Efficient Multicast Authentication

1B.Sangeetha, 2S.Puvaneswari, 3S.Hemalatha
1,2,3 AP/CSE
Kings College of Engineering, Punalkulam
Bsangeetha85@gmail.com

Abstract: **Conventional block-based multicast authentication protocols overlook the heterogeneity of receivers in mobile computing 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 or coding algorithms. They suffer from some drawbacks. First, they require that the entire block with its signature be collected before authenticating every packet in the block. This authentication latency can lead to the jitter effect on real-time applications at receivers. Second, the block-based approach is vulnerable to packet loss in mobile computing in the sense that the loss of some packets makes the other packets unable to be authenticated, especially when the block signature is lost. Third, they are also vulnerable to DoS attacks caused by the injection of forged packets. In this article we propose a novel multicast authentication protocol based on an efficient cryptographic primitive called a batch signature. Our protocol supports the verification of the authenticity of any number of packets simultaneously and avoids the shortcomings of the block-based approach.**

INTRODUCTION

Multicast is an efficient method to deliver multimedia content from a sender to a group of receivers, and is gaining popular applications such as real-time stock quotes or video on demand. Authentication is one of the critical topics in securing multicast in mobile computing, an open environment attractive to malicious attacks. Basically, multicast authentication may provide the following services:

Data integrity: Each receiver should be able to ensure that received packets have not been modified during transmissions.

Data origin authentication: Each receiver should be able to ensure that each received packet comes from the real sender as it claims.

Non-repudiation: 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. Designing a multicast authentication protocol needs to consider the following requirements:

Efficiency: While the sender of multimedia content is usually a powerful server, receivers can have different capabilities and resources. Receiver heterogeneity requires that the multicast authentication protocol be implementable on not only powerful computers but also resource-constrained mobile handsets.

Resilience to packet loss: Packets may be lost during wireless transmission. The impact of packet loss on the authenticity of the received packets should be as small as possible.

Resilience to denial of service (DoS) attacks: Forged packets injected into a multicast stream increase the workload of receivers and cause the drop of authentic packets, leading to DoS. A certain level of resilience to DoS attacks should be provided.

In order to meet the aforementioned three requirements, we can use asymmetric key techniques. In an ideal case, each packet includes a signature generated with the sender's private key, and each receiver verifies the signature with the sender's public key. As it is well known that existing digital signature algorithms are computationally expensive, the ideal approach raises a serious challenge to receivers' computational capabilities.

Conventional multicast authentication protocols use a block-based approach to reduce the number of signature verification operations at each receiver. In particular, the sender divides a multicast stream into blocks, associates each block with a signature, and spreads the effect of the signature across all the packets in the block through some data structures. One data structure is a hash chain. In each block, the hash of each packet is embedded into several other packets in a deterministic or probabilistic way. The hashes form chains linking each packet to the block signature. Each receiver verifies the block signature and authenticates all the packets through hash chains. A simple example is illustrated ,given a block of n packets Pi, $i = 1, 2, \ldots, n$. The hash of Pi is included in $Pi+1$ for $i = 1, 2, \ldots, n - 1$. For the hash of the last packet, $H(Pn)$, a signature S is computed. After receiving the signature S, each receiver can authenticate all the packets one by one in this block by tracing back through the hash chain. In order to tolerate a certain level of packet loss, each packet can be linked to multiple other packets and further to the block signature. Coding is another method to disperse the effect of a signature. In Fig. 2, for example, a block consists of n packets Pi, $i = 1, 2, \ldots, n$. For each payload Mi, a hash $H(Mi)$ is computed. A signature S is generated for the concatenation of all the hashes. Then the signature and all the hashes are input into an (m, n)-coding algorithm, which outputs n pieces Si, $i = 1, 2, \ldots, n$. Each packet in the block is attached to one piece before being sent out. Each receiver can recover the n hashes and the signature S as long as it receives at least m pieces. The block-based approach can achieve computational efficiency at receivers because the computation requirement is reduced to one signature verification plus some hash or decoding operations for a block of packets. However, they suffer from some drawbacks.

First, each receiver has to collect an entire block with a block signature before authenticating every packet in the block. A larger block size achieves higher computational efficiency, but incurs longer latency for authentication. This *authentication latency* can lead to the jitter effect on real-time applications at the receiver when some time-critical packets have to be delayed for block signature verification. In addition, the block size is chosen by the sender, which overlooks the heterogeneity of the receivers in mobile communications, where mobile devices with constrained memory resources may not be able to allocate enough memory space to buffer a block of packets.

