# MABS: Multicast Authentication Based on Batch Signature

Yun Zhou, Xiaoyan Zhu, and Yuguang Fang, *Fellow*, *IEEE*

**Abstract**—Conventional block-based multicast authentication schemes overlook the heterogeneity of receivers by letting the sender choose the block size, divide a multicast stream into blocks, associate each block with a signature, and spread the effect of the signature across all the packets in the block through hash graphs or coding algorithms. The correlation among packets makes them vulnerable to packet loss, which is inherent in the Internet and wireless networks. Moreover, the lack of Denial of Service (DoS) resilience renders most of them vulnerable to packet injection in hostile environments. In this paper, we propose a novel multicast authentication protocol, namely MABS, including two schemes. The basic scheme (MABS-B) eliminates the correlation among packets and thus provides the perfect resilience to packet loss, and it is also efficient in terms of latency, computation, and communication overhead due to an efficient cryptographic primitive called batch signature, which supports the authentication of any number of packets simultaneously. We also present an enhanced scheme MABS-E, which combines the basic scheme with a packet filtering mechanism to alleviate the DoS impact while preserving the perfect resilience to packet loss.

**Index Terms**—Multimedia, multicast, authentication, signature.

## 1 Introduction

Multicast [1] is an efficient method to deliver multimedia content from a sender to a group of receivers and is gaining popular applications such as realtime stock quotes, interactive games, video conference, live video broadcast, or video on demand. Authentication is one of the critical topics in securing multicast [2], [3], [4], [5], [6], [7] in an environment attractive to malicious attacks. Basically, multicast authentication may provide the following security services:

1. **Data integrity**: Each receiver should be able to assure that received packets have not been modified during transmissions.
2. **Data origin authentication**: Each receiver should be able to assure that each received packet comes from the real sender as it claims.
3. **Nonrepudiation**: The sender of a packet should not be able to deny sending the packet to receivers in case there is a dispute between the sender and receivers.

All the three services can be supported by an asymmetric key technique called *signature*. In an ideal case, the sender generates a signature for each packet with its private key, which is called *signing*, and each receiver checks the validity of the signature with the sender's public key, which is called *verifying*. If the verification succeeds, the receiver knows the packet is *authentic*.

Designing a multicast authentication protocol is not an easy task. Generally, there are following issues in real world challenging the design. First, efficiency needs to be considered, especially for receivers. Compared with the multicast sender, which could be a powerful server, receivers can have different capabilities and resources. The receiver heterogeneity requires that the multicast authentication protocol be able to execute on not only powerful desktop computers but also resource-constrained mobile handsets. In particular, latency, computation, and communication overhead are major issues to be considered. Second, packet loss is inevitable. In the Internet, congestion at routers is a major reason causing packet loss. An overloaded router drops buffered packets according to its preset control policy. Though TCP provides a certain retransmission capability, multicast content is mainly transmitted over UDP, which does not provide any loss recovery support. In mobile environments, the situation is even worse. The instability of wireless channel can cause packet loss very frequently. Moreover, the smaller data rate of wireless channel increases the congestion possibility. This is not desirable for applications like realtime online streaming or stock quotes delivering. End users of online streaming will start to complain if they experience constant service interruptions due to packet loss, and missing critical stock quotes can cause severe capital loss of service subscribers. Therefore, for applications where the quality of service is critical to end users, a multicast authentication protocol should provide a certain level of resilience to packet loss. Specifically, the impact of packet loss on the authenticity of the already-received packets should be as small as possible.

Efficiency and packet loss resilience can hardly be supported simultaneously by conventional multicast

- *Y. Zhou is with Microsoft Corporation, 1 Microsoft Way, Redmond, WA 98052-8300. E-mail: yun.zhou@microsoft.com.*
- *X. Zhu is with the National Key Laboratory of Integrated Services Networks, Xidian University, PO Box 106, 2 South Taibai Road, Xi'an, Shaanxi 10071, China. E-mail: xyzhu@mail.xidian.edu.cn.*
- *Y. Fang is with the Department of Electrical and Computer Engineering, University of Florida, 435 New Engineering Building, PO Box 116130, Gainesville, FL 32611, and with the National Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China. E-mail: fang@ece.ufl.edu.*

*Manuscript received 28 Jan. 2008; revised 30 Jan. 2009; accepted 17 Oct. 2009; published online 23 Feb. 2010.*
*For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2008-01-0026.*
*Digital Object Identifier no. 10.1109/TMC.2010.37.*

schemes. As is well known that existing digital signature algorithms are computationally expensive, the ideal approach of signing and verifying each packet independently raises a serious challenge to resource-constrained devices. In order to reduce computation overhead, conventional schemes use efficient signature algorithms [8], [9] or amortize one signature over a block of packets [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26] at the expense of increased communication overhead [8], [9], [10], [11] or vulnerability to packet loss [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26].

Another problem with schemes in [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26] is that they are vulnerable to packet injection by malicious attackers. An attacker may compromise a multicast system by intentionally injecting forged packets to consume receivers' resource, leading to *Denial of Service* (DoS). Compared with the efficiency requirement and packet loss problems, the DoS attack is not common, but it is still important in hostile environments. In the literature, some schemes [27], [28], [29], [30], [31], [32] attempt to provide the DoS resilience. However, they still have the packet loss problem because they are based on the same approach as previous schemes [10], [11], [22], [23], [24], [25], [26].

Recently, we demonstrated that batch signature schemes can be used to improve the performance of broadcast authentication [5], [6]. In this paper, we present our comprehensive study on this approach and propose a novel multicast authentication protocol called MABS (in short for *Multicast Authentication based on Batch Signature*). MABS includes two schemes. The basic scheme (called MABS-B hereafter) utilizes an efficient asymmetric cryptographic primitive called *batch signature* [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], which supports the authentication of any number of packets simultaneously with one signature verification, to address the efficiency and packet loss problems in general environments. The enhanced scheme (called MABS-E hereafter) combines MABS-B with packet filtering to alleviate the DoS impact in hostile environments. MABS provides data integrity, origin authentication, and nonrepudiation as previous asymmetric key based protocols. In addition, we make the following contributions:

1. Our MABS can achieve *perfect* resilience to packet loss in lossy channels in the sense that no matter how many packets are lost the already-received packets can still be authenticated by receivers.
2. MABS-B is efficient in terms of less latency, computation, and communication overhead. Though MABS-E is less efficient than MABS-B since it includes the DoS defense, its overhead is still at the same level as previous schemes.
3. We propose two new batch signature schemes based on BLS [36] and DSA [38] and show they are more efficient than the batch RSA [33] signature scheme.

The rest of the paper is organized as follows: We briefly review related work in Section 2. Then, we present a basic scheme for lossy channels in Section 3, which also includes three batch signature schemes based on RSA [33], BLS [36], and DSA, respectively [38]. An enhanced scheme is discussed in Section 4. After performance evaluation in Section 5, the paper is concluded in Section 6.

## 2 Related Work

Schemes in [8], [9] follow the ideal approach of signing and verifying each packet individually, but reduce the computation overhead at the sender by using one-time signatures [8] or $k$-time signatures [9]. They are suitable for RSA [33], which is expensive on signing while cheap on verifying. For each packet, however, each receiver needs to perform one more verification on its one-time or $k$-time signature plus one ordinary signature verification. Moreover, the length of one-time signature is too long (on the order of 1,000 bytes).

Tree chaining was proposed in [10], [11] by constructing a tree for a block of packets. The root of the tree is signed by the sender. Each packet carries the signed root and multiple hashes. When each receiver receives one packet in the block, it uses the authentication information in the packet to authenticate it. The buffered authentication information is further used to authenticate other packets in the same block. Without the buffered authentication information, each packet is independently verifiable at a cost of per-packet signature verification.

Graph chaining was studied in [12], [13], [14], [15], [16], [17], [18], [19], [20], [21]. A multicast stream is divided into blocks and each block is associated with a signature. In each block, the hash of each packet is embedded into several other packets in a deterministic or probabilistic way. The hashes form a graph, in which each path links a packet to the block signature. Each receiver verifies the block signature and authenticates all the packets through the paths in the graph.

Erasure codes were used in [22], [23], [24], [25], [26]. A signature is generated for the concatenation of the hashes of all the packets in one block and then is erasure-coded into many pieces. Erasure codes make each receiver be capable of recovering the block signature when receiving at least a certain number of pieces.

All these schemes [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26] are indeed computationally efficient since each receiver needs to verify only one signature for a block of packets. However, they all increase packet overhead for hashes or erasure codes and the block design introduces latency when buffering many packets. Another major problem is that most schemes [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26] are vulnerable to packet loss even though they are designed to tolerate a certain level of packet loss. If too many packets are lost, other packets may not be authenticated. In particular, if a block signature is lost, the entire block cannot be authenticated.

Moreover, previous schemes [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26] target at lossy channels, which are realistic in our daily life since the Internet and wireless networks suffer from packet loss. In a hostile environment, however, an active attacker can inject forged packets to consume receivers' resource, leading to DoS. In particular, schemes in [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21] are vulnerable to forged signature attacks because they require each receiver to verify each signature whereby to authenticate data packets, and schemes in [22], [23], [24], [25], [26] suffer from packet injection because each receiver has to distinguish a certain number of valid packets from a pool of a large number of packets including injected ones, which is very time-consuming.

