Adv}}{Z}$. The strategy of defending parallel attack is to select a relatively large l. In the case of $L = 54$, even if Adv has a cluster of 16000 workstation level machines, the average calculation time to find a proper collision is 6.1 days that is still significantly larger than $T_\triangle + \epsilon + \sigma$. [5]

B. Dictionary Attack

There exists an advanced strategy called "dictionary attack" to attack the hash functions with relatively short outputs. In this attack, Adv precomputes a dictionary containing numerous pairs of input and output of the hash function, so that given an output Adv only needs to try a limited number of matching inputs in the dictionary to determine the real preimage.

Our scheme is free from this attack. We use the hash function with short outputs in two cases: one is computing message digests $m_a = H_l(a||M_a||k_c)$, and the other is constructing light chains $s_{(i,j)} = h_{k_j}(s_{(i,j+1)}) = H_w(s_{(i,j+1)}||k_j)$. In both scenarios, a salt value is involved in the computation, and thus Adv cannot start establishing such a dictionary until S releases the salt. On the other hand, the salt is active only within a particular epoch (of duration $T_\triangle$), so the dictionary is helpful to Adv *only if* it can be completed before the salt expires.

C. DoS Attack

DoS attack is a serious threat to amortization based authentication schemes, where Adv floods R with bogus packets to exhaust R's CPU and buffer resources. The TV-HORS scheme is organically resistant to DoS attack. In our scheme, each packet sent by S is self-authentic and independent of any other packets, and R can immediately verify arrival packets without need of buffering. In addition, TV-HORS verification is very fast, which remarkably mitigates the threat of DoS attack.

D. Delay and Drop Packet Attack

For time-critical data transmission, a delay attack can be viewed as a weaker drop packet attack. Without loss of generality, we only analyze the drop packet attack.

A successful drop packet attack means that Adv *selectively* dropping a set of packets $\{P_{i_1}, ..., P_{i_V}\}$ can cause the failure of authentication of another packet $P_j, j \notin \{i_1, ..., i_V\}$. Our scheme is immune to drop packet attack, in that each packet is signed independently and R can authenticate the signature based on any previously verified SAGEs.

E. Replay Attack

Replay attack refers to that Adv intercepts the packets sent by S and replays them some time later. The goal of replay attack is to have R accept the same message twice as different ones or to disorder the packets to mess up the applications. First of all, since each signed message in our scheme is associated

[5] Since in most cases the users are unaware of the computational power of potential adversaries, we recommend to choose a relatively large L for safety.

with a specific epoch, the message signed in epoch i cannot be replayed in epoch j, $j \neq i$. Secondly, if Adv replays M_a in the same epoch, R can either accept M_a if R has not received M_a yet, or reject it if a message with sequence number a has already been verified[6]. Note that since the signing algorithm is applied to $H_l(a||M_a||k_c)$, Adv is unable to modify the sequence number of the packet it intercepted. In all the above cases, the replay attack cannot hurt out scheme.

VII. Evaluation and Comparison

A. Implementation

We implement our scheme on a machine with quad-core processor (2.0 GHZ, 4GB cache, 2GB memory), running a Linux operating system. We use SHA1 as the one-way hash function. The duration of the transmission session is $T_\phi = 1$ hour. The maximal transmission rate is $\lambda_m = 1000 pkt/sec$. We set the packet loss rate as $P_l = 0.2$. We use 48-bit SAGEs and 80-bit salts, which can provide enough security considering their short life time. Table III lists some sample implementation results of the TV-HORS protocol.

TABLE III: Experimental Results of TV-HORS.

L	Communication ovhd (bytes)	ST (us)	VT (us)	ETE comp. delay (us)	KGT. (min)	Pub. key size (KB)
44	80	0.2	90	91	2.1	9.3
46	80	0.2	102	103	2.4	9.4
48	80	0.2	115	116	2.7	9.3
50	86	0.2	109	110	2.6	7.6
52	86	0.2	122	123	2.9	8.5
54	86	0.2	137	138	3.2	8.0

ST – signing times, VT – verification times, ETE – end-to-end, KGT – key generation times.