Second, the relationship between the packets of each block due to the hashes or coding makes the multimedia stream vulnerable to packet loss in the sense that the loss of some packets makes the other packets unable to be authenticated. In an extreme case, the loss of the block signature makes the whole block of packets unable to be authenticated.

Third, previous protocols are vulnerable to DoS attacks. An attacker can inject a large number of forged packets to disrupt the authentication process and cost extra computation overhead at receivers.

In this paper, we propose a novel multicast authentication protocol based on an efficient asymmetric cryptographic primitive called a *batch signature*, which supports the authentication of any number of packets simultaneously. Our design can provide data integrity, origin authentication, and non-repudiation like previous asymmetric key-based protocols. In addition, we make the following contributions:

By using batch signatures, the authentication latency is eliminated in the sense that each receiver can verify the authenticity of any number of packets in its buffer simultaneously whenever high-layer applications require. This is a significant improvement in the quality of real-time applications compared to conventional block-based protocols.

It is perfectly resilient to packet loss in the sense that no matter how many packets are lost, the rest can also be verified by receivers. Most conventional protocols cannot totally solve the packet loss problem.

It provides strong resilience to DoS attacks by using packet filtering, while most conventional protocols are vulnerable to DoS.

Next we introduce the architecture of the proposal, including the idea of batch signature based on RSA and a packet filtering technique based on a Merkle tree. After performance evaluation, we conclude this article.

SYSTEM ARCHITECTURE

Our target is to authenticate multicast streams from a sender to multiple receivers. The sender is a powerful multicast server, while each receiver can be a less powerful device with resource constraints. We consider a multicast authentication protocol providing data integrity, origin authentication, and non-repudiation. Therefore, we use asymmetric key techniques. 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 in mobile communications. As mobile devices have different computation and communication capabilities, some could be powerful mobile vehicles or portable laptops, while others could be PDAs or handsets with constrained memory resource and low-end CPUs. This poses a demand on the capability to authenticate any number of packets simultaneously on request by the high-layer applications at each receiver. Our design can achieve this goal by using batch signatures, while the block-based approach in previous protocols cannot fulfil this requirement. In a batch signature scheme, each packet is signed by the sender with a signature. When a receiver collects *n* packets *pi* = {*mi*, s*i*}, *i* = 1,…, *n*, where *mi* is the data payload, s*i* is the corresponding signature, and *n* can be any positive integer, it inputs them into a verification algorithm *BatchVerify*(*p*1, *p*2, …, *pn*) Î {*True*, *False*}. If the output is *True*, the *n* packets are authentic; otherwise, they are not. To support authenticity and efficiency, *BatchVerify*() should have the following properties:

• Given a batch of packets that have been signed by the server, *BatchVerify*() outputs *True*.
• Given a batch of packets including some packets that have never been signed by the sender and are potentially forged by an attacker, the probability that *BatchVerify*() outputs *True* is very low.
• The computation complexity of *BatchVerify*() is much lower than that of verifying all the packets one by one and increases gradually when batch size *n* is increased. By *BatchVerify*(), our protocol can achieve computational efficiency comparable to conventional block-based protocols in the sense that a batch of packets can be authenticated simultaneously. In addition, the authentication latency in conventional blocked-based protocols is eliminated. Each receiver can verify the Authenticity of all the received packets in its buffer whenever high-layer applications require. This is a significant improvement in the quality of real-time applications. Multicast channels can be lossy, where packets are lost according to different loss models, such as random or burst loss. The traditional block-based approach is vulnerable to packet loss due to the correlation between packets caused by hash chains and coding algorithms. The batch-based approach discussed here removes the correlation between packets, which makes our design perfectly resilient to packet loss. No matter how many packets are lost, the rest can still be verified. A potential threat to batch signature is DoS. An attacker may inject forged packets into a batch of packets to disrupt the batch signature verification. 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; 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 each receiver to stop the batch operation and recover the basic per-packet signature verification. In order to deal with DoS attacks, we need a method to filter out forged packets. In particular, the sender attaches a *mark* to each packet, 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 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 verifications. Therefore, strong resilience to DoS due to forged packets can be provided. The architecture of our protocol is depicted in Fig. 3. For each message *mi*, the sender computes a signature *si*, then computes a mark *wi* for

{*mi*, s*i*}. Therefore, each packet is *pi* = {*mi*, s*i*, *wi*}. These packets are sent over a lossy and hostile channel to multiple receivers through multicast routing. Each receiver gets a stream of packets including both authentic and potentially forged ones. The receiver classifies received packets into disjoint sets based on marks. Each set consists of packets *pi* = {*mi*, s*i*} where *wi* is no longer needed. Then the receiver performs *BatchVerify*() over each set. If the verification over one set succeeds, the set of packets is authentic. Otherwise, the set of packets is forged and can be dropped without further verification on each packet.