In order to deal with DoS, schemes in [27], [28], [29], [30], [31], [32] were proposed. PARM [27] is similar to the tree chaining scheme [10], [11] in the sense that multiple one-way hash chains are used as shared keys between the sender and receivers. Schemes in [28], [29] combine erasure codes with one-way hash chains to detect forged packets. Unfortunately, these schemes [27], [28], [29] are still vulnerable to DoS because they require that one-way hash chains are signed and transmitted to each receiver and therefore an attacker can inject forged signatures for one-way hash chains.

PRABS [30] uses distillation codes to deal with DoS. In particular, valid packets and forged packets are partitioned into disjoint sets and erasure decoding is performed over each set. BAS [31] simply retransmits each signature multiple times to tolerate packet loss and uses selective verification to tolerate injected forged signatures. LTT [32] uses error correction codes to replace erasure codes in schemes [22], [23], [24], [25], [26]. The reason is that error correction codes tolerate error packets. These three schemes [30], [31], [32] are resilient to DoS, but they still have the packet loss problem.

Some other schemes [45], [46], [47], [48], [49], [50], [51], [52], [53] use shared symmetric keys between the sender and receivers to authenticate multicast streams. Though they are more efficient than those using signatures, they cannot provide nonrepudiation as the signature approach, and they require either time synchronization or trustful infrastructures. In this paper, we focus on the signature approach.

Though confidentiality is another important issue for securing multicast, it can be achieved through group key management [54]. In this paper, we focus on multicast authentication.

## 3 Basic Scheme

Our target is to authenticate multicast streams from a sender to multiple receivers. Generally, the sender is a powerful multicast server managed by a central authority and can be trustful. The sender signs each packet with a signature and transmits it to multiple receivers through a multicast routing protocol. Each receiver is a less powerful device with resource constraints and may be managed by a nontrustworthy person. Each receiver needs to assure that the received packets are really from the sender (authenticity) and the sender cannot deny the signing operation (nonrepudiation) by verifying the corresponding signatures.

Ideally, authenticating a multicast stream can be achieved by signing and verifying each packet. However, the per-packet signature design has been criticized for its high computation cost, and therefore, most previous schemes [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32] incorporate a block-based design as shown in Section 2. They do reduce the computation cost, but also introduce new problems. The block design builds up correlation among packets and makes them vulnerable to packet loss, which is inherent in the Internet and wireless networks. Received packets may not be authenticated because some correlated packets are lost.

Also, the heterogeneity of receivers means that the buffer resource at each receiver is different and can vary over the time depending on the overall load at the receiver. In the block design, the required block size, which is chosen by the sender, may not be satisfied by each receiver.

Third, the correlation among packets can incur additional latency. Consider the high layer application needs new data from the low layer authentication module in order to render a smooth video stream to the client user. It is desirable that the lower layer authentication module delivers authenticated packets to the high layer application at the time when the high layer application needs new data. In the per-packet signature design it is not a problem, since each packet can be independently verifiable at any time. In the block design, however, it is possible that the packets buffered at the low layer authentication module are not verifiable because the correlated packets, especially the block signatures, have not been received. Therefore, the high layer application has to either wait, which leads to additional latency, or return with a no-available-packets exception, which could be interpreted as that the buffered packets are "lost." This latency, which is incurred at the high layer when the high layer application waits for the buffered packets to become verifiable, is different from the buffering latency, which is required for the low layer authentication protocol to buffer received packets.

In view of the problems regarding the *sender-favored* block-based approach, we conceive a *receiver-oriented* approach by taking into account the heterogeneity of the receivers. As receiving devices have different computation and communication capabilities, some could be powerful desktop computers, while the others could be cheap handsets with limited buffers and low-end CPUs. Mixed with various channel loss rates, this heterogeneity poses a demand on the capability of adjusting the buffer size and authenticating buffered packets any time when the high layer application requires at each receiver.

In order to fulfill the requirement, the basic scheme MABS-B uses an efficient cryptographic primitive called batch signature [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], which supports simultaneously verifying the signatures of any number of packets. In particular, when a receiver collects $n$ packets:

$$p_i = \{m_i, \sigma_i\}, \quad i = 1, \ldots, n,$$

where $m_i$ is the data payload, $\sigma_i$ is the corresponding signature, and $n$ can be any positive integer, it can input them into an algorithm

$$\mathit{BatchVerify}(p_1, p_2, \ldots, p_n) \in \{True, False\}.$$

If the output is $True$, the receiver knows the $n$ packets are authentic, and otherwise not.

To support authenticity and efficiency, the $\mathit{BatchVerify}()$ algorithm should satisfy the following properties:

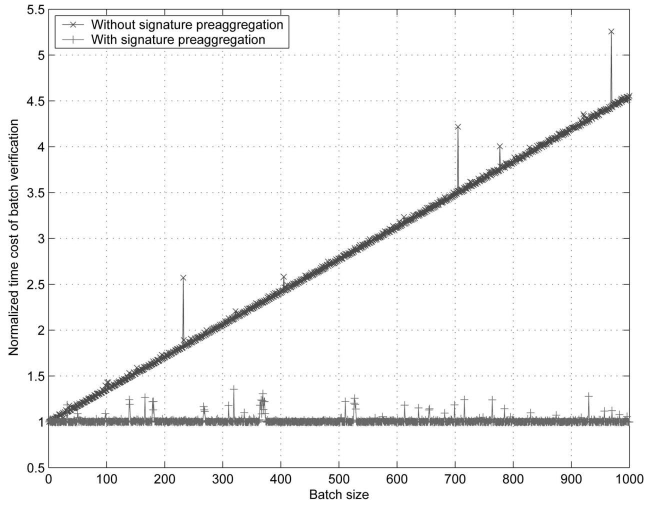


Fig. 1. Normalized time cost of Batch-BLS signature verification.

1. Given a batch of packets that have been signed by the sender, $BatchVerify()$ outputs $True$.
2. Given a batch of packets including some unauthentic packets, the probability that $BatchVerify()$ outputs $True$ is very low.
3. The computation complexity of $BatchVerify()$ is comparable to that of verifying one signature and is increased only gradually when the batch size $n$ is increased.

The computation complexity of $BatchVerify()$ comes with the fact that there are some additional cost on processing multiple packets. As we will show later, those additional computations are mostly modular additions and multiplications, which are much faster than modular exponentiations required in final signature verifications. Theoretically, a concern comes when the cost grows higher than the final signature verification if the batch size is too large. However, it is not the case in reality. The merit of batch signature is that the batch size is chosen by each receiver, which can optimize its own batch size, so that the batch size will not be unmanageably large. Most important, we will show later that in the implementation of batch signature a technique called *signature preaggregation* can be used so that the additional processing of multiple packets is shifted from the time of final batch verification to the time of each packet reception and thus the cost of final batch signature verification is *exactly the same* as that of original signature verification.

In order to show the merit of signature preaggregation, we implemented batch signature by using our Batch-BLS (will be discussed later) as an example. We measured the normalized time cost of batch signature verification with the batch size growing from 1 to 1,000, and recorded the results for two scenarios, with and without signature preaggregation. The result is shown in Fig. 1. We can see that the time cost of general batch verification (without signature aggregation) is still less than that of two single signature verifications when batch size grows up to about 270, which validates the third property of batch signature. The other curve in Fig. 1 shows that if the signature preaggregation is used, then the final batch verification has the same cost of single signature verification, no matter how large the batch size is.

MABS-B uses per-packet signature instead of per-block signature and thus eliminates the correlation among packets. The packet independency makes MABS-B perfect resilient to packet loss. The Internet and wireless channels tend to be lossy due to congestion or channel instability, where packets can be lost according to different loss models, such as random loss or burst loss. In MABS-B, however, no matter how many packets are lost, the already-received packets can still be authenticated by each receiver. This is a significant advantage over previous schemes [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32]. Meanwhile, efficiency can also be achieved because a batch of packets can be authenticated simultaneously through one batch signature verification operation. The packet independency also brings other benefits in terms of smaller latency and communication overhead compared with previous schemes [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32]. In particular, each receiver can verify the authenticity of all the received packets in its buffer whenever the high layer applications require, and there is no additional hash or code overhead in each packet.

Next, we present three implementations. In addition to the one based on RSA [33], we propose two new batch signature schemes based on BLS [36] and DSA [38], which are more efficient than batch RSA. We must point out and will show later that MABS is independent from these signature algorithms. This independency brings the freedom to optimize MABS for a particular physical system or platform as much as possible.

## 3.1 Batch RSA Signature

### 3.1.1 RSA

RSA [33] is a very popular cryptographic algorithm in many security protocols. In order to use RSA, a sender chooses two large random primes $P$ and $Q$ to get $N = PQ$, and then calculates two exponents $e, d \in \mathbb{Z}_N^*$ such that $ed = 1 \ mod \ \phi(N)$, where $\phi(N) = (P-1)(Q-1)$. The sender publishes $(e, N)$ as its public key and keeps $d$ in secret as its private key. A signature of a message $m$ can be generated as $\sigma = (h(m))^d \ mod \ N$, where $h()$ is a collision-resistant hash function. The sender sends $\{m, \sigma\}$ to a receiver that can verify the authenticity of the message $m$ by checking $\sigma^e = h(m) \ mod \ N$.