The results show that TV-HORS possesses very fast signing and verification. TV-HORS's signing only involves one hash computation, and the end-to-end computational delay is on the order of microseconds. Thus TV-HORS can perfectly meet the requirement of low computational latency of delay-sensitive applications.

B. Comparison

We compare our scheme against representative multicast authentication schemes in Table IV. Among these schemes, TV-HORS has the shortest end-to-end computational latency that is calculated as delay(S) + delay(R) + comp.(S) + comp.(R). In addition, TV-HORS incurs fair communication overhead, which is much smaller than traditional OTS schemes [11]–[13] and even smaller than RSA signature [5]. However, TV-HORS introduces a larger key size, which is comparable to BiBa [14]. The requirement of TV-HORS on synchronization is much weaker than other synchronization based authentication schemes [9], [14].

[6]R needs to record the sequence numbers of verified packets for *one* epoch, so the number of records is no larger than v.

VIII. Related Work

A well-known solution for multicast authentication is using public key signature [5] to sign each packet. This approach is straightforward but has expensive computational cost. Online/offline signature [6] reduces the online computational expense by precomputing signatures offline; however its signature verification is still slow. Batch signature based scheme [7] signs packets individually while verifies signatures in batch. This method achieves high verification efficiency, but the signing process is slow and packet buffering is needed at recipients. Amortization based schemes amortize a single signature operation over multiple packets. For example, in erasure code based scheme [24], a signature is computed over a group of packets, and is encoded using an erasure code by introducing some amount of redundancy and splitting the result into n pieces. Each packet carries a signature piece and reconstruction of the signature is possible with any combination of m pieces, where $m < n$. The amortization based schemes achieve low computational costs, but require packet buffering at the sender [19], [21], [23], or at receivers [20], or both [22], [24]–[27]. In comparison, TV-HORS scheme has substantially higher computational efficiency and meanwhile can support packet-based authentication avoiding buffering delay.

Symmetric cryptography can also be utilized to construct multicast authentication schemes. Canetti et al. [8] propose a source authentication protocol using k different symmetric keys to authenticate every message with k different keyed MACs. However, their scheme cannot scale to a large multicast group, since the security depends on the assumption that at most a bounded number of receivers collude. Perrig et al. [9] design an loose synchronization based scheme TESLA to authenticate multicast streams. TESLA has high computational and space efficiency, but requires packet buffering either at the sender or at receivers. OTS (One-Time Signature) [11]–[17] is conceptually similar to regular public key signatures [5], while OTS is constructed merely from one-way functions and consequently possesses significantly higher computational efficiency. OTS is a good candidate to authenticate low-entropy messages [18], but it incurs a large signature size when signing long messages [11]–[13]. Perrig [14] proposes an OTS scheme called BiBa, which is designed for broadcast authentication. It relies on time synchronization and has a reasonable signature size, but its signature generation is slow. To improve BiBa, Reyzin and Reyzin [15] put forward a new OTS scheme HORS, which has both fast signing and verification. Researchers attempt to extend HORS to authenticate multiple messages [16], [17], but these solutions result in either large communication overhead or vulnerability to delay packet attack. TV-HORS has a much smaller signature size and strong security for stream multicast authentication.

IX. Conclusion

In this paper, we proposed a new signature model TV-OTS to facilitate the usage of one-time signatures in constructing multicast authentication schemes. In comparison with regular

TABLE IV: Comparisons of TV-HORS with Representative Multicast Authentication Schemes.

