A Survey on Various Stream Authentication Techniques

Jini Jacob
M. Tech Student
Department of Computer Science and Engineering
SCT College of Engineering
Thiruvananthapuram, India

Sri. Subu Surendran
Associate Professor
Department of Computer Science and Engineering
SCT College of Engineering
Thiruvananthapuram, India

***Abstract*—Distribution of contents through distributed networking technologies is involved in many web-based services. Unfortunately, these modern distributed systems which are designed to distribute contents to a large group of users, also provide a platform for large number of adversarial users. Authenticating the content is the main way to prevent these attacks and protecting the content is the main responsibility of content providers. So for that, several techniques for stream authentication which aim at reducing the computation and communication overhead have been proposed. These techniques are associated with protecting individual blocks which comprise a stream in communication. These techniques can be divided into MAC-based schemes like TESLA and signature based schemes like EMSS, AC, SAIDA, and WL. TESLA method is efficient against data loss. But it requires time synchronization between a signer and a verifier, large buffers of unverified blocks until the verification key is received and storage of long key chains. Signature-based techniques either depend on amortizing a single signature over multiple blocks or designing extremely high speed signature schemes like k-time signatures, BiBa, HORS and TV-OTS to sign each block to reduce the per block overhead. In this paper, a comparison of various stream authentication techniques is performed.**

Keywords—authentication; signature generation; signature verification; keys; hash value

I. INTRODUCTION

Distribution of contents through distributed networking technologies is involved in many web-based services. Unfortunately, these modern distributed systems which are designed to distribute contents to a large group of users, also provide a platform for large number of adversarial users. Adversaries can impersonate as legitimate content providers to distribute malicious content possibly infected with viruses, worms, etc. Adversaries can also place themselves in the content distribution paths, for example, by compromising web caches, and modify the content in ways that can potentially harm client devices. So authenticating the content plays a crucial role in preventing these attacks.

Protecting content is the responsibility of content providers. They often provide highly personalized user experience by inserting targeted advertisements and dynamically generated contents. Today majority of mass-viewed content is dynamically generated and rich in multimedia like combination of text, audio, animation, still images, video, and interactive content forms which are gathered from a large number of sources, assembled and presented to the user. In such cases, malicious modification of content by malicious sources becomes a legitimate threat.

For the conventional message authentication, it requires that the sender and the receiver have the ability to store the entire message before processing the message. However, in most cases of distributing content, the content provider transmits the content in the form of digital streams that receivers consume at the stream arrival rate without large delay. To protect such delay-sensitive digital streams against malicious attacks, security mechanisms must proficiently process long sequence of bits in a manner that allows receivers to verify the authenticity of the stream in portions without excessive processing delays associated with each portion of the stream. This is done by dividing the stream into blocks or chunks and using an efficient security mechanism to secure each block of data.

For that, several techniques for stream authentication which aim at reducing the computation and communication overhead have been proposed. These techniques are associated with protecting individual blocks which comprise a stream in communication and they can be divided into two: MAC-based schemes like TESLA and signature based schemes like EMSS, AC, SAIDA, and WL. TESLA method is efficient against data loss. But it requires time synchronization between a signer and a verifier, large buffers of unverified blocks until the verification key is received and storage of long key chains. Signature-based techniques either depend on amortizing a single signature over multiple blocks or designing extremely high speed signature schemes like k-time signatures, BiBa, HORS and TV-OTS to sign each block to reduce the per block overhead.

II. VARIOUS STREAM AUTHENTICATION TECHNIQUES

Various stream authentication techniques include TESLA, BiBa, HORS, TV-HORS and trapdoor hash based mechanism. TESLA has high computational and space efficiency. But it requires packet buffering either at the sender or at receivers. An OTS (One-Time Signature) scheme

called BiBa is designed for broadcast authentication. OTS is similar to regular public key signatures, while it is constructed from one-way functions. So it has higher computational efficiency. It relies on time synchronization and has a reasonable signature size. But its signature generation is slow. To improve BiBa, a new OTS scheme called HORS is developed, which has both fast signing and verification. Researchers attempt to extend HORS to authenticate multiple messages. 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. Trapdoor hash based mechanism uses signature amortization technique based on trapdoor hash functions for authenticating individual data blocks in a stream.