### 3.1.2 Batch RSA

To accelerate the authentication of multiple signatures, the batch verification of RSA [34], [35] can be used. Given $n$ packets $\{m_i, \sigma_i\}, \quad i = 1, \ldots, n$, where $m_i$ is the data payload, $\sigma_i$ is the corresponding signature and $n$ is any positive integer, the receiver can first calculate $h_i = h(m_i)$ and then perform the following verification:

$$\left(\prod_{i=1}^{n} \sigma_i\right)^e mod \ N = \prod_{i=1}^{n} h_i \ mod \ N. \tag{1}$$

If all $n$ packets are truly from the sender, the equation holds because

$$\left(\prod_{i=1}^{n}\sigma_i\right)^e mod\ N = \prod_{i=1}^{n}\sigma_i^e\ mod\ N = \prod_{i=1}^{n}h_i^{ed}\ mod\ N = \prod_{i=1}^{n}h_i\ mod\ N. \tag{2}$$

Before the batch verification, the receiver must ensure all the messages are distinct. Otherwise batch RSA is vulnerable to the forgery attack [35]. This is easy to implement because sequence numbers are widely used in many network protocols and can ensure all the messages are distinct. It has been proved in [35] that when all the messages are distinct, batch RSA is resistant to signature forgery as long as the underlying RSA algorithm is secure.

In some circumstances, an attacker may not forge signatures but manipulate authentic packets to produce invalid signatures. For example, given two packets $\{m_i, \sigma_i\}$ and $\{m_j, \sigma_j\}$ for $i \neq j$, the attacker can modify them into $\{m_i, \sigma_i\lambda\}$ and $\{m_j, \sigma_j/\lambda\}$. The modified packets can still pass the batch verification, but the signature of each packet is not correct (that is why batch RSA verification is called *screening* in [35]). However, the attacker can do this only when it gets $\{m_i, \sigma_i\}$ and $\{m_j, \sigma_j\}$, which means the message $m_i$ and $m_j$ have been correctly signed by the sender. Therefore, this attack is of no harm to the receiver [35].

### 3.1.3 Requirements to the Sender

In most RSA implementations, the public key $e$ is usually small ($e = 3$ for instance) while the private key $d$ is large. Therefore, the RSA signature verification is efficient while the signature generation is expensive. This poses a challenge to the computation capability of the sender because the sender needs to sign each packet. Choosing a small private key $d$ can improve the computation efficiency but compromise the security. If the sender does not have enough resource, a pair of $\{e, d\}$ with comparable sizes can achieve a certain level of trade-off between computation efficiency and security at the sender part. If the sender is a powerful server, then signing each packet can be affordable in this scenario. Next, we propose two efficient batch signature schemes based on BLS [36] and DSA [38], which can reduce the computation complexity at the sender.

## 3.2 Batch BLS Signature

Here, we propose a batch signature scheme based on the BLS signature in [36].

### 3.2.1 BLS

The BLS signature scheme uses a cryptographic primitive called *pairing*, which can be defined as a map over two cyclic groups $G_1$ and $G_2$, $e : G_1 \times G_1 \rightarrow G_2$, satisfying the following properties:

1. Bilinear: For all $u, v \in G_1$ and $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Nondegenerate: For the generator $g_1$ of $G_1$, i.e., $g_1^p = 1 \in G_1$, where $p$ is the order of $G_1$, we have $e(g_1, g_1) \neq 1 \in G_2$.

The BLS signature scheme consists of three phases:

1. In the *key generation* phase, a sender chooses a random integer $x \in \mathbb{Z}_p$ and computes $y = g_1{}^x \in G_1$. The private key is $x$ and the public key is $y$.
2. Given a message $m \in \{0,1\}^*$ in the *signing* phase, the sender first computes $h = h(m) \in G_1$, where $h()$ is a hash function, then computes $\sigma = h^x \in G_1$. The signature of $m$ is $\sigma$.
3. In the *verification* phase, the receiver first computes $h = h(m) \in G_1$ and then check whether $e(h,y) = e(\sigma, g_1)$.

If the verification succeeds, then the message $m$ is authentic because

$$e(h, y) = e(h, g_1{}^x) = e(h^x, g_1) = e(\sigma, g_1). \tag{3}$$

One merit of the BLS signature is that it can generate a very short signature. It has been shown in [36] that an $n$-bit BLS signature can provide a security level equivalent to solving a *Discrete Log Problem* (DLP) [39] over a finite field of size approximately $2^{6n}$. Therefore, a 171-bit BLS signature provides the same level of security as a 1,024-bit DLP-based signature scheme such as DSA [38]. This is a very nice choice in the scenario where communication overhead is an important issue.

### 3.2.2 Batch BLS

Based on BLS, we propose our batch BLS scheme here. Given $n$ packets $\{m_i, \sigma_i\}, i = 1, \ldots, n$, the receiver can verify the batch of BLS signatures by first computing $h_i = h(m_i)$, $i = 1, \ldots, n$ and then checking whether $e(\prod_{i=1}^{n} h_i, y) = e(\prod_{i=1}^{n} \sigma_i, g_1)$. This is because if all the messages are authentic, then

$$e\left(\prod_{i=1}^{n} h_i, y\right) = \prod_{i=1}^{n} e(h_i, g_1{}^x) = \prod_{i=1}^{n} e\left(h_i{}^x, g_1\right) = e\left(\prod_{i=1}^{n}\sigma_i, g_1\right). \tag{4}$$

We can prove that our batch BLS is secure to signature forgery as long as BLS is secure to signature forgery.

**Theorem 1.** *Suppose an attacker $\mathcal{A}$ can break the batch BLS by forging signatures. Then, another attacker $\mathcal{B}$ can break BLS under the chosen message attack by colluding with $\mathcal{A}$.*

**Proof.** Suppose $\mathcal{B}$ is given $n-1$ messages and their valid signatures $\{m_i, \sigma_i\}, i = 1, \ldots, n-1$, $\mathcal{B}$ can forge a signature $\sigma_n$ for any chosen message $m_n$, such that $\{m_n, \sigma_n\}$ satisfies the BLS signature scheme, by colluding with $\mathcal{A}$ in the following steps:

1. $\mathcal{B}$ sends $n$ messages $m_i, i = 1, \ldots, n$ and $n-1$ signatures $\sigma_i, i = 1, \ldots, n-1$ to $\mathcal{A}$.
2. Because $\mathcal{A}$ can break the batch BLS scheme, $\mathcal{A}$ generates $n$ false signatures $\sigma_i{}', i = 1, \ldots, n$ that pass the batch BLS verification, then returns to $\mathcal{B}$ a value $V = \prod_{i=1}^{n} \sigma_i{}'$.
3. $\mathcal{B}$ computes $\sigma_n = V / \prod_{i=1}^{n-1} \sigma_i$ as the signature for $m_n$, because

$$\begin{aligned} & e\left(\prod_{i=1}^{n} h_i, y\right) = e(V, g_1) \\ & \Rightarrow e\left(\prod_{i=1}^{n} h_i, y\right) = e\left(\prod_{i=1}^{n} \sigma_i, g_1\right) \\ & \Rightarrow e\left(\prod_{i=1}^{n-1} h_i, y\right) e(h_n, y) = e\left(\prod_{i=1}^{n-1} \sigma_i, g_1\right) e(\sigma_n, g_1) \\ & \Rightarrow e(h_n, y) = e(\sigma_n, g_1) \ . \end{aligned} \tag{5}$$

□

Since BLS is forgery-secure under the chosen message attack [36], our batch BLS scheme is also secure to forgery under the chosen message attack.

Also like batch RSA, an attacker may not forge signatures but manipulate authentic packets to produce invalid signatures. For instance, two packets $\{m_i, \sigma_i\}$ and $\{m_j, \sigma_j\}$ for $i \neq j$ can be replaced with $\{m_i, \sigma_i \lambda\}$ and $\{m_j, \sigma_j / \lambda\}$ and still pass the batch verification. However, it does not affect the correctness and the authenticity of $m_i$ and $m_j$ because they have been correctly signed by the sender.

#### 3.2.3 Requirements to the Sender

In our batch BLS, the sender needs to sign each packet. Because BLS can provide a security level equivalent to conventional RSA and DSA with much shorter signature [36], the signing operation is more efficient than the RSA signature generation. Moreover, BLS can be implemented over elliptic curves [55], [56], which have been shown in the literature to be more efficient than finite integer fields on which RSA is implemented. Therefore, we can expect that our batch BLS is more affordable by the sender than batch RSA and also achieve computation efficiency at the receiver.

### 3.3 Batch DSA Signature

DSA [38] is another popular digital signature algorithm. Unlike RSA, which is based on the hardness of factoring two large primes, DSA is deemed secure based on the difficulty of solving DLP [39]. A batch DSA signature scheme was proposed in [40], but later was found insecure [41]. Harn improved the security of [40] in [42], [43]. Unfortunately, Boyd and Pavlovski pointed out in [44] that Harn's work is still vulnerable to malicious attacks. Here, we propose a batch DSA scheme based on Harn's work and counteract the attack described in [44].

