International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

COPY RIGHT

IJIEMR Transactions, online available on 17 April 2019.

Link : http://www.ijiemr.org

Title:- Validating The Traditional Trapdoor Hash Function For Stream Authentication.

Volume 08, Issue 04, Pages: 218 - 225.

Paper Authors

GANGABATTULA V N D KISHORE , N.SRINIVASA RAO.

Department of MCA, SKBR PG College.

USE THIS BARCODE TO ACCESS YOUR ONLINE PAPER

To Secure Your Paper As Per UGC Approvals We Are Providing A Electronic Bar Code

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

VALIDATING THE TRADITIONAL TRAPDOOR HASH FUNCTION FOR STREAM AUTHENTICATION

[1]GANGABATTULA V N D KISHORE , [2]N.SRINIVASA RAO
[1]PG Scholar, Department of MCA, SKBR PG College, Amalapuram
[2]Assistant Professor, Department of MCA, SKBR PG College, Amalapuram

ABSTRACT: Web based services involve distribution of content like digital audio, video, software, games, stock quotes, streaming presentations, and live news feeds through distributed networking technologies, like Content Distribution Networks (CDN's), multicast networks, and peer-to-peer networks. We protect delay sensitive streams against malicious attacks, security mechanisms and auditing mechanisms need to be designed to efficiently process long sequence of bits. We propose a novel signature amortization technique based on trapdoor hash functions for authenticating each and every individual data blocks in the stream. Our technique provides for each and every intermediate blocks in the stream we want to avoid the transmission loss and we will provide constant memory requirements for sender as well as receiver and we want to authenticate and verify the stream to avoid unauthenticated user and to avoid malicious content.

KEY WORDS: Stream authentication, cryptography, content distribution network, trap door functions.

I.INTRODUCTION

ANY web-based services involve distribution of content like digital audio, video, software, games, stock quotes, streaming presentations, and live news feeds through distributed networking technologies, like content distribution networks (CDN's), multicast networks, and peer-to-peer networks. Unfortunately, these modern distributed systems, designed to distribute content to a large group of users, also provide a platform for adversarial users to launch a myriad attacks with widespread consequences. Adversaries can masquerade as legitimate content providers to distribute malicious content possibly infected with worms, viruses, etc. Adversaries can also place themselves in the content distribution path, for example, by compromising web caches [1], and modify the content in ways that can potentially harm client devices. Authenticating the content plays a crucial role in preventing these attacks.

Although CDNs like Akamai employ mechanisms for providing physical security, host system security, access control, software reliability and integrity, and 24 7 monitoring and response, these mechanisms are primarily designed for providing security of the CDN's service network infrastructure and ensuring proper functioning of its distributed network of servers, rather than protecting content distributed through the CDN [2]. The task of protecting content is the responsibility of content providers. Popular content providers often provide highly personalized user experience by inserting targeted advertisements and dynamically generated content. Today majority of mass-viewed content is

dynamically generated and rich in multimedia that include combination of text, audio, still images, animation, video, and interactive content forms that are gathered from a myriad of sources, assembled and presented to the user. In such scenarios, malicious modification of content by malicious sources becomes a legitimate threat. To highlight this point, recently, Google and MSN (Microsoft(R)) were observed to be distributing malware after attackers were able to trick the networks by masquerading as a legitimate advertising provider and inserting malicious advertisements (by exploiting two Internet Explorer, one Java, and four Adobe Reader flaws) that installed the HDD Plus malware [3]. YouTube was also a victim of an attack where malicious code was inserted into pages (by exploiting a cross-site scripting vulnerability) displaying the targeted videos that would launch when users opened the video clip redirecting users to pornographic sites and display falsified news alerts [4]. While flaws in software that were exploited were eventually patched, these attacks could be prevented by using authentication mechanisms to protect the content.