A. TESLA

In the case of one sender and one receiver, there is no problem of continuous stream authentication since it is solved through standard mechanisms. The sender and receiver agree on a secret key which is used along with a message authenticating code (MAC) to ensure authenticity of each packet. But in the case of multiple receivers, the problem becomes much tougher to solve, because a symmetric approach would allow any receiver holding a key to forge packets. Also, the sender can use digital signatures to sign every packet with its private key. This solution provides proper authentication, but digital signatures are very inefficient. As the real-time data streams are lossy, the security problems are even harder. With many receivers, the high packet loss is for the receivers with relatively low bandwidth. Along with, the data authenticity should be assured even in the presence of this high packet loss.

Timed Efficient Stream Loss-tolerant Authentication [5] uses only symmetric cryptographic primitives such as pseudorandom functions (PRFs) and message authentication codes (MACs). It is based on timed release of keys by the sender. The scheme is based on this idea: The sender first commits to a random key k without revealing it and transmits it to the receivers. The sender then attaches a message authenticating code to the next packet P_i and uses the key k as the MAC key. In a later packet P_{i+1}, the sender decommits to k, which allows the receivers to verify the commitment and the MAC of packet P_i. If both verifications are correct, and if it is guaranteed that packet P_{i+1} was not sent before packet P_i was received, then a receiver knows that the packet P_i is authentic. To start this scheme, the sender uses a regular signature scheme to sign the initial commitment. All subsequent packets are authenticated through chaining.

TESLA has the properties like low computation overhead, low per-packet communication overhead, arbitrary packet loss tolerated, unidirectional data flow, freshness of data, no sender side buffering, and high guarantee of authenticity. There are five schemes for stream authentication. Each scheme builds up on the previous schemes and tries to solve its limitations. The final scheme, which is scheme V, is known as the TESLA.

All five schemes begin with an initial synchronization protocol where each receiver compares its local time with that of the sender, and registers the difference. All that the receiver needs is a value □ such that the sender's clock is no more than □ time-units ahead of the receiver's clock. The local internal clocks of the sender and recipient do not drift too much during a session [5]. So the basic assumption that underlies the security of this scheme is that the local internal clocks of the sender and recipient do not drift too much during a session.

1) Scheme I: The Basic Scheme

The sender commits to a random key k without revealing it and transmits it to the receivers. The sender then attaches a message authenticating code to the next packet P_i and uses the key k as the MAC key. In a later packet P_{i+1}, the sender decommits to k, which allows the receivers to verify the commitment and the MAC of packet P_i. If both verifications are correct, and if it is guaranteed that packet P_{i+1} was not sent before packet P_i was received, then a receiver knows that the packet P_i is authentic [5].