#### 3.3.1 Harn DSA

In Harn DSA [43], some system parameters are defined as:

1. $p$, a prime longer than 512 bits.
2. $q$, a 160-bit prime divisor of $p-1$.
3. $g$, a generator of $\mathbb{Z}_p^*$ with order $q$, i.e., $g^q = 1 \ mod \ p$.
4. $x$, the private key of the signer, $0 < x < q$.
5. $y$, the public key of the signer, $y = g^x \ mod \ p$.
6. $h()$, a hash function generating an output in $\mathbb{Z}_q^*$.

Given a message $m$, the signer generates a signature by:

1. randomly selecting an integer $k$ with $0 < k < q$,
2. computing $h = h(m)$,
3. computing $r = (g^k \ mod \ p) \ mod \ q$, and
4. computing $s = rk - hx \ mod \ q$.

The signature for $m$ is $(r, s)$.

The receiver can verify the signature by first computing $h = h(m)$ and then checking whether

$$((g^{sr^{-1}} y^{hr^{-1}}) \ mod \ p) \ mod \ q = r.$$

This is because if the packet is authentic, then

$$\begin{aligned} & ((g^{sr^{-1}} y^{hr^{-1}}) \ mod \ p) \ mod \ q \\ & = ((g^{(s+hx)r^{-1}}) \ mod \ p) \ mod \ q \\ & = (g^k \ mod \ p) \ mod \ q \\ & = r. \end{aligned} \tag{6}$$

#### 3.3.2 Harn Batch DSA

Given $n$ packets $\{m_i, (r_i, s_i)\}, i = 1, \ldots, n$, the receiver can verify the batch of signatures by first computing $h_i = h(m_i)$ and then checking whether

$$\left(\left(g^{\sum_{i=1}^{n} s_i r_i^{-1}} y^{\sum_{i=1}^{n} h_i r_i^{-1}}\right) \ mod \ p\right) \ mod \ q = \prod_{i=1}^{n} r_i \ mod \ q \ . \tag{7}$$

This is because, if the batch of packets is authentic, then

$$\begin{aligned} & \left(\left(g^{\sum_{i=1}^{n} s_i r_i^{-1}} y^{\sum_{i=1}^{n} h_i r_i^{-1}}\right) \ mod \ p\right) \ mod \ q \\ & = \left(\left(g^{\sum_{i=1}^{n} (s_i + h_i x) r_i^{-1}}\right) \ mod \ p\right) \ mod \ q \\ & = \left(g^{\sum_{i=1}^{n} k_i} \ mod \ p\right) \ mod \ q \\ & = \prod_{i=1}^{n} r_i \ mod \ q. \end{aligned} \tag{8}$$

#### 3.3.3 The Boyd-Pavlovski Attack

Boyd and Pavlovski [44] pointed out an attack against the Harn batch DSA scheme [43], where an attacker can forge signatures for any chosen message set that has not been signed by the sender. The process is:

1. Choose $B$ and $C$, calculate $A = (g^B y^C \ mod \ p) \ mod \ q$.
2. For any message set $m_i$, $i = 1, \ldots, n$, randomly choose $r_i$, $i = 1, \ldots, n-2$.
3. Compute $r_{n-1}$ and $r_n$ to ensure that

$$\prod_{i=1}^{n} r_i \ mod \ q = A \ mod \ q \tag{9}$$

$$\sum_{i=1}^{n} h_i r_i^{-1} \ mod \ q = C \ mod \ q. \tag{10}$$

4. Randomly choose $s_i$, $i = 1, \ldots, n-1$ and compute $s_n$ to ensure that

$$\sum_{i=1}^{n} s_i r_i^{-1} \ mod \ q = B \ mod \ q. \tag{11}$$

The probability that $\{m_i, r_i, s_i\}$, $i = 1, \ldots, n$ are forged messages satisfying the batch verification is $\frac{1}{2}$ [44].

#### 3.3.4 Our Batch DSA

In order to counteract the Boyd-Pavlovski attack, our batch DSA makes an improvement to the Harn DSA algorithm.

We replace the hash operation $h(m)$ in the signature generation and verification process with $h(r, m)$. All the other steps are the same as those in Harn's scheme.

Though it is simple, our method can significantly increase the security of batch DSA. In the Boyd-Pavlovski attack, the attacker can compute $r_i$ values according to (9) and (10) because parameters $A$, $C$, $h_i$ values are known. By introducing $r_i$ into the hash operation, the hash values $h_i$ in (10) are unknown to the attacker. Therefore, the attacker cannot compute $r_i$ values and the forgery attack discussed in [44] is defeated.

Like the cases in batch RSA and our batch BLS, the attacker may manipulate authentic packets $\{m_i, (r_i, s_i)\}$ to produce invalid signatures $\{m_i, (r_i', s_i')\}$, which can still pass the batch verification. The attacker can keep $r_i$ unchanged, randomly choose $s_i'$, $i = 1, \ldots, n-1$ and solve $s_n'$ satisfying

$$\sum_{i=1}^{n} s_i' r_i^{-1} \ mod\ q = \sum_{i=1}^{n} s_i r_i^{-1} \ mod\ q. \tag{12}$$

However, this attack does not affect the correctness and authenticity of messages because they have been really signed by the sender [44]. Therefore, the receiver can still accept them because the batch verification succeeds.

### 3.3.5 Requirements to the Sender

In batch RSA and our batch BLS, the sender needs to compute one modular exponentiation to sign each packet. In our batch DSA, the sender needs to compute one modular exponentiation to get $r$ and two modular multiplications to get $s$. However, $r$ is independent on the message $m$. Therefore, the sender can generate many $r$ values offline. When the sender starts a multicast session, it can use reserved $r$ values to compute $s$ values. In this way, only two modular multiplications are necessary to sign a packet. Therefore, our batch DSA is much more efficient than batch RSA and our batch BLS at the sender, while also achieving computation efficiency at the receiver.

## 3.4 Preaggregation of Message Hashes and Signatures

If we take a closer look at batch-RSA and batch-BLS, we can notice that they use exactly the orignal RSA and BLS algorithms, respectively. The only difference is that the batch algorithms take the aggregations of message hashes and signatures $\{\prod_{i=1}^{n} h_i, \prod_{i=1}^{n} \sigma_i\}$ as the parameters to the original signature algorithms. This aggregation is independent of the final signature verification. Therefore, each receiver can compute and update $\{\prod_{i=1}^{n} h_i, \prod_{i=1}^{n} \sigma_i\}$ after receiving every packet. When the time of batch verification comes, each receiver needs *just one* signature verification.

For batch-DSA, the aggregations of message hashes and signatures $\{\sum_{i=1}^{n} s_i r_i^{-1}, \sum_{i=1}^{n} h_i r_i^{-1}, \prod_{i=1}^{n} r_i\}$ are inside the signature verification. Therefore, the cost of the aggregations is incurred with the final signature verification. However, if we modify the original DSA so that it takes $\{sr^{-1}, hr^{-1}, r\}$ instead of $\{h, s, r\}$ as parameters, then batch-DSA can take the advantage of preaggregating message hashes and signatures, just like batch-RSA and batch-BLS.

Obviously, the computation cost at each receiver can be more manageable no matter how large each batch is if signature preaggregation is enforced, as shown in Fig. 1.

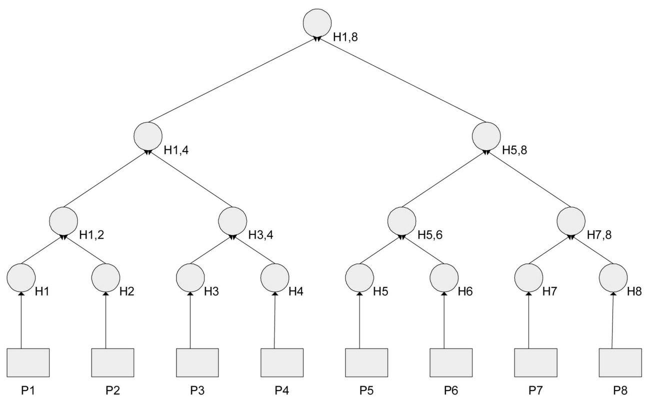


Fig. 2. An example of Merkle tree. Each leaf is a hash of one packet. Each internal node is the hash value on both its left and right children.

## 4 Enhanced Scheme

The basic scheme MABS-B targets at the packet loss problem, which is inherent in the Internet and wireless networks. It has perfect resilience to packet loss, no matter whether it is random loss or burst loss. In some circumstances, however, an attacker can inject forged packets into a batch of packets to disrupt the batch signature verification, leading to DoS. A naive approach to defeat the DoS attack is to divide the batch into multiple smaller batches and perform a batch verification over each smaller batch, and this *divide-and-conquer* approach can be recursively carried out for each smaller batch, which means more signature verifications at each receiver. In the worst case, the attacker can inject forged packets at very high frequency and expect that each receiver stops the batch operation and recovers the basic per-packet signature verification, which may not be viable at resource-constrained receiver devices.