Using conventional techniques for message authentication require the sender and the receiver to have the ability to store the entire message before processing the message. However, in most instances of distributing content like digitized multimedia, the content provider transmits the content in the form of digital streams that receivers consume at more or less the stream arrival rate without excessive delay. To protect such delay-sensitive digital streamsagainst malicious attacks, security mechanisms must efficiently process long sequence of bits in a manner that allows receivers to verify the authenticity of the stream in portions (to avoid possessing the entire stream before verification) without excessive processing delays associated with each portion of the stream. This is typically done by dividing the stream into blocks (or chunks) and using an efficient security mechanism to secure each block of data.

Efficient processing of streams, both at servers and clients, is critical as the Internet continues to grow, server loads continue to increase, and the amount of Internet traffic continues to grow. In a 2011 conference presentation, Intel projected that by 2015, the Internet will connect more than a billion people and more than 15 billion devices [5]. Among the factors fueling this growth is the recent explosion of Internet-enabled mobile devices like smartphones and tablets (such as those powered by iOS, Android, and Windows Phone operating systems) [6], televisions, and incar entertainment systems into the marketplace. A recent news article presented an estimate by Intel that a new server needs to be deployed for every 120 tablets or 600 smartphones hitting the market [7].

These trends strongly motivate the need to design a stream authentication scheme that incurs low overheads for signing (to reduce server load), communication (to reduce bandwidth consumption), and verification (to ensure mobile devices can cope with the additional processing). In this paper, we focus on the problem of efficient stream authentication using digital signatures. The goal is to provide integrity, origin authentication, and nonrepudiation for individual blocks that comprise a digital stream. The problem. Efficient authentication of stream poses several challenges:

1. Authentication of delay-sensitive streams requires high verification rates which translates to requiring minimal computational overhead to verify individual blocks and avoiding excessive accumulation of data in buffers before verification can proceed. For instance, to maintain jitter-free playback of ondemand media distributed through CDNs, per-block verification rates at client devices must equal or exceed the rate at which blocks arrive at the device.

2. For real-time generated digital streams, a sender must be able to sign a block as soon as it is generated with minimal computational overhead. For instance, to prevent delays in distribution of real-time content like stock quotes and live news feeds that can influence critical business decisions, per-block signing rates at content originators must exceed the rate at which blocks are generated.

3. Stream transmission is typically done using unreliable transport protocols like UDP to provide a high throughput, which can cause loss of datagrams during transmission. Thus, stream authentication mechanisms must be designed to tolerate arbitrary loss of data blocks without affecting the ability of a receiver to verify remaining blocks.

4. Authenticating information such as signatures and hash values placed within a block (which we call perblock communication overhead) must be limited to a small, constant size to prevent excessive bandwidth utilization while transmitting signed streams.

II. RELATED WORK

In this section, we discuss related work on trapdoor hash functions and stream authentication schemes. Trapdoor hash functions. Trapdoor hash functions are collision-resistant hash functions containing a trapdoor for finding collisions. A trapdoor hash function is associated with a (private, public) key pair, also referred to as a (trapdoor, hash) key. Collisions are difficult to find without the knowledge of the trapdoor key. However, given the trapdoor key along with the trapdoor hash on a message, it is feasible to find a collision. The concept of a trapdoor hash function was originally derived from the notion of trapdoor commitments proposed by Brassard et al. [9]; Krawczyk and Rabin [10] used trapdoor hash functions (referred to as chameleon hash) to construct a noninteractive nontransferable signature scheme, called chameleon signatures (closely related to undeniable signatures), under the hash-and-sign paradigm. Chameleon signature allows a signer to undeniably commit to the contents of a signed document, but does not allow the recipient of the signature to disclose the signer's commitment to a third party without the signer's consent. Shamir and Tauman [8] employed trapdoor hash functions to develop a new paradigm, called hash-sign-switch, that can be used to convert any signature scheme into an online/offline signature scheme [11]. In online/offline signature schemes, the signature generation procedure is split into two phases that are performed offline (before the message to be signed is known) and online (after the message is known). Ateniese and de Medeiros [12] introduced the first identity-based trapdoor hash functions and designed a novel sealed-bid auction scheme based on the identity-based chameleon signatures, where all bid remain hidden until the auction ends.

