TrustStream: A Novel Secure and Scalable Media Streaming Architecture

Hao Yin[1], Chuang Lin[1], Feng Qiu[1], Xuening Liu[1], Dapeng Wu[2]
[1]Computer Science and Technology Department, Tsinghua University, Beijing, 100084, P.R.China
{hyin,clin,fqiu,liuxn}@csnet1.cs.tsinghua.edu.cn
[2]Department of Electrical & Computer Engineering, University of Florida, USA.
wu@ece.ufl.edu

ABSTRACT

Streaming media over networks has gained renewed interest recently due to the emerging IP-TV and mobile TV. The success of commercial media streaming systems critically depends on two important capabilities, namely, 1) scalability in distributing media content to diverse clients, and 2) security management of the media and the systems. However, existing media streaming systems such as content distribution networks (CDN) and Peer-to-Peer (P2P) networks lack either security or scalability. In this paper, we propose a novel secure and scalable media streaming architecture, called TrustStream. Our architecture combines the best features of CDN and P2P networks to achieve unprecedented security, scalability, and certain quality of service simultaneously. Our experimental results demonstrate the advantages of the TrustStream.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: *Security and protection*

General Terms

Design, Security, Algorithm

Keywords[1]

Streaming media, scalability, content distribution network, p2p

1. INTRODUCTION

Streaming media over networks has gained renewed interest recently due to the emerging IP-TV and mobile TV. To build large scale commercial media streaming systems and networks, scalability and security issues must be addressed. The current solutions for media streaming can be classified into two categories, namely, content distribution networks (CDN) and Peer-to-Peer (P2P) networks. CDN can provide security management and quality of service (QoS) but at the cost of high load on servers in CDN; hence, CDN is not scalable for very large number of clients. On the other hand, P2P networks only incur low load on each host and hence it is scalable; but it is very hard to provide security and QoS in P2P networks. To address scalability, QoS, and security issues for media streaming, we propose a novel secure and scalable media streaming architecture, called TrustStream. Our architecture combines the best features of CDN and P2P networks; i.e., TrustStream is scalable and can provide QoS and security.

In TrustStream, we utilize the unique features of scalable video to design security management and transport mechanisms. The key idea is the following. We first employ Progressive Fine Granularity Scalability (PFGS) coding technique [5] to encode the video content into two layers of streams: one layer called "base layer", has very low bit rate and contains the most important information of the video content; and another layer called "enhancement layer", has much higher bit rate and only improves the playback quality at the client side. The enhancement layer can only be decoded when the base layer is available. In other words, the enhancement layer is useless for the client if the base layer is lost. With PFGS video coding, we obtain two video layers; then we apply security management to the base layer and distribute the base layer stream through our proposed scalable Secure Application Layer Multicast (S-ALM); and we use P2P networks to transport the enhancement layer. In this way, TrustStream can provide QoS and security while achieving scalability.

The contributions of this paper are that 1) we propose a novel architecture that utilizes the best features of both CDN and P2P networks; 2) we propose S-ALM that improves the performance of current ALM scheme and is able to address the following security issues:

- **Confidentiality:** To manage a cluster, we design an election and removal mechanism, by which cluster members can elect two trusted nodes as cluster leaders (one acts as a security manager and a media content source in this cluster, and another as a backup) and remove a cluster leader, in which cluster members lose confidence. With the elected cluster leaders, we implement the following security mechanism: in each cluster, the media content is encrypted with an encryption key by the cluster leader; only authorized users can get the encryption key from the cluster leader to correctly decode the base layer. In this scheme, the server does not need to carry out a global key update and distribution, which eliminates potential bottleneck at the server.
- **Authentication of messages and users:** We use a public key cryptosystem so that a cluster leader can verify the legitimacy of a user, and a legitimate user can authenticate the received content and security management messages.
- **Availability:** Our S-ALM utilizes the structure of clusters to achieve efficiency in security management. Specifically,

a P2P scheme is used to forward data within each cluster, and a hierarchical ALM scheme is employed for data forwarding between clusters. In this way, we can provide almost the same level of availability as P2P networks. In addition, backup leaders in clusters improve the availability.

The rest of the paper is organized as follows. Section 2 overviews the related work. Section 3 describes our design of TrustStream in details. In Section 4, we present our experimental results to demonstrate the performance of the TrustStream. Section 5 concludes the paper.