BATCH SIGNATURE

We can construct batch signature based on RSA, BLS, and DSA. In addition, we developed two novel batch signature schemes based on BLS and DSA. For the sake of simplicity, we use the well-known RSA as an example to illustrate the idea of batch signature. In RSA, a sender chooses two large random primes *P* and *Q* to get *N* = *PQ*, and calculates two exponents *e*, *d* Î Z*N**, where Z*N** = {1, 2, …,*N* – 1} such that *ed* = 1 *mode* f(*N*), where Euler's totient function f(*N*) = (*P* – 1)(*Q* – 1).The sender publishes (*e*, *N*) as its public key and keeps *d* secret as its private key. A signature of a message *m* is generated as s= *hd mod N*, where *h* = *h*(*m*) and *h*() is a collision resistant hash function. The sender sends {*m*, s} to each receiver that verifies the authenticity of *m* by checking s*e* = *h*(*m*) *mod N*, since s*e mod N* = h ed *mod N* = *h*1 *mod* f(*N*) *mod N* =*h mod N* if the packet is authentic.

To accelerate the authentication of multiple signatures, the batch verification of RSA can be used. Given *n* packets {*mi*, s*i*}, *i* = 1, …, *n*, where *mi* is the data payload and s *i* is the corresponding signature, the receiver can first calculate *hi* = *h*(*mi*) and then perform the verification of (P*ni* =1 s*i*)*e* = P*ni* =1 *hi mod N*. If all *n* packets are truly from the sender, the equation holds because Before the batch verification, each receiver must ensure that all the messages are distinct. Otherwise, the batch RSA is vulnerable to packet forgery. This is easy to implement because sequence numbers are widely used in many network protocols and can ensure all the messages are distinct. An attacker may attempt to inject forged packets and expect them to pass the batch verification. However, the attacker can hardly succeed because he does not know the sender's private key. It has been shown that when all the messages are distinct, the batch RSA is resistant to packet forgery as long as the underlying RSA is secure. The attacker may not inject forged packets but manipulate authentic packets to produce invalid signatures. For example, given two packets {*mi*, s*i*} and {*mj*, s*j*} for *i* ¹ *j*, the attacker can modify them into {*mi*, s*i*l} and {*mj*, s*j*/l}. The modified packets can still pass the batch verification, but the signature of each packet is not correct. However, the attacker can do this only when he gets {*mi*, s*i*}and {*mj*,s*j*}, which means the message *mi* and *mj* have been correctly signed by the sender. Therefore, this attack is of no harm to the receiver.

PACKET FILTERING

In our design, a Merkle tree is used to generate marks. The sender constructs a binary tree for 8 packets. Each leaf is a hash of one packet. Each internal node is the hash value for both its left and right children. For each packet, a mark is constructed as the set of siblings of the nodes along the path from the packet to the root.

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 operation is 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 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. Once a set is authentic, the corresponding root can be used to authenticate the rest of the packets under the same Merkle tree without batch-verifying them, which saves computation overhead at each receiver.

PERFORMANCE EVALUATION

In this section we compare our protocol with the block-based approach.

AUTHENTICATION LATENCY

The block-based approach requires each receiver to collect an entire block before authenticating every packet in the block. A larger block size achieves higher computation efficiency, but also incurs longer authentication latency. Given a block size is *n*, before starting to authenticate each packet in one block, each receiver has to buffer at least *n* packets when hash chains are used and *m* packets when (*m*, *n*)-coding is used. Our design does not have authentication latency. Because there is no relationship among packets and no limit on the number of packets in batch verification, each receiver can perform the batch verification over its buffered packets whenever higher-layer applications require. This can greatly increase the quality of service (QoS) performance of multicast streams.

