Group-Based Peer-to-Peer 3D Streaming Authentication

Mo-Che Chan, Jehn-Ruey Jiang, Chao-Wei Hung
Department of Computer Science and Information Engineering
National Central University, Jhongli, Taiwan, ROC

Wei Tsang Ooi
Department of Computer Science
National University of Singapore, Singapore

Abstract**—In this paper, we present an authentication scheme for group-based peer-to-peer 3D streaming that takes advantage of secure group communication to reduce computation overheads of authentication. Users of the same interest first form a group, and one of the users is then elected to be the trusted leader, who is to download 3D contents, to verify their authenticity and integrity, and to send group members a checksum value for each 3D content piece via group secure channels. By the encrypted checksum values, a group member can authenticate 3D contents downloaded from any source. Since checksum is cheap to compute, much computation is saved. We have evaluated the computation saving of the proposed scheme for the case of progressive meshes based on the hash chain signature 3D streaming authentication scheme. We have also evaluated the rendering quality when embedding hash chain signatures into the least significant bits (LSBs) of the mesh data to make the signatures imperceptible.**

Keywords**-peer-to-peer; 3D streaming; authentication;**

I. Introduction

3D streaming refers to the continuous and real-time delivery of 3D contents (e.g., meshes, textures, etc.) over networks to allow user interactions without a prior full download. 3D streaming is applying to virtual environments (VEs) [1], such as massively multiplayer online games (MMOGs), because VE pre-installations via CD/DVD or a prior download are unpractical, for the contents are large and dynamic (e.g., Second Life [2] hosts 34 terabytes of user-generated dynamic contents in 2007). Similar to audio or video media streaming, 3D contents need to be fragmented into pieces at a server, before they can be transmitted, reconstructed, and displayed at the clients. However, unlike media streaming, because users accessing 3D contents often have different visibility or interests, transmission sequence in 3D streaming thus varies from user to user and requires individualized visibility calculations.

Current 3D streaming schemes can be classified into four main types: *object streaming*, *scene streaming*, *visualization streaming*, and *image-based streaming*. In this paper, we focus on scene streaming and object streaming together; when we mention 3D streaming in the following context, we refer to object streaming and scene streaming. A virtual environment scene usually involves a collection of 3D objects placed arbitrarily in space that are streamed to clients according to object visibility and/or user interests. As many more objects may exist than what a user can see at a moment, scene streaming techniques in general determine the object download priorities dynamically and then download the objects for the user according to the user's ever-changing position and angle of view as well as objects' importance. Object streaming techniques, such as *progressive mesh streaming* [3], can then be used to help download objects progressively. For example, in progressive mesh streaming, objects are represented as *progressive meshes*, in which a *base mesh*, the lowest level-of-detail version of the mesh, can first be sent to a user, and then a series of refinements, called *vertex splits*, can be delivered to improve the level-of-detail of the mesh. In this way, a user can quickly view a coarse version of the mesh, trading off between waiting time and mesh quality. Further, a user may stop the transmission if it looses interest in the mesh or finds the quality of the intermediate mesh good enough.

3D streaming can be based on the client-server model, as used in Second Life [2]. This model is easy to design; however, it is hard to scale due to prohibitively vast amount of server-side bandwidth required for a massive audience. On the other hand, 3D streaming can be based on the peer-to-peer (P2P) model, as used in FLoD [4], [5]. In this model, clients of 3D streaming applications can obtain 3D contents from other clients instead of the server. Since clients navigating nearby virtual environment regions usually share similar 3D contents, most contents can be downloaded from clients and the server bandwidth consumption is reduced dramatically. P2P 3D streaming can thus achieve very high scalability.

As 3D contents may be obtained from clients in P2P 3D streaming, the need to verify the integrity and authenticity of the the contents arises. Integrity verification and authentication are particular important in applications such as online shops or auction, where a user needs to ensure that the 3D model of an item he or she is interested in is indeed from a trusted source and has not been tampered with. One method to achieve this is to generate a *message authentication code* (MAC), taking as inputs the content to verify and a secret key known only to the content owner and receiver. The MAC is then sent together with the content. The receiver, using the content received and the secret key, can recompute the MAC and verify it against the MAC received. Since MACs are computed and verified with the same secret key, they can be verified only by intended recipients who possess the key beforehand. Another method for content authentication is to make the content owner sign a *digital signature* (DS), which is a piece of information based on both the content and the owner's private key. The receiver can verify the content by the owner's public key, which is publicly accessible, so the DS scheme is suitable for public authentication [6]. Some watermarking methods [6], [7] were proposed for authenticating 3D contents. They are based on concepts similar to either the DS or the MAC. For example, Wu and Cheung proposed a public authentication method to imperceptibly embed the owner's signature into the 3D content by "stealing" least significant bits (LSBs) of floating point numbers representing the content.