In this section, we present an enhanced scheme called MABS-E, which combines the basic scheme MABS-B and a packet filtering mechanism to tolerate packet injection. In particular, the sender attaches each packet with a *mark*, which is unique to the packet and cannot be spoofed. At each receiver, the multicast stream is classified into disjoint sets based on marks. Each set of packets comes from either the real sender or the attacker. The mark design ensures that a packet from the real sender never falls into any set of packets from the attacker, and vice versa. Next, each receiver only needs to perform $BatchVerify()$ over each set. If the result is $True$, the set of packets is authentic. If not, the set of packets is from the attacker, and the receiver simply drops them and does not need to divide the set into smaller subsets for further batch verification. Therefore, a strong resilience to DoS due to injected packets can be provided.

In MABS-E, Merkle tree [57] is used to generate marks. An example is illustrated in Fig. 2. The sender constructs a binary tree for eight packets. Each leaf is a hash of one packet. Each internal node is the hash value on the concatenation of its left and right children. For each packet, a mark is constructed as the set of the siblings of the nodes along the path from the packet to the root. For example, the mark of the packet $P_3$ is $\{H_4, H_{1,2}, H_{5,8}\}$ and the root can be recovered as $H_{1,8} = H((H_{1,2}, (H(P_3), H_4)), H_{5,8})$.

Constructing a Merkle tree is very efficient because only hash operations are performed. Meanwhile, the one-way property of hash operation ensures that given the root of a Merkle tree it is infeasible to find a packet, which is not in

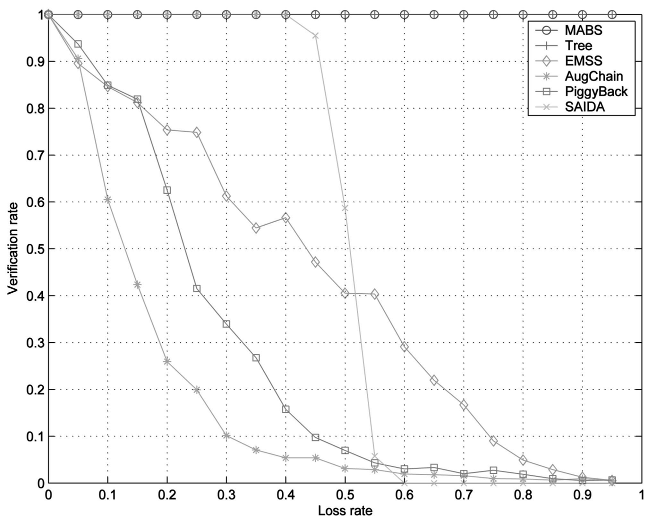


Fig. 3. Verification rate under the random loss model.

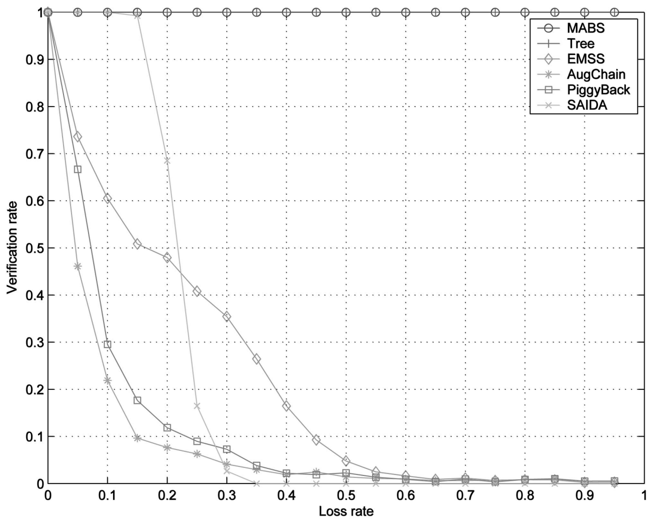


Fig. 4. Verification rate under the burst loss model with the maximum burst length 10.

the set associated with the Merkle tree and from which there is a path to the root. This guarantees that forged packets cannot fall into the set of authentic packets.

When the sender has a set of packets for multicast, it generates a Merkle tree for the set and attaches a mark to each packet. The root can be recovered based on each packet and its mark. Each receiver can find whether two packets belong to the same set by checking whether they lead to the same root value. Therefore, the recovered roots help classify received packets into disjoint sets. Each receiver does not need to wait for a set to include all the packets under the Merkle tree, and it can batch-verify the set anytime. Once the set is authentic, the corresponding root can be used to authenticate the rest of packets under the same Merkle tree without batch-verifying them, which saves computation overhead at each receiver.

An attacker may inject small sets of forged packets or even inject single packet that does not belong to any set. In this case, each receiver has many small sets and each of them has only a few packets. Doing the $BatchVerify$ algorithm on each small set compromises efficiency. Since the sets from the sender can have a large number of packets, each receiver can choose a threshold (say $t$) and start batch verification over one set only when the set has no less than $t$ packets. If a set has less than $t$ packets and the root value recovered from the set has not been authenticated, the receiver simply drop the set of packets without processing them and thus save computation resource. For each receiver, the threshold $t$ can vary according to the receiver's computation resource. In this way, the impact of DoS due to packet injection can be reduced by a factor of $t$.

## 5 Performance Evaluation

In this section, we evaluate MABS performance in terms of resilience to packet loss, efficiency, and DoS resilience. As we discussed before, MABS does not assume any particular underlying signature algorithm. This is also true for all the literature multicast authentication schemes referenced in this paper. Therefore, all the discussions and evaluations of MABS and the literature works in Section 5.1 and Section 5.2 are under the assumption that they are using the same underlying signature algorithm. The discussion of signature algorithms is in Section 5.3.

### 5.1 Resilience to Packet Loss

We use simulations to evaluate the resilience to packet loss. The metric here is the *verification rate*, i.e., the ratio of the number of authenticated packets to the number of received packets.

We compare MABS with some well-known loss tolerant schemes EMSS [14], augmented chain (AugChain) [18], PiggyBack [16], tree chain (Tree) [11], and SAIDA [23]. These schemes are representatives of graph chaining, tree chaining, and erasure coding schemes and are widely used in performance evaluation in the literature.

For EMSS [14], we choose the chain configuration of $5-11-17-24-36-39$, which has the best performance among all the configurations of length 6 as is shown in [14]. For AugChain [18], we choose $C_{3,7}$ chain configuration. For PiggyBack [16], we choose two class priorities. For Tree chain [11], we choose binary tree. For SAIDA [23], we choose the erasure code $(256, 128)$. For all these schemes, we choose the block size of 256 packets and simulate over 100 blocks. We consider the random loss and the burst loss with a maximum loss length of 10 packets. The verification rates under different loss rates are given in Fig. 3 and Fig. 4.

We can see that the verification rates of EMSS [14], augmented chain (AugChain) [18] and PiggyBack [16] are decreased quickly when the loss rate is increased. The reason is that graph chaining results in the correlation among packets and this correlation is vulnerable to packet loss. SAIDA [23] illustrates a resilience to packet loss up to a certain threshold, because of the threshold performance of erasure codes. Our MABS and Tree schemes [11] have perfect resilience to packet loss in the sense that all the received packets can be authenticated. This is because all the packets in MABS and Tree schemes are independent from each other. As we will show later, however, Tree [11] achieves this independency by incurring large overhead and latency at the sender and each receiver and is vulnerable to DoS, while our MABS-B has less overhead and latency and MABS-E is resilient to DoS at the same level of overhead as Tree [11].

TABLE 1
Notations

| Symbols | Definitions |
|---|---|
| $n$ | The block or batch size, i.e., there are $n$ packets per block or per batch. |
| $S$ | Signing operation or one signature. |
| $V$ | Signature verification operation. |
| $H$ | Hash operation or one hash value. |
| $d$ | The number of hashes in each packet in EMSS [14] ($d \geq 6$). |
| $r$ | The number of classes in PiggyBack [16]. |
| $k_i$ | Loss resilience factor for class $i$ in PiggyBack [16] ($0 < k_i < n/r$). |
| $p$ | The number of buffered packets at the sender in AugChain [18] ($0 < p < n$). |
| $m$ | The least number of packets for decoding in SAIDA [23], PRABS [30] and LTT [32] ($0 < m < n$). |
| $E_{EC}$, $D_{EC}$ | Erasure encoding and decoding in SAIDA [23], PRABS [30] and BAS [31]. |
| $E_{ECC}$, $D_{ECC}$ | Encoding and decoding of error-correction codes in LTT [32]. |
| $n_d$, $n_h$, $n_s$ | The numbers of data packets, hash packets and signature packets in BAS [31] ($n_d + n_h + n_s = n$). |
| $\pi$ | The signature verification probability in BAS [31] ($0 < \pi < 1$). |
| $k$ | The number of signature verifications in LTT [32] ($k \geq 1$). |
| $\beta$ | The attack factor. for $n$ valid packets, there are $\beta n$ injected packets ($\beta > 1$). |
| $t$ | The authentication threshold in MABS-E. Each receiver starts batch verification when the batch size is no less than $t$ ($0 < t < n$). |
| $C_m^{m+\beta n}$ | Choosing $m$ packets out of $m + \beta n$ packets. |

One thing needs to be pointed out is that we do not differentiate between MABS-B and MABS-E in Fig. 3 and Fig. 4. MABS-B is perfect resilient to packet loss because of its inherent design. While it is not designed for lossy channels, MABS-E can also achieve the perfect resilience to packet loss in lossy channels. In the lossy channel model, where no DoS attack is assumed to present, we can set the threshold $t = 1$ (refer to Section 4) for MABS-E, and thus each receiver can start batch-verification as long as there is at least one packet received for each set of packets constructed under the same Merkle tree. Once the received packets are authenticated, the extracted root of the Merkle tree can be used to authenticate the rest of packets in the same set within several hash operations. If we set $t > 1$, which is not the best choice for the lossy channel, MABS-E can tolerate up to $n - t$ lost packets for each set of $n$ packets, which shows a threshold-based loss resilience, similar to SAIDA [23].