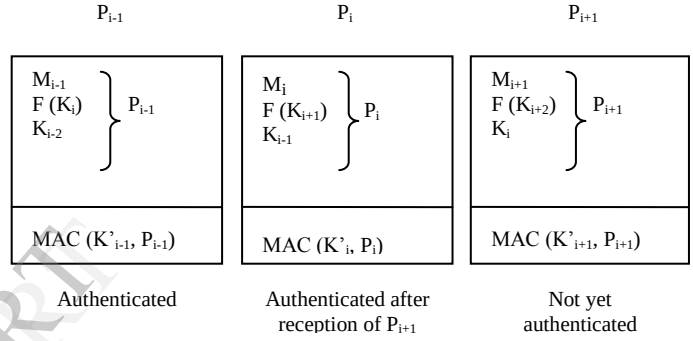

Fig. 1: Basic Stream Authentication Scheme

In the fig. 1., to send the message M_i, the sender picks a fresh random key K_{i+1} and then constructs the following packet $P_i = <D_i, MAC(K_i', D_i)>$, where $D_i = < M_i, F(K_{i+1}), K_{i-1}>$ and the $MAC(K_i', D_i)$ computes a message authenticating code of D_i under key K_i'. When the receiver receives packet P_i, it cannot verify the MAC instantly, since it does not know K_i and cannot reconstruct K_i'. Packet $P_{i+1} = <D_{i+1}, MAC(K'_{i+1}, D_{i+1})>$ (where $D_{i+1} = <M_{i+1}, F(K_{i+2}), K_i>$) discloses K_i and allows the receiver first to verify that K_i is correct ($F(K_i)$ equals the commitment which was sent in packet P_{i-1}); and second to compute $K_i' = F'(K_i)$ and check the authenticity of packet P_i by verifying the MAC of packet P_i [5]. After the receiver has authenticated P_i, the commitment $F(K_{i+1})$ is also authenticated and the receiver repeats this scheme to authenticate P_{i+1} after P_{i+2} is received and so on. To start this scheme, the first packet needs to be authenticated with a regular digital signature scheme, for example RSA [9].

A data packet P_i arrived safely, if the receiver can unambiguously decide, based on its synchronized time and □t, that the sender did not yet send out the corresponding key disclosure packet P_j [5].

2) Scheme II: Tolerating Packet Loss

To authenticate lossy multimedia streams, tolerating packet loss is very important. The solution for this lossy multimedia streams is to generate a sequence of keys $\{K_i\}$ through a sequence generated through pseudo-random

function applications. An extra advantage is that the key commitment does not need to be embedded in each packet. Due to the intractability of inverting the pseudo-random function, any value of the chain is a commitment for the entire chain. So the commitment in the initial authenticated packet is a sufficient.

3) Scheme III: Achieving Fast Transfer Rates

The receiver needs to be assured that it receives the packet P_i before the key disclosure packet P_{i+1} is sent by the sender. This condition limits the transmission rate of the previous two schemes since P_{i+1} can only be sent after every receiver has received P_i. This problem is solved by disclosing the key K_i of the data packet P_i in a later packet P_{i+d}, instead of in the immediately following packet, where d is a delay parameter that is set by the sender and announced as the session set-up.

4) Scheme IV: Dealing with Dynamic Packet Rates

A fixed or predictable sender schedule is used in the previous schemes. So the exact sending time of each packet is known to each recipient. A scheme which allows senders to send at dynamic transmission rates is designed since the previous scheme restricts the flexibility of senders. So every receiver does not need to know about the correct sending schedule of each packet. The solution to this problem is to pick the MAC key and the disclosed key in each packet only on a time interval basis instead of on a packet index basis. The sender uses the same key K_i to compute the MAC for all packets which are sent in the same interval i. All packets sent in interval i disclose the key K_{i-d}.

5) Scheme V: Accommodate a Broad Spectrum of Receivers

There was a tradeoff in the choice of the key disclosure period for the previous schemes. If the time difference is short, the packet can be authenticated quickly. But if the packet travel time is long, then the security condition will not hold for remote receivers that forces them to drop the packet. Conversely, a long time period will be comfortable to remote receivers. But the authentication time delay may not be acceptable for receivers with fast network access. Since the scheme needs to scale to a large number of receivers, the receivers to have a wide variety of network access are expected. This approach is to use multiple authentication chains with different disclosure periods simultaneously. Each receiver can then use the chain with the minimal disclosure delay which is sufficient to prevent spurious drops which are caused if the security condition does not hold.

B. BiBa

BiBa [4] uses one-way functions without trapdoors. Its features are a low verification overhead and a relatively small signature size. But the public keys are larger and the time to generate signatures is also higher. In contrast to TESLA, the authentication in BiBa is instant and does not depend on the time synchronization error. The security of the BiBa signature comes from the security of finding k-way collisions for a one-way function. BiBa has exponentially increasing security such that it is secure even if the signer only has modest computation resources. BiBa also provides a more compact signature and it is faster to verify than the previous schemes.