Later, Chen et al. [13] introduced the problem of key exposure in existing trapdoor hash functions and presented a trapdoor hashing

scheme in the gap Diffie-Hellman group with bilinear pairings to solve the problem of key exposure. The key exposure problem results in the compromise of the (private) trapdoor key in the presence of a pair of messages with the same trapdoor hash value. Later, Ateniese and de Medeiros [14] presented additional constructions of key exposure-free trapdoor hash functions that were based on more efficient cryptographic primitives compared to pairing-based operations. More recently, Mehta and Harn [15] introduced the idea of using trapdoor hash functions to build one-time proxy signatures [16] by exploiting the key-exposure property of trapdoor hash functions. Chandrasekhar et al. [17] presented a provably secure generic technique to build proxy signature schemes using trapdoor hash functions in the random oracle model. Unlike the scheme by Mehta and Harn, the technique proposed by Chandrasekhar et al. did not restrict the number of signatures a proxy can generate on behalf of the delegator. Stream authentication.

Researchers have proposed several techniques for stream (or flow) authentication that aim at reducing the computation and communication overhead associated with securing individual blocks that comprise a stream. These techniques can be divided into MAC-based schemes like TESLA [18] (and its variants) and signaturebased schemes like EMSS [18], AC [19], SAIDA [20], and WL [21]. While TESLA is efficient and robust against data loss, it requires time synchronization between a signer and a verifier, sufficiently large buffers of all unverified blocks (until the verification key is received), and storage of long key chains which can lead to scalability issues. This makes TESLA less suitable for authenticating stream, and vulnerable to DoS attacks that cause buffer overflow.

To reduce per-block overhead, signature-based stream authentication techniques either rely on amortizing a single signature over multiple blocks or designing extremely fast signature schemes like k-time signatures [22], BiBa [23], HORS [24] and TV-OTS [25] to sign each block. Designing extremely fast signature schemes often come at the cost of unreasonably high storage and communication overheads [20], [25], [21] that tend to increase linearly with the size of the message that is signed. For instance, the HORS scheme (considered more efficient than BiBa) generates signatures between 2,560 bits and 3,200 bits with public keys that are about 10 KiB with reasonable sized parameters [24], [25]. Moreover, BiBa and HORS require frequent redistribution of new public keys (that are very large) to maintain the security of the scheme, adding significant overheads to the communication and storage costs [25]. The TV-OTS scheme avoids some problems of the BiBa and HORS schemes, but still requires large public keys on order of 10KiB, requires time synchronization and does not provide long-term nonrepudiation as the signatures can be forged by the receiver after a reasonable effort [25].

To amortize a signature over multiple blocks in a stream, EMSS and AC use hash chains, SAIDA splits the signature and hash value of each block over multiple blocks, and WL uses Merkle trees. EMSS, AC, and SAIDA are probabilistic authentication schemes, i.e., the ability of a receiver to verify a received block depends on whether the receiver has some additional blocks of the stream in its possession (which inherently requires a verifier to maintain a buffer with multiple data blocks)—

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

thus, the probability that a receiver is able to verify a block depends on the nature (bursty, independent, etc.) and probability of data loss during stream transmission. EMSS and AC rely on redundant placement of multiple hashes in each block to deal with blocks that are lost during transmission. SAIDA relies on erasure codes to recover from losses. In probabilistic authentication schemes, maintaining a reasonable probability of verification in the presence of high data loss leads to higher per-block communication overhead and increased size of verifier-side data buffer. If per-block communication overhead is restricted, verification probability drops as data loss increases.