2. RELATED WORK

2.1 Scalable Video Coding

In the video coding community, layered video coding is often referred to as *scalable coding* [4]. Examples include the *Fine Granularity Scalability* (FGS) [3] and *Progressive FGS* (PFGS) coding [5]. With scalable video coding, a client can regulate the receiving rate of video streams by adding/dropping layers to cope with fluctuations in available bandwidth, while the server does not participate in rate control, resulting in reduced load on the server. In TrustStream, a simplified PFGS coding is used.

2.2 Secure Application-layer Multicast

Confidentiality in multicast is usually achieved by encrypting the content using an encryption key, known as the session key (SK), which is known only by the content provider and all legitimate group members [2]. However, it is not an easy task to securely deliver the SK to all the members because the group membership is dynamic due to clients' joining and leaving the group from time to time. Note that, if a client is not a legitimate member during certain period, e.g., before he joins or after he leaves the session, he should not be able to decrypt the media content. In other words, the SK needs to be updated once a client joins or leaves the session. Hence, the key distribution scheme for multicast should have the following three security properties: 1) Forward Secrecy: to ensure that an expired member does not receive the new SK after he leaves the group. Note that anyone is allowed to receive the encrypted multicast data at anytime. 2) Backward Secrecy: to ensure that a new member does not receive the SK used before he joins. 3) Collusion: to prevent expired members from regaining access to a new SK through their joint efforts [2]. In the literature, several key distribution schemes for IP multicast have been proposed but none for application layer multicast (ALM) [1]. Although the existing key distribution schemes could potentially be modified for ALM, such modifications require imposing significant additional structure to the overlay and having the key distribution nodes possess lots of knowledge of the group members and/or the distribution tree. Furthermore, tree structure and neighboring relationships in ALM may change frequently, making the design of key distribution schemes a great challenge. To address these issues, we propose a new secure and scalable ALM for the TrustStream framework.

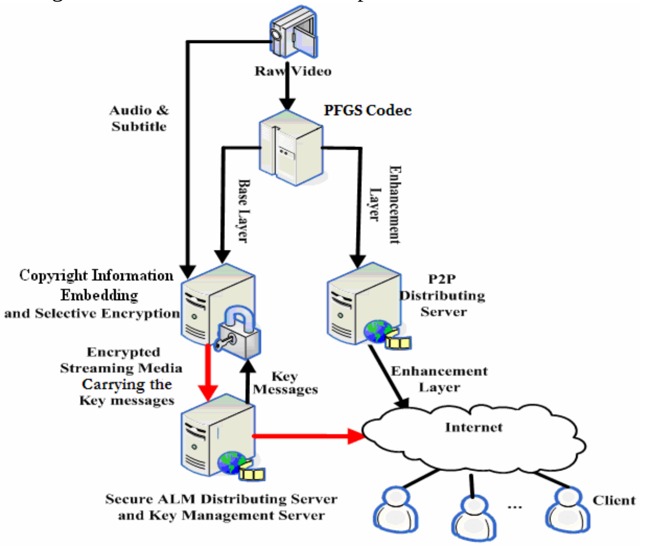
3. TRUSTSTREAM

In this section, we present the TrustStream. We first introduce the architecture, and then we present our mechanisms for application-layer multicast and security management.

3.1 Architecture

The TrustStream architecture is shown in Fig. 1. For the purpose of efficiency, we choose a simplified PFGS coder that generates two video layers, i.e., a base layer and an enhancement layer. To encrypt copyright information, we use our proposed video data embedding codec [6] to embed some copyright information in the base layer; to encrypt the video content, we apply a selective encryption [7] to the base layer. The encrypted base layer is transported to clients through the S-ALM framework along with session keys, while the enhancement layer is transported through P2P networks. The media coding and encryption are carried out by the server.

The key component of the TrustStream architecture is S-ALM, which consists of application layer multicast and security management mechanisms, which are presented next.