BiBa stands for Bins and Balls signature – a collision of balls under a hash function in bins forms the signature [4]. BiBa exploits the birthday paradox such that the signer has many balls to throw into the bins those results in a high probability to find a signature. But an adversary has few balls so it has a low probability to forge a signature.

The signer precomputes values that it subsequently uses to generate BiBa signature. These values are random numbers generated in a way that the receivers can instantly authenticate them with the public key. These precomputed values are known as SEALs which is SElf Authenticating vaLues. The property that is needed for SEALs is that the verifier can efficiently authenticate the SEAL based on the public key, and that is computationally infeasible for an adversary to find a valid SEAL for a given public key. The simplest approach is to use the PRF F as a commitment scheme. For a given SEALs, the public key is $f_s=F_s(0)$. If the verifier learns f_s in an authentic fashion, it can easily authenticate s by verifying $F_s(0) = f_s$. In BiBa, the signer needs multiple SEALs, so a public key could consist of multiple commitments [4]. Another alternative for SEAL authentication is a Merkle hash tree [10]. So the SEALs would be the leaf nodes of the tree and the public key is the root node of the tree.

To sign a message m, the signer first computes the hash h=H (m). The signer then computes the hash function G_h to all the SEALs $s_1,\ldots,s_t$. The signer looks for a two-way collision of two SEALs: $G_h(s_i) = G_h(s_j)$, with $s_i \neq s_j$. The pair (s_i,s_j) forms the signature [4].

The verifier receives message M and the BiBa signature $<s_i,s_j>$. To verify the BiBa signature, the verifier computes h=H (m), checks that $s_i \neq s_j$, and $G_h(s_i) = G_h(s_j)$[4]. The verification is very simple. Without considering the SEAL authentication, it only requires one hash function computation.

C. HORS

HORS [3] is a one-time signature scheme with very efficient signing and verifying, and short signatures. It is well-suited for broadcast authentication, and, an improvement of the BiBa one-time signature. HORS is a simple one-time signature scheme which maintains BiBa's advantages and removes its main disadvantage. This signature scheme can be used r times, instead of just once, for all small values of r. In both schemes, security decreases as r increases. In HORS, signing can be done by just one hash function evaluation, and verifying can be done by 17 hash function evaluations for a high level of security.

A cryptographic hash function Hash is proposed to construct H as: split the output of the hash function into *k* substrings of length log t each; interpret each (log *t)*-bit substring as integer written in binary; combine these integers to form the subset of T of size at most *k.* This construction of H results in the scheme called HORS (Hash to Obtain Random Subset) [3]. HORS has very efficient signing. It needs just one evaluation of the cryptographic hash function. Also it has very efficient verifying. It needs just one

evaluation of the hash function in addition to k evaluations of the one-way function *f*.

The security proof for this scheme follows directly from subset-resilience of the hash function and one-wayness of *f*. The proof reduces a signature forgery. The HORS one-time signature scheme is given in fig. 2.

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

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

Verifying
Input: Message m, signature $\sigma = (s'_1,s'_2,\ldots,s'_k)$, and public key PK = $(k,v_1,v_2,\ldots,v_t)$
Let h = Hash (m)
Split into k substrings $h_1,h_2,\ldots,h_k$, of length $\log_2 t$ bits each
Interpret each h_j as an integer i_j for $1 \le j \le k$
Output: "accept" if for each j, $1 \le j \le k$, $f(s'_j) = v_{ij}$; "reject" otherwise

Fig. 2: HORS One-Time Signature Scheme

D. TV-HORS

TV-HORS [2] requires not only the efficient authentication algorithms to minimize computational cost, but also the avoidance of buffering packets. The properties for a multicast authentication scheme include tolerance to packet loss, small communication overhead, and resistance against malicious attacks. Time Valid One-Time Signature (TV-OTS) is a signature model to boost the efficiency of regular one-time signature schemes. TV-HORS combines one-way hash chains with TV-OTS to avoid frequent public key distribution. It provides fast signing and verification, buffering free data processing, perfect tolerance to packet loss, strong robustness against malicious attacks, and smaller communication overhead [2].