## 5.2 Efficiency

We consider latency, computation, and communication overhead for efficiency evaluation under lossy channels and DoS channels. The notations used here are defined in Table 1. All the evaluations are carried out over $n$ packets. The results are depicted in Table 2 and Table 3.

### *5.2.1 Comparisons over Lossy Channels*

Table 2 shows the comparisons between MABS-B and well-known loss-tolerant schemes tree chain (Tree) [11], EMSS [14], PiggyBack [16], augmented chain (AugChain) [18], and SAIDA [23]. We also include MABS-E and three DoS resilient schemes PRABS [30], BAS [31], and LTT [32] in the table just for comparisons even though they are not designed for lossy channels.

Previous block-based schemes introduce latency either at the sender [11], [16] or at each receiver [14], [31] or both [18], [23], [30], [32]. The latency is inherent in the block design due to chaining or coding. At the sender side, the correlation among a block of packets has to be established before the sender starts sending the packets. At each receiver, the latency is incurred when the high layer application waits for the buffered packets to be authenticated after the correlation is recovered. This receiver side latency is variable depending on whether the correlation among the underline buffered packets has been recovered or not when the high layer application needs new data, and its maximum value is the block size. MABS-B eliminates the correlation among packets. Each packet is independently sent out at the sender. At each receiver, the high layer application does not need to wait because the low layer authentication module can deliver authenticated packets at any time when the high layer application needs new data. This makes MABS-B pretty suitable for realtime multimedia applications. MABS-E introduces latency at the sender because it includes a tree construction to counteract DoS, but it still preserves the merit of instantaneous authentication at each receiver as MABS-B.

Both the block-based schemes and MABS require one signature verification operation on a block or a batch of $n$ packets at each receiver. In addition, the schemes using

TABLE 2
Comparisons over Lossy Channels

| Schemes | Latency | | Computation Overhead | | Communication Overhead |
|---|---|---|---|---|---|
| | Sender | Receiver | Sender | Receiver | |
| Tree [11] | $n$ | 1 | $1S + (2n-1)H$ | $1V + (n\log_2 n + n)H$ | $nS + n\log_2 nH$ |
| EMSS [14] | 1 | $n$ | $1S + nH$ | $1V + nH$ | $1S + dnH$ |
| PiggyBack [16] | $n$ | 1 | $1S + nH$ | $1V + nH$ | $1S + (2n - \sum_{i=1}^{r-1} k_i)H$ |
| AugChain [18] | $p$ | $n$ | $1S + nH$ | $1V + nH$ | $1S + 2nH$ |
| SAIDA [23] | $n$ | $m$ | $1S + (n+1)H + 2E_{EC}$ | $1V + (n+1)H + 2D_{EC}$ | $\frac{n}{m}S + \frac{n^2}{m}H$ |
| PRABS [30] | $n$ | $m$ | $1S + 3nH + 2E_{EC}$ | $1V + (n\log_2 n + 2n + 1)H + 2D_{EC}$ | $\frac{n}{m}S + (\frac{n^2}{m} + n\log_2 n)H$ |
| BAS [31] | 1 | $2n$ | $1S + (n_d + n_h)H + 1E_{EC}$ | $1V + (n_d + n_h)H + 1D_{EC}$ | $n_sS + (n_d + n_h)H$ |
| LTT [32] | $n$ | $m$ | $1S + nH + 1E_{ECC}$ | $1V + nH + 1D_{ECC}$ | $\frac{n}{m}S + \frac{n^2}{m}H$ |
| MABS-B | 1 | 1 | $nS$ | $1V$ | $nS$ |
| MABS-E | $n$ | 1 | $nS + (2n-1)H$ | $1V + (n\log_2 n + n)H$ | $nS + n\log_2 nH$ |

TABLE 3
Comparisons over DoS Channels

| Schemes | Receiver Computation Overhead |
|---|---|
| Tree [11] | $(1+\beta n)V + (n\log_2 n + n)H$ |
| EMSS [14] | $(1+\beta n)V + nH$ |
| PiggyBack [16] | $(1+\beta n)V + nH$ |
| AugChain [18] | $(1+\beta n)V + nH$ |
| SAIDA [23] | $C_m^{m+\beta n}V + (C_m^{m+\beta n} + n)H + 2C_m^{m+\beta n}D_{EC}$ |
| PRABS [30] | $(1+\frac{\beta n}{m})V + ((1+\beta)n\log_2 n + (2+\frac{m+1}{m}\beta)n + 1)H + 2(1+\frac{\beta n}{m})D_{EC}$ |
| BAS [31] | $\pi(n_s+\beta n)V + (n_d+n_h)H + 1D_{EC}$ |
| LTT [32] | $kV + (n+\beta n)H + 1D_{ECC}$ |
| MABS-B | $(1+\beta)nV$ |
| MABS-E | $(1+\frac{\beta n}{t})V + (1+\beta)(n\log_2 n + n)H$ |

chaining also require many hashes, and the ones using coding require multiple hashes and one or two decoding operations. MABS-B is more efficient since there is no more hashing or decoding operations. MABS-E requires $n\log_2 n$ hashes, which is comparable to previous schemes using hashing and coding.

In MABS, a trade-off for perfect resilience to packet loss is that the sender needs to sign each packet, which incurs more computation overhead than conventional block-based schemes. Therefore, efficient signature generation is desirable at the sender. Compared with RSA [33], which is efficient in verifying but is expensive in signing, BLS [36] and DSA [38] are pretty good candidates as we will show later. Moreover, in multimedia multicast, the sender is usually a powerful server and thus the per-packet signature generation can be affordable, and the advance of computing technology makes it easier in the long run.

For $n$ packets, Tree [11] require an overhead of $n$ signature and $\mathcal{O}(n\log_2 n)$ hashes, schemes in [14], [16], [18], [23], [30], [31], [32] require one or more signatures and up to $\mathcal{O}(n^2)$ hashes. MABS-B and MABS-E require $n$ signatures and MABS-E requires additional $\mathcal{O}(n\log_2 n)$ hashes. If long signatures are used (like 1,024-bit RSA), MABS-B and MABS-E have more communication overhead than those in [14], [16], [18], [23], [30], [31], [32], which is the same case as Tree [11]. However, BLS [36] generates short signatures of 171 bits, which is comparable to most well-known hash algorithms MD5 [58] (128 bits) and SHA-1 [59] (160 bits). Therefore, MABS can have the same level of communication efficiency as conventional schemes when BLS is used.

### *5.2.2 Comparisons over DoS Channels*

DoS is a method for an attacker to deplete the resource of a receiver. Processing injected packets from the attacker always consumes a certain amount of resource. Here, we assume an attacker factor $\beta$, which means that for $n$ valid packets $\beta n$ invalid packets are injected. The computation overheads at each receiver for different schemes are depicted in Table 3.

For schemes in [11], [14], [16], [18], which authenticate signatures first and then authenticate packet through hash chains, the attacker can inject $\beta n$ forged signature packets because signature verification is an expensive operation. For SAIDA [23], which requires erasure decoding, the attacker simply injects $\beta n$ forged packets because each receiver has to choose a certain number of valid packets from all the $(1+\beta)n$ packet to do decoding, which can have a significant number of tries.

Schemes in [30], [31], [32] are designed for DoS defense. They can alleviate the impact of packet injection by a constant factor to reduce the computation cost at each receiver.

MABS-B is vulnerable to packet injection as schemes in [11], [14], [16], [18], [23] since they are designed only for lossy channels. MABS-E has the same level of DoS resilience as those in [30], [31], [32].

## 5.3 Comparisons of Signature Schemes

We compare the computation overhead of three batch signature schemes in Table 4. RSA [33] and BLS [36] require one modular exponentiation at the sender and DSA [38] requires two modular multiplications when $r$ value is computed offline. Usually one $c$-bit modular exponentiation is equivalent to $1.5c$ modular multiplications over the same field [35], [44]. Moreover, a $c$-bit modular exponentiation in DLP is equivalent to a $\frac{c}{6}$-bit modular exponentiation in BLS for the same security level. Therefore, we can estimate that the computation overhead of one 1,024-bit RSA signing operation is roughly equivalent to that of 768 DSA signing operations (1,536 modular multiplications) and that of 6 BLS signing operations (each one is corresponding to 255 modular multiplications).