The WL scheme is the only known deterministic stream authentication protocol. In the WL scheme, a stream is divided into segments, with each segment containing multiple blocks (in WL each block is a single packet). The computational overhead at the signer and verifier, the signer's buffer size, and the per-block communication overhead are highly dependent on the segment size. When segment sizes are small, computational overhead at the signer and verifier increases. On the other hand, large segment sizes causes the signer's buffer size and the perblock communication overhead to increase. With a reasonable block size (say, 16 [21]), the signer-side delay increases (affecting real-time performance) along with the per-block communication overhead (to a magnitude of 100's of bytes). Recently, Lysyanskaya et al. [26] proposed a stream authentication technique, AECC using error correcting codes that is provably secure in a formal adversarial network model that limits the capabilities of an adversary to inject and delete packets by discrete quantities. The AECC scheme only requires one signature operation for the entire stream and adds only a constant size authentication overhead per packet, however, requires the sender to possess the entire stream before signing. This limits the application of the AECC scheme to delay-sensitive content and cannot be applied to real-time generated content.

III. PROPOSED SYSTEM

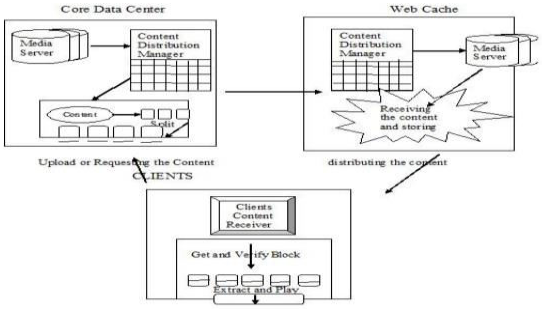

Fig. 1: PROPOSED SYSTEM

Fig 1 shows system architecture of content distribution network. The components include the core data center, web cache and it serving the multiple clients and the back end of the content distribution network is internet or wan and the data centers. The caches should be distributed widely and serving the requested clients. Both the core data centers and web caches contains media server and the content distribution manager.

The media content should be stored in the media servers and it should serve the content in both the real time as well as on demand users. Clients can include laptops, tablets and mobile phones. A content distribution manager has the following functionalities

- Tracking or auditing or monitoring the content usage by the clients and accounting the usage of their respective clients.

- The contents requested by the clients should be fetched from the media server and partition the file into multiple blocks and transmitting the requesting blocks into appropriate clients.
- If the client requested the digital media content the request should be sent to closest web cache of the client. If the web cache contains the request which will be ask by the client it should be fetched and transmits to the client.
- If the request is not in the web cache means the request should be forwarded to the core data center and fetches the data and should transmit to the appropriate client.

A. CONTENT UPLOADING: Server should upload the multimedia content was given by the content provider and store in a media Server. The client can also be allowed to upload the multimedia content after the registration process is done.

B. STREAM AUTHENTICATION: Stream authentication can help prevent some type of attacks by providing the ability to sign and verify each block in the stream. All content originates at the core data center and the stream signing mechanism is implemented at the core CDM as part of its content processing service. We assume the existence of a Public Key Infrastructure (PKI) responsible for generating certificates for the core CDM, and distributing the public key and certificate of the core CDM to all verifying entities. When a request arrives at the core CDM, the content processing service retrieves the content from the media server. The core CDM then splits the content it into a stream of blocks, signs each block (using a suitable signature amortization technique), places the authentication information within the block, and transmits the signed stream of blocks to the requesting entity. If the content is not generated in real time, the content processing service stores the signed stream at the media server to prevent redundant signing operations when subsequent requests arrive for the same content.

C. STREAM VERIFICATION: Threats involved in distribution of content include: Compromising attacks, where an adversary takes control of legitimate content providing hosts (edge cache/core data center/third-party provider) to inject malicious content, and Man-in-the-middle attacks, where an adversary performs modification of content during transmission from core data center to the edge cache or from the edge cache to the client or from the core data center to the client. Verification of signed streams at edge caches ensures that packets failing verification are not forwarded to the requesting client, thereby preventing unnecessary usage of bandwidth and processing time at the client machine. When a signed stream arrives at the client machine, the requesting application verifies each block in the stream and removes the authenticating information placed inside the block before beginning playback of the media content.