The nature of 3D streaming, however, introduces a new complication in authentication. Suppose we generate a single DS (or MAC) for the whole 3D contents, then the 3D contents can only be authenticated when the they are completely received. It is crucial, however, for users to be able to authenticate the contents progressively (i.e., to authenticate the 3D contents as they are being received), since 1) we want to detect any malicious tampering early, and 2) not all users wait to receive the whole 3D contents. For the case of 3D mesh objects, a naive solution for progressive authentication is to generate a DS (or MAC) for the base mesh and for each of the refinement sent. This solution, however, is expensive. In [8], the hash-chain and Rabin signature schemes are proposed to decrease the cost of signing digital signatures.

In this paper, we present an efficient solution for progressive authentication of P2P 3D streaming that takes advantage of secure group communication to further save computation. Users in virtual environments usually form a group to perform specific actions. For example, friends might visit a virtual shopping mall together. One of the users of the group is assumed to be the trusted leader, who is to download 3D contents and verify their authenticity and integrity by a specified 3D streaming progressive authentication scheme mentioned above. The leader then sends group members a checksum value for each 3D content piece via group secure channels. A group member can download 3D contents from any user, even from non-group members, and can authenticate the downloaded 3D contents by the encrypted checksum values received from the leader. Since checksum is cheap to compute, much computation is saved. We will evaluate the computation saving of the proposed scheme for the case of progressive meshes based on the hash-chain signature 3D streaming authentication scheme which is so far of the lowest computation cost to the best of our knowledge. We will also evaluate the rendering quality when embedding the hash-chain signature into the LSBs of the mesh data. We show that such embedding is imperceptible (i.e., it does not introduce significant distortion to the mesh), while at the same time, preserves the total size of the mesh.

The rest of the paper is organized as follows. In Section 2, we introduce some related work. We describe the proposed method in Section 3, and show its evaluations in Section 4. Finally, some concluding remarks are drawn in Section 5.

II. Related Work

In this section, we introduce some related work, including P2P 3D streaming, authenticated 3D streaming, progressive meshes, imperceptible public 3D mesh authentication, and secret key exchange. The notations used in this section and in the rest of this paper are listed in Table I.

Table I
Notations

$peer_i$	the peer/member i
PK_i	the $peer_i$'s public key
$H(x)$	hash operator with input message x
$E_{sk}[x]$	message encrypting x by symmetric key sk
$\{x\}_{PK}$	message encrypting x by asymmetric key PK
$x\|y$	concatenation of the messages x and y
c_j	checksum of 3D data piece D_j
t_j	timestamp of c_j
ct_j	checksum of c_j and t_j
$A \rightarrow B : x$	peer A sends message x to peer B.
$x \overset{?}{=} y$	to check if value x is equal to value y
$x \overset{?}{<} y$	to check if value x is small than value y

A. Peer-to-Peer 3D Streaming

FLoD [4], [5] is the first P2P 3D streaming framework that partitions the VE into rectangular cells, and specify *scene descriptions* (i.e., files containing lists of objects within each cells) for object discovery. It relies on the recent research of P2P virtual environment (P2P VE) [**?**], [9], [10], where a 2D spatial overlay returns a list of nearby users within the *area of interest* (AOI) for the discovery of content sources. The returned neighbors are called *AOI neighbors*. Once a navigating user obtains a list of AOI neighbors, the user can then send queries to these AOI neighbors to exchange states on scene content availability, and request the AOI neighbors to exchange contents. The server is contacted only if no neighbors have the relevant contents. As the query-response approach to inquire content availability may be slow, a follow-up work of FLoD [11] adopts an alternative strategy where peers would actively push content availability to their AOI neighbors to reduce time for state exchange. Additional AOI neighbors are also maintained to increase the potential pool of source peers who could provide contents.