According to the report [60] on the computational overhead of signature schemes on PIII 1 GHz CPU, the signing and verification time for 1,024-bit RSA with a 1,007-bit private key are 7.9 ms and 0.4 ms, for 157-bit BLS are 2.75 ms and 81 ms, and for 1,024-bit DSA with a 160-bit private key (without precomputing $r$ value) are 4.09 ms and 4.87 ms. We can observe that for BLS and DSA the signing is efficient but the verification is expensive, and vice versa for RSA. Therefore, we can save more computation resource at the sender by using our batch BLS and batch DSA than batch RSA. It is also meaningful to use our batch BLS and batch DSA at the receiver to save computation resources.

We also compare the length of two popular hash algorithm MD5 [58] and SHA-1 [59] and the signature length of three

TABLE 4
Computational Overhead of Different Batch Schemes

| Schemes | Sender(per packet) | Receiver (per $n$ packets) |
|---|---|---|
| Batch RSA | $1\ E$ | $1\ E + (2n-2)\ M$ |
| Batch BLS | $1\ E$ | $2\ P + (2n-2)\ M$ |
| Batch DSA | $2\ M$ | $2\ E + 3n\ M$ |

*$E$-modular exponentiation. $M$-modular multiplication. $P$-pairing.*

TABLE 5
Communication Overhead of Signature Schemes

| Schemes | Length (bits) |
|---|---|
| MD5 | 128 |
| SHA-1 | 160 |
| RSA | 1024 |
| BLS | 171 |
| DSA | 320 |

signature algorithms in Table 5. Given the same security level as 1,024-bit RSA, BLS generates a 171-bit signature and $DSA$ generates a 320-bit signature. It is clear that by using BLS or DSA, MABS can achieve more bandwidth efficiency than using RSA, and could be even more efficient than conventional schemes using a large number of hashes.

## 6 Conclusions

To reduce the signature verification overheads in the secure multimedia multicasting, block-based authentication schemes have been proposed. Unfortunately, most previous schemes have many problems such as vulnerability to packet loss and lack of resilience to denial of service (DoS) attack. To overcome these problems, we develop a novel authentication scheme MABS. We have demonstrated that MABS is perfectly resilient to packet loss due to the elimination of the correlation among packets and can effectively deal with DoS attack. Moreover, we also show that the use of batch signature can achieve the efficiency less than or comparable with the conventional schemes. Finally, we further develop two new batch signature schemes based on BLS and DSA, which are more efficient than the batch RSA signature scheme.

## Acknowledgments

This work was partially supported by the US National Science Foundation under grants CNS-0916391, CNS-0716450, and CNS-0626881. The works of Fang and Zhu were also partially supported by the 111 Project under Grant B08038. The work of Zhu was also partially supported by the National Natural Science Foundation of China under grant 60772136 and supported by the Fundamental Research Funds for the Central Universities.

## References

[1] S.E. Deering, "Multicast Routing in Internetworks and Extended LANs," *Proc. ACM SIGCOMM Symp. Comm. Architectures and Protocols,* pp. 55-64, Aug. 1988.

[2] T. Ballardie and J. Crowcroft, "Multicast-Specific Security Threats and Counter-Measures," *Proc. Second Ann. Network and Distributed System Security Symp. (NDSS '95),* pp. 2-16, Feb. 1995.

[3] P. Judge and M. Ammar, "Security Issues and Solutions in Mulicast Content Distribution: A Survey," *IEEE Network Magazine,* vol. 17, no. 1, pp. 30-36, Jan./Feb. 2003.

[4] Y. Challal, H. Bettahar, and A. Bouabdallah, "A Taxonomy of Multicast Data Origin Authentication: Issues and Solutions," *IEEE Comm. Surveys & Tutorials,* vol. 6, no. 3, pp. 34-57, Oct. 2004.

[5] Y. Zhou and Y. Fang, "BABRA: Batch-Based Broadcast Authentication in Wireless Sensor Networks," *Proc. IEEE GLOBECOM,* Nov. 2006.

[6] Y. Zhou and Y. Fang, "Multimedia Broadcast Authentication Based on Batch Signature," *IEEE Comm. Magazine,* vol. 45, no. 8, pp. 72-77, Aug. 2007.

[7] K. Ren, K. Zeng, W. Lou, and P.J. Moran, "On Broadcast Authentication in Wireless Sensor Networks," *Proc. First Ann. Int'l Conf. Wireless Algorithms, Systems, and Applications (WASA '06),* Aug. 2006.

[8] S. Even, O. Goldreich, and S. Micali, "On-Line/Offline Digital Signatures," *J. Cryptology,* vol. 9, pp. 35-67, 1996.

[9] P. Rohatgi, "A Compact and Fast Hybrid Signature Scheme for Multicast Packet," *Proc. Sixth ACM Conf. Computer and Comm. Security (CCS '99),* Nov. 1999.

[10] C.K. Wong and S.S. Lam, "Digital Signatures for Flows and Multicasts," *Proc. Sixth Int'l Conf. Network Protocols (ICNP '98),* pp. 198-209, Oct. 1998.

[11] 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.

[12] R. Gennaro and P. Rohatgi, "How to Sign Digital Streams," *Information and Computation,* vol. 165, no. 1, pp. 100-116, Feb. 2001.

[13] R. Gennaro and P. Rohatgi, "How to Sign Digital Streams," *Proc. 17th Ann. Cryptology Conf. Advances in Cryptology (CRYPTO '97),* Aug. 1997.

[14] A. Perrig, R. Canetti, J.D. Tygar, and D. Song, "Efficient Authentication and Signing of Multicast Streams over Lossy Channels," *Proc. IEEE Symp. Security and Privacy (SP '00),* pp. 56-75, May 2000.

[15] Y. Challal, H. Bettahar, and A. Bouabdallah, "A$^2$Cast: An Adaptive Source Authentication Protocol for Multicast Streams," *Proc. Ninth Int'l Symp. Computers and Comm. (ISCC '04),* vol. 1, pp. 363-368, June 2004.

[16] S. Miner and J. Staddon, "Graph-Based Authentication of Digital Streams," *Proc. IEEE Symp. Security and Privacy (SP '01),* pp. 232-246, May 2001.

[17] Z. Zhang, Q. Sun, W-C Wong, J. Apostolopoulos, and S. Wee, "A Content-Aware Stream Authentication Scheme Optimized for Distortion and Overhead," *Proc. IEEE Int'l Conf. Multimedia and Expo (ICME '06),* pp. 541-544, July 2006.

[18] P. Golle and N. Modadugu, "Authenticating Streamed Data in the Presence of Random Packet Loss," *Proc. Eighth Ann. Network and Distributed System Security Symp. (NDSS '01),* Feb. 2001.

[19] Z. Zhang, Q. Sun, and W-C Wong, "A Proposal of Butterfly-Graphy Based Stream Authentication over Lossy Networks," *Proc. IEEE Int'l Conf. Multimedia and Expo (ICME '05),* July 2005.

[20] S. Ueda, N. Kawaguchi, H. Shigeno, and K. Okada, "Stream Authentication Scheme for the Use over the IP Telephony," *Proc. 18th Int'l Conf. Advanced Information Networking and Application (AINA '04),* vol. 2, pp. 164-169, Mar. 2004.

[21] D. Song, D. Zuckerman, and J.D. Tygar, "Expander Graphs for Digital Stream Authentication and Robust Overlay Networks," *Proc. 2002 IEEE Symp. Security and Privacy (S&P '02),* May 2002.

[22] J.M. Park, E.K.P. Chong, and H.J. Siegel, "Efficient Multicast Packet Authentication Using Signature Amortization," *Proc. IEEE Symp. Security and Privacy (SP '02),* pp. 227-240, May 2002.

[23] 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, May 2003.

[24] A. Pannetrat and R. Molva, "Authenticating Real Time Packet Streams and Multicasts," *Proc. Seventh IEEE Int'l Symp. Computers and Comm. (ISCC '02),* pp. 490-495, July 2002.

[25] A. Pannetrat and R. Molva, "Efficient Multicast Packet Authentication," *Proc. 10th Ann. Network and Distributed System Security Symp. (NDSS '03),* Feb. 2003.

[26] Y. Wu and T. Li, "Video Stream Authentication in Lossy Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '06),* vol. 4, pp. 2150-2155, Apr. 2006.

[27] Y. Lin, S. Shieh, and W. Lin, "Lightweight, Pollution-Attack Resistant Multicast Authentication Scheme," *Proc. ACM Symp. Information, Computer, and Comm. Security (ASIACCS '06),* Mar. 2006.

[28] J. Jeong, Y. Park, and Y. Cho, "Efficient DoS Resistant Multicast Authentication Schemes," *Proc. Int'l Conf. Computational Science and Its Applications (ICCSA '05),* May 2005.

[29] S. Choi, "Denial-of-Service Resistant Multicast Authentication Protocol with Prediction Hashing and One-Way Key Chain," *Proc. Seventh IEEE Int'l Symp. Multimedia (ISM '05),* Dec. 2005.