Raw Video
Audio & Subtitle
PFGS Codec
Base Layer
Enhancement Layer
Copyright Information Embedding and Selective Encryption
P2P Distributing Server
Encrypted Streaming Media Carrying the Key messages
Key Messages
Enhancement Layer
Internet
Secure ALM Distributing Server and Key Management Server
...
Client

Fig.1 Architecture of TrustStream

3.2 Application-layer Multicast

Similar to NICE [9], we organize the multicast members into a hierarchy as shown in Fig. 2. The hierarchy consists of K layers (K=2 in Fig. 2). Denote Layer k by L_k. The source of the hierarchy is *Server*, which sends data packets directly to all the members of layer L_0. Members in each layer are further organized into clusters to achieve efficient management (it is easier to manage a smaller group of members). Denote a cluster on L_k by C_k. Note that there is only one cluster on L_0. Each cluster has a cluster leader and a backup leader. The cluster leader on L_k is also a member of the corresponding cluster in the upper layer L_{k-1}. The backup leader becomes a cluster leader when the incumbent cluster leader leaves.

To maintain the topology of a cluster, each member in the cluster exchanges maintenance-messages periodically with other members in the same cluster. Since a cluster leader on L_k is also a member of the corresponding cluster in the upper layer L_{k-1}, hence maintaining each cluster is equivalent to maintaining the hierarchy; in other words, we do not need to explicitly maintain the connection between two adjacent layers.

Now, we describe how to forward media data. First, the *Server* sends media data to all the members on L_0. Since some members on L_0 are also cluster leaders on L_1, these cluster leaders will forward the media data to all other members in the clusters on L_1; the data forwarding from a cluster leader to its members uses a P2P technique [8]. This process repeats for the nodes on L_{k-1}, which are also cluster leaders on L_k. In this way, media data can be conveyed to all the nodes in the hierarchy.

Next, we describe how to elect/remove a cluster leader and how to join and leave a cluster, respectively.

Election of a cluster leader: A node that has the best computing performance and/or the highest network bandwidth and/or the minimum average round trip time (RTT) to other cluster members, is elected as the cluster leader. The second best node in the cluster is elected as a backup leader.

Removal of a cluster leader: If another member *h* in the cluster has a better performance (in terms of computing performance, bandwidth, or RTT) than the incumbent cluster leader, node *h* can replace the incumbent leader to improve the system performance.

Join: When a new member tries to join the multicast tree, it first sends a request of joining C_0 to the *Server*. If the *Server* accepts this request (according to the link status of the *Server*), the new member joins C_0; otherwise, the *Server* sends a list of recommended cluster leaders on L_1 to the new member. Then, the new member sends joining requests to the recommended cluster leaders. If a cluster leader on L_1 rejects the request, the leader sends a list of recommended cluster leaders on L_2. If none of the leaders on L_1 accepts the request, the new member sends joining requests to the recommended cluster leaders L_2. This process repeats till the new member is accepted by a cluster leader.

Leave: When a member leaves the multicast tree, it sends Leave Message to the *Server* and other members in the same cluster. Then the remaining members re-organize the cluster. If the departing member is a cluster leader, the backup leader will become the new cluster leader and join the corresponding cluster on the higher layer.

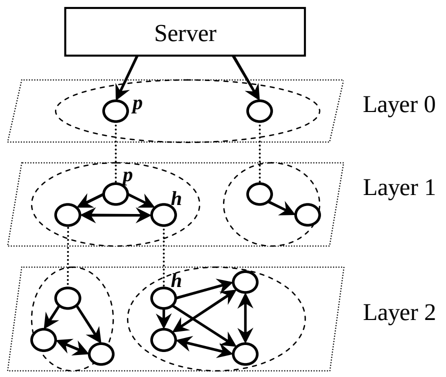


Fig. 2 Data forwarding in our ALM protocol

3.3 Security Management

Table 1 lists the abbreviations used in the paper. Next, we describe how our secure ALM (S-ALM) addresses the three security issues.

Confidentiality: Confidentiality is achieved through encryption, which requires encryption key management. There are two types of keys, i.e., session key (*SK*) and cluster key (*CK*). A *SK* is used to encrypt the streaming media while a *CK* is used to encrypt a *SK*. Next, we describe how to generate and distribute *SK* and *CK*.

<u>*Generation and distribution of SK*</u>: *KMS* generates a new *SK* periodically. The new *SK* is encrypted by *KMS* with CK_0 (Cluster Key of C_0) and sent to all the members in C_0. Then the *SK* is distributed through the data paths as shown in Fig.2. Specifically, when a node *h* on L_{k-1} receives the encrypted *SK* from a node *p* (obviously *h* and *p* are in the same cluster), *h* decrypts the *SK* with the *CK* shared with *p*. If *h* is also a leader on L_k, then *h* re-encrypts the *SK* with the *CK* shared by the cluster on L_k and forwards the encrypted *SK* to the members in the cluster on L_k.