Using public key signatures to sign each message is too expensive to authenticate time-critical messages. Signature amortization based approaches achieve higher computational efficiency. But they suffer from long buffering delay. Even if online or offline signature can mitigate the online signature generation cost, it results in a larger signature size and expensive verification.

The TV-HORS multicast authentication scheme is given in fig. 3. The basic idea is first applying the TV-OTS model to the HORS OTS [11] to convert it into a time valid υ-time signature scheme. Then use one-way hash chains to link multiple key pairs together. Since the public key of HORS consists of N values, a set of N hash chains to form the keys is needed. Divide the transmission session (of duration T_φ) into a number of epoches (of duration T_Δ). Each epoch is assigned a key pair, and in each epoch S can sign υ packets at most using the specific private key.

The total number of epoches is $P = T_\varphi/T_\Delta$. S constructs a salt chain $\{k_j\}_{0\le j\le P}$ of length P+1. S further constructs N light chains $\{s_{(i,j)}\}_{1\le i\le N,\ 0\le j\le P}$ by computing $s_{(i,j)}=h_{kj}(s_{i,j+1})$. The elements of light chains are referred to as SAGE (Signature Authentic Generation Element) [2].

Sender
Compute $m_a \leftarrow H_1(a\|M_a\|K_c)$.
Split m_a into t $(\log_2 N)$-bit strings $\{b_1,\ldots,b_t\}$.
Interpret each b_u into an integer i_u.

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

Compute $m_a' \leftarrow H_1(a\|M_a\|K_c)$.
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}, $s(i_1', j_{i1'}),\ldots,s(i_t', j_{it'})$,
Verify the validity of k_c, $s(i_1, c),\ldots,s(i_t, c)$.

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

Fig. 3: TV-HORS Multicast Authentication Scheme

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

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 $t^S_c = t^S_0 + cT_\Delta$. If $t^R + \epsilon - t^S_c \ge T_{Adv}$, R discards the packet directly [2].

The new receiver R initializes time synchronization with S. Then S sends to R the following bootstrapping information through an authenticated channel [2].

E. Trapdoor Hash-Based Mechanism

Authentication of delay-sensitive streams requires high verification rates. For real-time generated digital streams, a block must be signed by a sender as soon as it is generated with minimal computational overhead. Stream transmission is typically done using unreliable transport protocols like UDP to provide a high throughput. Authenticating information such as signatures and hash values must be limited to a small, constant size. This technique tolerates out-of-order arrival of blocks in the stream and is resilient to transmission losses. This technique minimizes transmitting delays in a stream following the block-signing process and playback of the stream following the block-verification process.

This technique limits communication overhead while sending the authenticating material to a small, constant-sized signature. The unique capability to find collisions between hashes of different messages are provided by trapdoor hash functions using a secret trapdoor key. In signature amortization technique, a signature is computed and amortized it over multiple blocks by finding trapdoor collisions with the hash of the signed block. The authenticity of a block is verified by a receiver by computing its trapdoor hash value and comparing it with the hash value of any other block in the stream.

This signature amortization technique [1] can be divided into two phases: Stream signing and stream verification. The sender generates a signature σ using SK on the trapdoor hash $h_0 = TH_{HK0}(_{m0,r0})$ of the contents m_0 of first block p_0 in the stream. The first block p_0 contains $<m_0, r_0,\sigma>$ [1]. To sign subsequent blocks p_i ($i \geq 1$) with content m_i, the signer generates a collision parameter r_i such that $TH_{HKi-1}(_{mi-1,ri-1}) = TH_{HKi}(_{mi,ri})$ using trapdoor keys TK_i and Tk_{i-1}[1]. When the receiver obtains the initial block p_0 of the stream, it extracts $<m_0, r_0,\sigma>$ from the block, computes $h_0 = TH_{HK0}(_{m0,r0})$ and verifies the signature σ on h_0 under PK [1]. For each subsequent block, p_i ($i \geq 1$) in the stream, the received block is parsed by the receiver as $<m_i, r_i>$ and computes the trapdoor hash of mi as $h_i=TH_{HKi}(_{mi,ri})$. It then checks whether h_i matches the trapdoor hash $h_j=TH_{HKj}(_{mj,rj})$ of the contents m_j of an arbitrary block p_j that was received prior to p_i [1].