RESILIENCE TO PACKET LOSS

Block-based protocols introduce correlation between packets due to the use of hash chains or coding. This correlation makes the multicast stream vulnerable to packet loss. The protocols using hash chains have poor resilience to packet loss. If some packets are lost, other packets may not be authenticated, and if the signature of a block is lost, the entire block of packets cannot be authenticated. Protocols using (*m*, *n*)-coding provide a threshold-based resilience to packet loss. On the contrary, our design is perfectly resilient to packet loss in the sense that each packet is independently verifiable. No matter how many packets are lost, the remaining packets can still be authenticated. This is a significant improvement over previous work.

DOS RESILIENCE

DoS is a method for an attacker to deplete the resources of a receiver. Processing forged packets from the attacker always consumes a certain amount of resources. The block-based approach has poor resilience to DoS. Because there is no filtering, each receiver has to recover the relationship among authentic packets mixed with forged packets, which is very time- and computation- intensive. In addition, a deadlock can form at the receiver in the hash-chain approach when the receiving buffer is exhausted by mixing forged packets and authentic packets without block signatures. Those authentic packets are waiting for signatures, but signatures cannot be received because the receiving buffer is exhausted by forged packets. By using a Merkle tree in our design, authentic packets and forged packets are separated into disjoint sets. Batch verification is carried out over each set. Therefore, each batch verification can authenticate a set of packets, and no more is needed. The deadlock experienced by the hash chain approach can also be eliminated. If an attacker wants to inject some forged packets into a batch consisting of authentic packets, he must break the one-way property of a Merkle tree. However, this attempt fails because given the root of a Merkle tree; it is infeasible to find out a packet from which there is a path to the root due to the one-way property of hash functions. Therefore, by using a Merkle tree, our design has strong resilience to DoS attacks.

COMPUTATIONAL OVERHEAD

Both the block-based protocols and our design require one signature verification operation on a block or a batch of *n* packets. In addition, the protocols using hash chains also require *n* hashes, and those using coding require *n* hashes and one decoding operation. Our design requires *n*log*n* hashes, which is more expensive than using hash chains and less expensive than using coding. However, the overall computation overhead of all these protocols at each receiver is at the same level since hash operation is much more efficient (on the order of microseconds) than signature operation (on the order of milliseconds).There are other protocols that use symmetric key techniques. They are more computationally efficient because they do not use signatures. However, they cannot provide non-repudiation, and they also suffer from security threats such as DoS or time synchronization attacks. We need to point out that in our design the sender needs to sign each packet, which costs more computation overhead than conventional block-based protocols. In multimedia multicast, however, the sender is usually a powerful server; thus, per-packet signature generation can be affordable.

COMMUNICATION OVERHEAD

For *n* packets, conventional protocols require an overhead of one signature and O(*n*) hashes, while our design requires an overhead of *n* signatures and O(*n*log*n*) hashes. The increased overhead is a trade-off for increased security. However, we have developed more bandwidth efficient batch signature schemes. When BLS is used the signature length is 171 bits. A most well-known hash algorithm, SHA-1, generates a hash value of 160 bits. Therefore, our protocol can also achieve the same level of communication efficiency as conventional protocols.

CONCLUSIONS

In this article we propose a novel multicast authentication protocol based on batch signature by leveraging the heterogeneity of receivers in mobile computing. Compared to traditional solutions, our protocol supports the verification of the authenticity of any number of packets simultaneously whenever high-layer applications require and thus eliminates the authentication latency. In addition, our design is perfectly resilient to packet loss in the sense that no matter how many packets are lost, the rest can also be verified by the receiver. Moreover, our design can efficiently defeat DoS attacks by using packet filtering.

REFERENCES

[1] Y. Zhou and Y. Fang, “BABRA: Batch-Based Multicast Authentication in Wireless Sensor Networks,” *Proc.49th IEEE GLOBECOM*

[2] K. Ren *et al.*, “On Multicast Authentication in Wireless Sensor Networks,” *Proc. WASA ’06*

[3] http://wiki.answers.com

[4] http://www.techgig.com

[5] P. Judge and M. Ammar, “Security Issues and Solutions inMulicast Content Distribution: A Survey,” IEEE Network Magazine,vol. 17, no. 1, pp. 30-36, Jan./Feb. 2003.

[6]Y. Zhou and Y. Fang, “BABRA: Batch-Based Broadcast Authentication in Wireless SensorNetworks,” Proc. IEEE GLOBECOM, Nov. 2006.