<u>*Generation and distribution of CK*</u>: A cluster leader updates *CK* when cluster members change; the new *CK* is distributed to cluster members through secure channels. When a new member joins the cluster, it needs to establish a secure channel with the leader (possibly through a public key cryptosystem). When the cluster leader leaves, the backup leader needs to establish secure channels with cluster members, and then generates a new *CK* and distributes it.

Table 1 Nomenclature

SK	Session Key	Shared by all the multicast group members and *KMS*; used to encrypt/decrypt streaming media content; generated by KMS periodically
CK	Cluster Key	Shared by each cluster; used to encrypt/decrypt *SK*; generated by the cluster leader
KMS	Key Management Server	Generates *SK*; Distributes *SK*; Manages C_0
SC	Secure Channel	Established by using Public key and Private key between members and leader

Integrity: Every member has a public key and a private key; the public key is also maintained in the *KMS*. When a member sends some message, it appends its digital signature to the message. When the receiver receives the message, it can authenticate the message by the public key of the intended sender, which can be obtained from the *KMS*.

Availability: Since the encrypted *SK* is distributed over an error-prone network along with the streaming media, it is important to make the *SK* distribution reliable. In TrustStream, we provide a backup leader in each cluster to improve reliability. If a leader leaves or collapses, the backup leader can easily join the higher layer and get the media content by using the backup information, which is periodically updated by the leader.

Although there are some similarities between NICE [9] and our S-ALM, we would like to point out the key differences between NICE and S-ALM. First, NICE uses unicast to convey messages from a cluster leader to its members, while S-ALM uses P2P. Second, NICE elects a cluster leader based on relative distance (number of hops) between the members, while S-ALM elects a cluster leader based on QoS and computing capability since a cluster leader in S-ALM needs to be computationally powerful to perform encryption, and be reliable to provide good QoS. Third, when a node joins or leaves a cluster, S-ALM maintains the network topology based on QoS and computing capability, while NICE does not. Fourth, S-ALM provides security while NICE does not.

4. PERFORMANCE EVALUATION

4.1 Source Coding

We have implemented a prototype of our system and conduct a series of performance evaluations on a PIII-800MHz PC. As shown in Fig. 3, although PFGS does not achieve as good coding efficiency as non-scalable video coding (denoted as “Single Layer”), PFGS still provides acceptable quality. Fig. 4 shows the effect of selective encryption that is applied to the base layer of a compressed video sequence “foreman” in CIF format. The right figure of Fig. 4 is obtained by decoding the combination of the encrypted base layer and non-encrypted enhancement layer. It can be seen that the right figure of Fig. 4 is not recognizable.

We also conduct experiments to see how much the video encoder/decode slows down due to encryption. From Table 2, we can see that the computation overhead incurred by encryption is very low and can be ignored.

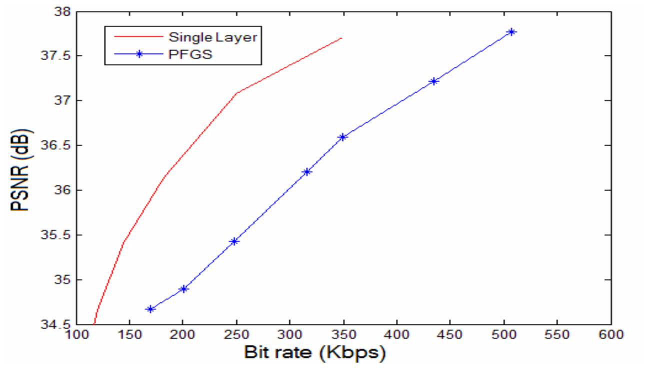
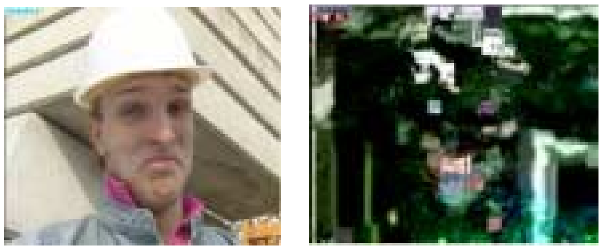


Fig.3 Performance of non-scalable video coding and PFGS on “foreman” video sequence in CIF format



Fig, 4 Video quality under selective encryption

Table 2 Computation overhead of encryption

	Without Enc/Dec	With Enc/Dec	Increased overhead
Encoding	50.4 fps	49.1 fps	2.64%
Decoding	112.8 fps	112.2 fps	0.54%

4.2 S-ALM

Now we analyze the overhead incurred by the S-ALM mechanisms proposed in Section 3.