[30] C. Karlof, N. Sastry, Y. Li, A. Perrig, and J.D. Tygar, "Distillation Codes and Applications to DoS Resistant Multicast Authentication," *Proc. 11th Ann. Network and Distributed System Security Symp. (NDSS '04),* Feb. 2004.

[31] C.A. Gunter, S. Khanna, K. Tan, and S. Venkatesh, "DoS Protection for Reliably Authenticated Broadcast," *Proc. 11th Ann. Network and Distributed System Security Symp. (NDSS '04)*, Feb. 2004.

[32] A. Lysyanskaya, R. Tamassia, and N. Triandopoulos, "Multicast Authentication in Fully Adversarial Networks," *Proc. IEEE Symp. Security and Privacy (SP '04)*, May 2004.

[33] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM*, vol. 21, no. 2, pp. 120-126, Feb. 1978.

[34] L. Harn, "Batch Verifying Multiple RSA Digital Signatures," *IEE Electronic Letters*, vol. 34, no. 12, pp. 1219-1220, June 1998.

[35] M. Bellare, J.A. Garay, and T. Rabin, "Fast Batch Verification for Modular Exponentiation and Digital Signatures," *Proc. Advances in Cryptology (EUROCRYPT '98)*, pp. 236-250, May 1998.

[36] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," *Proc. Seventh Int'l Conf. Theory and Application of Cryptology and Information Security Advances in Cryptology (ASIACRYPT '01)*, pp. 514-532, Dec. 2001.

[37] S. Cui, P. Duan, and C.W. Chan, "An Efficient Identity-Based Signature Scheme with Batch Verifications," *Proc. First Int'l Conf. Scalable Information Systems*, 2006.

[38] *FIPS PUB 186, Digital Signature Standard (DSS)*, May 1994.

[39] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Trans. Information Theory*, vol. IT-31, no. 4, pp. 469-472, July 1985.

[40] D. Naccache, D. M'Raihi, S. Vaudenay, and D. Raphaeli, "Can D.S.A. be Improved? Complexity Trade Offs with the Digital Signature Standard," *Proc. Workshop Theory and Application of Cryptographic Techniques Advances in Cryptology (EUROCRYPT '94)*, pp. 77-85, May 1995.

[41] C.H. Lim and P.J. Lee, "Security of Interactive DSA Batch Verification," *IEE Electronic Letters*, vol. 30, no. 19, pp. 1592-1593, Sept. 1994.

[42] L. Harn, "DSA-Type Secure Interactive Batch Verification Protocols," *IEE Electronic Letters*, vol. 31, no. 4, pp. 257-258, Feb. 1995.

[43] L. Harn, "Batch Verifying Multiple DSA-Type Digital Signatures," *IEE Electronic Letters*, vol. 34, no. 9, pp. 870-871, Apr. 1998.

[44] C. Boyd and C. Pavlovski, "Attacking and Repairing Batch Verification Schemes," *Proc. Sixth Int'l Conf. Theory and Application of Cryptology and Information Security Advances in Cryptology (ASIANCRYPT '00)*, pp. 58-71, Dec. 2000.

[45] Y. Desmedt, Y. Frankel, and M. Yung, "Multi-Receiver/Multi-Sender Network Security: Efficient Authenticated Multicast/Feedback," *Proc. IEEE INFOCOM*, vol. 3, pp. 2045-2054, May 1992.

[46] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast Security: A Taxonomy and Some Efficient Constructions," *Proc. IEEE INFOCOM*, vol. 2, pp. 708-716, Mar. 1999.

[47] L. Lamport, "Password Authentication with Insecure Communication," *Comm. ACM*, vol. 24, no. 11, pp. 770-772, Nov. 1981.

[48] N.M. Haller, "The S/Key One-Time Password System," *Proc. ISOC Symp. Network and Distributed Security*, Feb. 1994.

[49] F. Bergadano, D. Cavagnino, and B. Crispo, "Individual Single-Source Authentication on The Mbone," *Proc. IEEE Int'l Conf. Multimedia and Expo (ICME '00)*, vol. 1, pp. 541-544, July 2000.

[50] A. Perrig, R. Canetti, D. Song, and J. Tygar, "Efficient and Secure Source Authentication for Multicast," *Proc. Network and Distributed System Security Symp. (NDSS '01)*, 2001.

[51] A. Perrig, "The BiBa One-Time Signature and Broadcast Authentication Protocol," *Proc. Eighth ACM Conf. Computer and Comm. Security (CCS '01)*, pp. 28-37, Nov. 2001.

[52] Q. Li and W. Trappe, "Reducing Delay and Enhancing DoS Resistance in Multicast Authentication through Multigrade Security," *IEEE Trans. Info. Forensics and Security*, vol. 1, no. 2, pp. 190-204, June 2006.

[53] S. Xu and R. Sandhu, "Authenticated Multicast Immune to Denial-of-Service Attack," *Proc. ACM Symp. Applied Computing (SAC '02)*, 2002.

[54] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication," *ACM Computing Surveys*, vol. 35, no. 3, pp. 309-329, Sept. 2003.

[55] N. Koblitz, "Elliptic Curve Cryptosystems," *Math. Computation*, vol. 48, pp. 203-209, 1987.

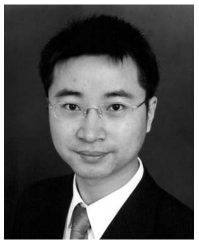
[56] V. Miller, "Uses of Elliptic Curves in Cryptography," *Proc. Int'l Cryptology Conf. (CRYPTO '85)*, pp. 417-426, 1986.

[57] R. Merkle, "Protocols for Public Key Cryptosystems," *Proc. IEEE Symp. Security and Privacy*, Apr. 1980.

[58] R. Rivest, "The MD5 Message-Digest Algorithm," RFC 1319, Apr. 1992.

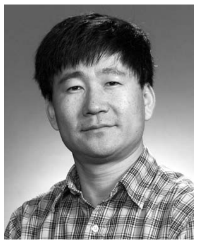
[59] D. Eastlake and P. Jones, "US Secure Hash Algorithm 1 (SHA1)," RFC 3174, Sept. 2001.

[60] P. Barreto, H. Kim, B. Lynn, and M. Scott, "Efficient Algorithms for Pairing-Based Cryptosystems," *Proc. Int'l Cryptology Conf. (CRYPTO '02)*, 2002.



**Yun Zhou** received the BE degree in electronic information engineering and the ME degree in communication and information system from the Department of Electronic Engineering and Information Science, University of Science and Technology, Hefei, China, in 2000 and 2003, respectively, and the PhD degree from the Department of Electrical and Computer Engineering, University of Florida, Gainesville, in 2007. He is currently with Microsoft Corporation. His research interests include security, cryptography, wireless communications and networking, signal processing, and operating systems.

**Xiaoyan Zhu** received the BE, ME, and PhD degrees from the School of Telecommunications Engineering at Xidian University, Xi'an, China, in 2000, 2004, and 2009, respectively. She is now a visiting research scholar in the Department of Electrical and Computer Engineering at the University Florida, Gainesville, on leave from the School of Telecommunications Engineering, Xidian University, Xi'an, China, where she is a lecturer. Her research interests include wireless networks, network security, and network coding.



**Yuguang Fang** received the PhD degree in systems engineering from Case Western Reserve University in 1994 and the PhD degree in electrical engineering from Boston University in 1997. He was an assistant professor in the Department of Electrical and Computer Engineering at the New Jersey Institute of Technology from 1998 to 2000. He then joined the Department of Electrical and Computer Engineering at University of Florida in 2000 as an assistant professor, got an early promotion to an associate professor with tenure in 2003, and to a full professor in 2005. He holds a University of Florida Research Foundation (UFRF) Professorship from 2006 to 2009, a Changjiang scholar chair professorship with Xidian University, Xi'an, China, from 2008 to 2011, and a guest chair professorship with Tsinghua University, China, from 2009 to 2012. He has published more than 250 papers in refereed professional journals and conferences. He received the US National Science Foundation Faculty Early Career Award in 2001 and the US Office of Naval Research Young Investigator Award in 2002, and is the recipient of the Best Paper Award from the IEEE International Conference on Network Protocols (ICNP) in 2006 and the IEEE TCGN Best Paper Award from the IEEE High-Speed Networks Symposium, IEEE GLOBECOM, in 2002. He is also active in professional activities. He is currently serving as the editor-in-chief for *IEEE Wireless Communications* and serves/served on several editorial boards of technical journals including the *IEEE Transactions on Communications*, the *IEEE Transactions on Wireless Communications*, *IEEE Wireless Communications Magazine*, and *ACM Wireless Networks*. He was an editor for the *IEEE Transactions on Mobile Computing* and currently serves on its steering committee. He has been actively participating in professional conference organizations such as serving as the steering committee cochair for QShine (2004-2008), the technical program vice-chair for IEEE INFOCOM 2005, a technical program symposium cochair for IEEE GLOBECOM 2004, an area technical program committee chair for IEEE INFOCOM (2009-2010), and a member of the technical program committee for IEEE INFOCOM (1998, 2000, 2003-2008) and ACM MobiHoc (2008-2009). He is a fellow of the IEEE and the IEEE Computer Society and a member of the ACM.