III. CONCLUSION

Mechanisms of flow authentication in content distribution networks help prevent masquerading attacks and malicious modification of content during transmission. However, efficient authentication of live, on-demand content is a challenging task, and requires fast signing and verification, tolerance against transmission loss and small per-block communication overhead.

TESLA, short for Timed Efficient Stream Loss-tolerant Authentication, offers strong loss robustness, sender authentication, minimal overhead, and high scalability, at the cost of loose initial time synchronization and slightly delayed authentication. TESLA does not provide non-repudiation. Since most multimedia applications discard the data after it is decoded and played, they do not need non-repudiation. Stream signature schemes are very important, due to two cases. First, some applications really need continuous non-repudiation of each data packet. Second, in settings where time synchronization is difficult, TESLA might not work. TESLA has high computational and space efficiency, but requires packet buffering either at the sender or at receivers.

BiBa signature scheme is a new signature construction which uses one-way functions without trapdoors. It features small signature size and has a low verification overhead. In comparison to other one-way function based signature schemes, BiBa has smaller signatures and is atleast twice as fast to verify. On the downside, the BiBa has very large public key, and the signature generation overhead is higher than the previous schemes based on one-way functions without trapdoors. The birthday paradox is used by the BiBa signature scheme to construct a digital signature scheme from a one-way function without a trapdoor. The BiBa one-time signature is very useful in settings where the signer can send the public key to the verifier efficiently, and where the verifier is constrained on computational power. Along with, the BiBa signature generation has a linear speedup on multiprocessor systems until signature generation and verification only require two sequential hash function evaluations.

In HORS, verifying is as fast as in BiBa, and signing is faster than verifying. The key and signature sizes are slightly improved. Also it has short signatures. Like BiBa, this signature scheme can be used r times for small values of r, instead of just once, and in both schemes, security decreases when r increases. The security of this scheme relies only on complexity-theoretic assumptions, and it does not require the use of random oracles. BiBa requires about 2t calls to the random oracle, while, in contrast, HORS requires only one call to H for signing messages. Verification in BiBa requires k calls to the one-way function f, just like in HORS. BiBa verification also requires k calls to the random oracle, while HORS requires only one call to H. Thus, HORS scheme is significantly more efficient in signing and more efficient in verifying. Another advantage of HORS is that slightly smaller values of t and k can be used to achieve the same security levels. HORS is more secure against an attack than BiBa is. The various disadvantages of HORS are its variable communication overhead for each block and the inability to support authentication of multiple simultaneous streams. Along with, it has a little sender side delay.

TABLE I. COMPARISON OF VARIOUS STREAM AUTHENTICATION TECHNIQUES

Schemes	TESLA	BiBa	HORS	TV-HORS	Trapdoor Hash-Based Mechanism
Per-pkt computation costs at Sender	1H	NH	1H	1H	0.03x + 1H
Per-pkt computation costs at Receiver	GH	(k+kD)H	1H	βH	2.06x + 1H
Per-pkt comm. overhead	2h	kh	2h	th	2h
Pkts buffered at Sender	1RTT	1	1	1	1
Pkts buffered at Receiver	1	1	1	1	1
Public key size	O(1)	O(N)	O(N)	O(N)	O(N)
Need synchron.	Loose	Yes	Yes	Very loose	Loose
Resistance to chosen lose	Yes	Yes	Yes	Yes	Yes
Resistance to DoS attack	Yes	Yes	Yes	Yes	Yes

In the table 1, H is the hash and h is the hash size. In TESLA, one key chain is used, 1 RTT between the sender and the receiver is chosen as the authentication delay, 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, 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. In trapdoor hash-based mechanism, x denotes the Schnorr exponentiation.