Schemes	Per-pkt comp. costs		Per-pkt comm. overhead	# pkts buffered		Public key size	Need synchron.	Resistance to	
	Sndr	Rcvr		Sndr	Rcvr			chosen loss	DoS attack
Pub. key sig. [5]	$1S$	$1V$	$1s$	1	1	$O(1)$	–	✓	–
Tree based [19]	$\frac{1}{n}S + 2H$	$\frac{1}{n}V + \log_2 nH$	$1s + \log_2 nh$	n	1	$O(1)$	–	–	✓
AugChain [22]	$\frac{1}{n}S + 1H$	$\frac{1}{n}V + 1H$	$\frac{1}{n}s + 2h$	p	n	$O(1)$	–	–	–
EMSS [20]	$\frac{1}{n}S + 1H$	$\frac{1}{n}V + 1H$	$\frac{1}{n}s + dh$	1	n	$O(1)$	–	–	–
Graph based [21]	$\frac{1}{n}S + 1H$	$\frac{1}{n}V + 1H$	$1s + O(n)h$	n	1	$O(1)$	–	–	–
Erasure code [24]	$\frac{1}{n}S + 1H$ $+1EC$	$\frac{1}{n}V + 1H$ $+1ED$	$\frac{1}{n}s + \frac{n}{m}h$	n	m	$O(1)$	–	✓	–
Batch RSA [7]	$1S$	$\frac{1}{n}V + 2M$	$1s$	1	n	$O(1)$	–	✓	✓
Online/offline [6]	$0.1M$	$2V$	$2s$	1	1	$O(1)$	–	✓	–
TESLA [9]	$1H$	GH	$2h$	1RTT *	1	$O(1)$	loose	✓	✓
BiBa [14]	NH	$(k + kD)H$	kh	1	1	$O(N)$	✓	✓	✓
Ours	$1H$	βH	th	1	1	$O(N)$	very loose	✓	✓

S – public key signature, V – public key verification, H – hash, n – block size, s – public key signature size, h – hash size, EC – erasure coding, ED – erasure decoding, M – modular multiplication. *All packets sent in one round trip time are buffered.

Parameters: AugChain [22] uses the $C_{a,p}$ configuration. In EMSS [20] each packet piggybacks d hashes of previous packets. In erasure code [24], the (n, m) erasure code is used. In TESLA [9], one key chain is used, 1 RTT (between the sender and the receiver) is chosen as the authentication delay (e.g. 2s), and packet buffering is placed at the sender to resist DoS attack. G is the average distance between the current released key and the latest verified key. In BiBa [14], each signature contains k SEALs. D is the average distance between the current SEAL and the latest verified SEAL in the same SEAL chain. β is the average number of hash computations in TV-HORS verification. See Table II for sample values of β.

one-time signatures, TV-OTS has a much smaller signature size and can be extended to v-time signatures more efficiently. Based on the TV-OTS model, we designed a time-critical multicast authentication scheme TV-HORS, which combines hash chains with TV-OTS to authenticate streaming packets. TV-HORS provides short end-to-end computational latency, perfect tolerance to packet loss, and strong resistance against malicious attacks. The only drawback is its relatively large key size. We believe that this problem can be mitigated by fairly trading off some tolerance to packet loss, which is interesting aspect to explore for our future work.

X. Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0524695. Authors would like to thank anonymous reviewers for the invaluable suggestions to improve this paper.