To allow users to be timely and accurately informed about their data usage, our distributed logging mechanism is complemented by an innovative auditing mechanism. We support two complementary auditing modes: 1. Push mode, 2. Pull mode.

1. PUSH MODE: In this mode, the logs are periodically pushed to the data owner (or auditor) by the harmonizer. The push action

will be triggered by either type of the following two events: one is that the time elapses for a certain period according to the temporal timer inserted as part of the JAR file, the other is that the JAR file exceeds the size stipulated by the content owner at the time of creation. After the logs are sent to the data owner, the log files will be dumped, so as to free the space for future access logs. This mode serves two essential functions in the logging architecture: a. It ensures that the size of the log files does not explode b. It enables timely detection and correction of any loss or damage to the log files.

2. PULL MODE: This mode allows auditors to retrieve the logs anytime when they want to check the recent access to their own data. The pull message consists simply of an FTP pull command, which can be issues from the command line. For naive users, a wizard comprising a batch file can be easily built. The request will be sent to the harmonizer, and the user will be informed of the data's locations and obtain an integrated copy of the authentic and sealed log file.

IV. CONCLUSION

The authentication flow in the content distribution network prevents malicious modification or threats in the middle of the data transmission. The challenging task is to verification and signing for the on demand content and the toleranceagainst the transmission loss and the communication overhead should be small per block. We present the authentication of online digitized signature using trap door hash function method that challenges that meet real time streaming in content distribution and provide efficient authentication of delay sensitive streams. Our the authentication of online digitized signature method by authenticating from initial blocks in the stream using signature on the trap door hash function and by authenticating subsequent blocks in the stream.

V. REFERENCES

[1] B.M. Luettmann and A.C. Bender, "Man-in-the-Middle Attacks on Auto-Updating Software," Bell Labs Technical J., vol. 12, no. 3, pp. 131-138, 2007.

[2] Akamai, "Akamai Information Security Management System Overview: Securing the Cloud," White Paper, http://stag-wwwweb01.akamai.com/dl/whitepapers/Akamai_ ISMS.pdf?campaig n_id=AANA-65TPAC, 2012.

[3] P. Bright, "Google, Microsoft Distribute Malware After Domain Name Trickery," Ars Technica, http://arstechnica. com/security/ news/2010/12/google-microsoft-distribute-malware-after-domain-name-trickery.ars, 2010.

[4] A. Gonsalves, "YouTube Confirms Justin Bieber Hack Attack," InformationWeek, http://www.informationweek.com/news/ security/attacks/showArticle.jhtml?articleID= 225702490, 2010.

[5] K. Skaugen, "Cloud 2015," Proc. Interop, http://www.interop.-com/lasvegas/2011/presentations/free/136-kirk-skaugen.pdf, 2012.

[6] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011-2016," White Paper,

http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.pdf, 2012.

[7] D. Grabham, "Intel: New Server Needed for Every 120 Tablets Sold," Techradar, http://www.techradar.com/news/computingcomponents/processors/intel-new-server-needed-for-every-120- tablets-sold-1069021, 2012.

[8] A. Shamir and Y. Tauman, "Improved Online/Offline Signature Schemes," CRYPTO '01: Proc. 21st Ann. Int'l Cryptology Conf., pp. 355-367, 2001.

[9] G. Brassard, D. Chaum, and C. Cre´peau, "Minimum Disclosure Proofs of Knowledge," J. Computer and System Sciences, vol. 37, no. 2, pp. 156-189, 1988.

[10] H. Krawczyk and T. Rabin, "Chameleon Signatures," Proc. Network and Distributed System Security Symp. (NDSS), 2000.

[11] S. Even, O. Goldreich, and S. Micali, "Online/Offline Digital Schemes," CRYPTO: Proc. Ninth Ann. Int'l Cryptology Conf., pp. 263-275, 1989.

[12] G. Ateniese and B. de Medeiros, "Identity-Based Chameleon Hash and Applications," Proc. Eighth Int'l Conf, Financial Cryptography (FC), pp. 164-180, 2004.