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). Along with, TV-HORS incurs fair communication overhead, which is much smaller than traditional OTS schemes and even smaller than RSA signature. However, TV-HORS implements a larger key size, which is comparable to BiBa. The requirement of TV-HORS on synchronization is much weaker than other synchronization based authentication schemes. TV-OTS has a much smaller signature size and can be extended to υ-time signatures more efficiently. Based on the TV-OTS model, 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. It provides fast signing and verification and buffering free data processing. The only drawback is its relatively large key size. This problem can be mitigated by fairly trading off some tolerance to packet loss, which is interesting aspect to explore for future work.

A trapdoor hash-based signature amortization technique meets the various challenges to provide a very good authentication of delay sensitive and real-time streams in content distribution, multicast, and peer-to-peer networks. It has resilience against transmission losses. Also it has small and constant memory and compute requirements at the sender and receiver and minimal constant communication overhead. Along with, it has least per block communication and signature generation overheads. The disadvantage of this scheme is a small sender side delay.

Designing extremely high speed signature schemes come at the cost of unreasonably high storage and communication overheads which tend to increase linearly with the size of the message which is signed. Furthermore, BiBa and HORS require frequent redistribution of new public keys to maintain the security of the scheme. This adds significant overheads to the communication and storage costs. The TV-OTS scheme avoids some problems of the BiBa and HORS schemes. But it still requires large public keys on order of 10KiB, also requires time synchronization and does not provide long-term nonrepudiation as the signatures can be forged by the receiver after a reasonable effort.

IV. REFERENCES

[1] Santosh Chandrasekhar, Saikat Chakrabarti, and Mukesh Singhal, "A Trapdoor Hash-Based Mechanism for Stream Authentication," IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 9, NO. 5, SEPTEMBER/OCTOBER 2012.

[2] Q. Wang, H. Khurana, Y. Huang, and K. Nahrstedt, "Time Valid One-Time Signature for Time-Critical Multicast Data Authentication," Proc. IEEE INFOCOM, pp. 1233-1241, 2009.

[3] L. Reyzin and N. Reyzin, "Better Than Biba: Short One-Time Signatures with Fast Signing and Verifying," Proc. Seventh Australian Conf. Information Security and Privacy (ACISP), L.M. Batten and J. Seberry, eds., pp. 144-153, 2002.

[4] Perrig, "The BiBa One-Time Signature and Broadcast Authentication Protocol," Proc. ACM Conf. Computer and Comm. Security (CCS), pp. 28-37, 2001.

[5] Perrig, R. Canetti, J.D. Tygar, and D.X. Song, "Efficient Authentication and Signing of Multicast Streams over Lossy Channels," Proc. IEEE Symp. Security and Privacy, pp. 56-73, 2000.

[6] P. Golle and N. Modadugu, "Authenticating Streamed Data in the Presence of Random Packet Loss," Proc. Network and Distributed System Security Symp. (NDSS), 2001.

[7] J.M. Park, E.K.P. Chong, and H.J. Siegel, "Efficient Multicast Stream Authentication Using Erasure Codes," ACM Trans. Information and System Security, vol. 6, no. 2, pp. 258-285, 2003.

[8] C.K. Wong and S.S. Lam, "Digital Signatures for Flows and Multicasts," IEEE/ACM Trans. Networking, vol. 7, no. 4, pp. 502- 513, Aug. 1999.

[9] 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.

[10] R. Merkle., "Protocols for public key cryptosystems," in Proceedings of the IEEE Symposiumon Research in Security and Privacy, Oakland, CA, Apr.1980.

[11] L. Lamport, "Constructing digital signatures from one-way function," Technical Report SRI-CSL-98, SRI International Computer Lab, 1979.