References

[1] North American SynchroPhasor Initiative (NASPI). http://www.naspi.org/
[2] R. Hasan and R. Bobba and H. Khurana, *Analyzing NASPInet data flows*, PSCE'06, 2006.
[3] U.S. -Canada Power System Outage Task Force, *Final report on the august 14, 2003 blackout in the united states and canada: Causes and recommendations*, April 2004.
[4] *IEEE Standard Communication Delivery Time Performance Requirements for Electric Power Substation Automation* Inst. Electr. & Electron. Eng., New York, NY, USA, IEEE Std. 1646-2004, pp. 1-24, 2004.
[5] R. L. Rivest and A. Shamir and L. M. Adleman, *Method for obtaining digital signatures and public-key cryptosystems*, Commun. ACM, 21(2), pp. 120-126, 1978.
[6] A. Shamir and Y. Tauman, *Improved online/offline signature schemes*, CRYPTO'01, pp. 355-367, 2001.
[7] Y. Zhou and Y. Fang, *Multicast authentication over lossy channels*, MILCOM'07, 2007.
[8] R. Canetti and J. Garay and G. Itkis and D. Micciancio and M. Naor and B. Pinkas, *Multicast Security: A taxonomy and some efficient constructions*, INFOCOM'99, 1999.
[9] A. Perrig and R. Canetti and D. Song, and J. D. Tygar, *Efficient and secure source authentication for multicast*, NDSS'01, 2001.
[10] J. N. E. Bos and D. Chaum, *Provably unforgeable signatures*, CRYPTO'92, LNCS 740, pp. 1-14, 1992.
[11] L. Lamport, *Constructing digital signatures from one-way function*, Technical Report SRI-CSL-98, SRI International Computer Lab, 1979.
[12] R. C. Merkle, *A certified digital signature*, CRYPTO'89, 1989.
[13] D. Bleichenbacher and U. M. Maurer, *Directed acyclic graphs, one-way functions and digital signatures*, CRYPTO'94, pp. 75-82, 1994.
[14] A. Perrig, *The BiBa one-time signature and broadcast authentication protocol*, CCS'01, Nov. 2001.
[15] L. Reyzin and N. Reyzin, *Better than BiBa: short one-time signatures with fast signing and verifying*, In Information Security and Privacy (ACISP'02).
[16] J. Pieprzyk and H. Wang and C. Xing, *Multiple-time signature Schemes against Adaptive Chosen Message Attacks*, SAC'03, 2003.
[17] W. D. Neumann, *HORSE: an extension of an r-time signature scheme with fast signing and verification*, ITCC'04, 2004.
[18] M. Luk and A. Perrig and B. Whillock, *Seven cardinal properties of sensor network broadcast authentication*, SASN'06, Oct. 2006.
[19] C. Wong and S. Lam, *Digital signatures for flows and multicasts*, ICNP '98, 1998.
[20] A. Perrig and R. Canetti and J. D. Tygar and D. Song, *Efficient authentication and signing of multicast streams over lossy channels*, In Proceedings of IEEE Symposium on Security & Privacy, pp. 56-73, 2000.
[21] S. Miner and J. Staddon, *Graph-based authentication of digital streams*, IEEE Security & Privacy, 2001.
[22] P. Golle and N. Modadugu, *Authenticating streamed data in the presence of random packet loss*, NDSS'01, 2001.
[23] D. Song and D. Zuckerman and J. D. Tygar, *Expander graphs for digital stream authentication and robust overlay networks*, In Proceedings of IEEE Symposium on Security & Privacy, pp. 258-270, May 2002.
[24] J. M. Park and E. K. P. Chong and H. J. Siegel, *Efficient multicast packet authentication using signature amortization*, In Proceeding of IEEE Symposium on Security & Privacy, pp. 227-240, 2002.
[25] A. Pannetrat and R. Molva, *Efficient multicast packet authentication*, NDSS'03, 2003.
[26] A. Lysyanskaya and R. Tamassia and N. Triandopoulos, *Multicast Authentication in Fully Adversarial Networks*, In Proceeding of IEEE Symposium on Security & Privacy, pp. 241- 255, 2004.
[27] Chris Karlof and Naveen Sastry and Yaping Li and Adrian Perrig and J. D. Tygar, *Distillation codes and applications to DoS resistant multicast authentication*, NDSS'04, pp. 37-56, 2004.
[28] Yih-Chun Hu and Markus Jakobsson and Adrian Perrig, *Efficient Constructions for One-Way Hash Chains*, ANCS'05, July 2005.
[29] D. Coppersmith and M. Jakobsson, *Almost optimal hash sequence traversal*, In Proc. of the 4th Conf. on Financail Cryptography (FC'02), 2002.