Number of secure channels. To securely distribute cluster keys in a cluster, every member in the cluster must maintain a secure channel with its cluster leader. By simple calculation, the total number of secure channels in a multicast tree is n, where n is the total number of nodes in the multicast tree.

Overhead for *CK* updating. When a member in a cluster joins/leaves, the number of *CK* update messages is $O(J)$, where J is the number of members in the cluster. Since the size of each cluster is small enough and independent of the total number of nodes in the multicast tree, hence $O(J)$ can be regarded as $O(1)$.

Overhead for *SK* re-encryption. The *SK* needs to be decrypted and re-encrypted at each cluster leader. Therefore, the total number of re-encryption operations is $O(n/j)$, where j is the average number of members in a cluster.

Maintaining message. Maintaining message is used to maintain a cluster and the hierarchical architecture. Each member exchanges messages with other members in the same cluster. Hence, the total number of messages is $O(J^2)$. Since J is small and independent of n, hence $O(J^2)$ can be regarded as $O(1)$.

5. CONCLUSION

Streaming video over networks has received tremendous attention from academia and industry due to the explosive growth of the Internet and people's enormous demand for multimedia content. The success of commercial media streaming systems critically depends on 1) scalability in media distribution and 2) security of the media and the systems. However, existing media streaming systems such as content distribution networks and P2P networks lack either security or scalability. In this paper, we propose a novel secure and scalable media streaming architecture, called TrustStream. Our architecture combines the best features of content distribution networks and peer-to-peer networks to achieve unprecedented security, scalability, and certain quality of service simultaneously. Our experimental results demonstrate the advantages of the TrustStream.

6. Acknowledgment

The research was support in part by grants from NSF China under the contracts 60429202, 60372019, 60473086 and 60432030.

7. REFERENCES

[1] C. Abad, I.Gupta, and W. Yurcik. Adding Confidentiality to Application-Level Multicast by Leveraging the Muticast Overlay, *IEEE International Workshop on Assurance Distributed Systems and Network*, 2005.

[2] K. Chan, S.-H. Gary Chan. Key management approaches to offer data confidentiality for secure multicast. *IEEE Network*, September/October 2003. pp. 30-39.

[3] W. Li. Overview of Fine Granularity Scalability in MPEG-4 Video Standard. *IEEE Trans. on Circuit and Systems for Video Technology*, Vol. 11, No. 3, Mar. 2001.

[4] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. Peha. Streaming video over the internet: approaches and directions. *IEEE Trans. Circuits System Video Technology*, vol. 11, no. 3, pp. 282–300, Mar. 2001.

[5] S. Li, F. Wu, and Y.-Q. Zhang. Experimental results with Progressive Fine Granularity Scalable (PFGS) Coding. *ISO/IEC JTC1/SC29/WG11, MPEG99/m5742*, Mar. 2000.

[6] H. Yin, F. Qiu, C.Lin, H. Zou, A Key Embedded Video Codec for Secure Video Multicast, 9th IFIP TC-6 TC-11 Conference on Communications and Multimedia Security, Lecture Notes in Computer Science, Springer, 2005.

[7] H. Yin, C. Lin, F. Qiu, B. Li, Z. Tan, “A Media-Dependent Secure Multicast Protocol for Adaptive Video Applications”, *in Proceeding. of ACM SIGCOMM Asia Workshop, Beijing, China, 2005.*

[8] X. Zhang, J. Liu, B. Li, and T.-SP Yum. CoolStreaming / DONet : A Data- driven Overlay Network for Live Media Streaming. In Proceeding of *IEEE INFOCOM'05, Miami, FL, USA, March 2005.*

[9] S. Banerjee, B. Bhattacharjee, and C. Kommeareddy. Scalable application layer multicast. *In Proceedings of ACM Sigcomm,* August 2002.