B. Authenticated 3D Streaming

Authenticity of progressive mesh streaming can be confirmed if a digital signature is generated for each piece of 3D streaming data. However, to generate digital signature for each piece is not practical because public key cryptosystem consumes a lot of computation power. Some efficient stream signing mechanisms [8], [12] are proposed to avoid generating digital signature for each piece. Below in this subsection, we introduce a protocol proposed in [8] using hash chains to achieve very low computation cost.

$$\begin{aligned}
\mathbb{D}_n &= H(D_n)\\
\mathbb{D}_{n-1} &= H(D_{n-1}\|\mathbb{D}_n)\\
\mathbb{D}_{n-2} &= H(D_{n-2}\|\mathbb{D}_{n-1})\\
\ldots &= \ldots\\
\mathbb{D}_1 &= H(D_1\|\mathbb{D}_2)\\
\mathbb{D}_0 &= H(meta\|D_0\|\mathbb{D}_1)\\
\Delta_D &= Sign_{sk}(\mathbb{D}_0)
\end{aligned}$$

Figure 1. Production rules for hash chain-based 3D streaming authentication

A 3D object, consisting of both mesh and texture data, can be treated logically as a *base piece* plus many *refinement pieces* [5], where each refinement piece depends on the previous piece. When a 3D object owner publishes 3D data, he/she first divides the data into $D_0, D_1, D_2, \ldots, D_n$, where D_0 is the base piece and $D_1, D_2, D_3, \ldots, D_n$ is refinement pieces. The owner then computes, according to the production rules in Figure 1, a hash chain $\mathbb{D}_0, \ldots, \mathbb{D}_n$ out of those pieces and only signs $\mathbb{D}_0$. The owner then disseminates the digital signature Δ_D, the metadata $meta$ (e.g., the object's ID, owner, size, number of pieces, etc.) of the object, and $\mathbb{D}_0$, D_0, $\mathbb{D}_1$, D_1, ... as a data stream. Note that in order to verify the i-th piece D_i, the hash value $\mathbb{D}_i$ is sent before D_i. This is because $\mathbb{D}_{i+1}$ is required when verifying D_i, as shown below.

The authenticity and integrity of 3D contents can be progressively verified by the receiver receiving the stream Δ_D, $meta$, $\mathbb{D}_0$, D_0, $\mathbb{D}_1$, D_1, ... The receiver can first verify the main signature $\Delta_D(= Sign_{sk}(\mathbb{D}_0))$ by the owner's public key. The base piece D_0 is verified if $\mathbb{D}_0 = H(meta|D_0|\mathbb{D}_1)$, and the first refinement piece D_1 is verified if $\mathbb{D}_1 = H(D_1|\mathbb{D}_2)$. Likewise, the receiver can verify all subsequent pieces with just one hash operator. To sum up, this protocol allows a receiver to efficiently verify a sequence of 3D contents. Note that the protocol has the property of public authentication, since it utilizes the owner's public key, which is publicly accessible, as the basis of authenticating pieces.

C. Progressive Meshes

Progressive mesh, proposed by Hoppe in 1996, is one of the well-known techniques for representing level of details for a 3D mesh at fine granularity [3]. Given a non-progressive 3D mesh, the technique derives a coarser version of the mesh with fewer vertices by repeatedly merging two adjacent vertices. This operation is called *edge collapse*. Typically the operation is performed in a sequence that results in minimal distortion to the mesh (e.g., by collapsing shortest edge first). The final, simplified model obtained after the sequence of operations becomes the *base mesh*. From the base mesh, we can obtain the original mesh by performing the reverse operation *vertex split*, by repeatedly splitting the merged vertices, thereby restoring the edges that has been collapsed. A progressive mesh therefore consists of the base mesh, plus a series of vertex split operations.

To stream a progressive mesh, the base mesh is sent first, followed by a sequence of packets containing the vertex splits, allowing the receiver to render a simplified version of the mesh, before refining the quality of the rendered mesh by performing the vertex split operations received. Grouping of vertex splits into packets (or *packetization*) for progressive mesh streaming is addressed in [13].

D. Imperceptible Public 3D Mesh Authentication

Wu and Cheung proposed an authentication scheme to verify the authenticity and integrity of 3D meshes by public keys [6]. The basic idea of the scheme is to embed a digital signature within the 3D mesh so that only the public key that is publicly accessible is required for authentication. The scheme provides a convenient way to publicly authenticate 3D meshes. Furthermore, it inserts an individual digital signature to each 3D mesh piece so that a tampered mesh piece can be identified precisely.

Wu and Cheung's scheme also proposes a way to embed digital signature within 3D meshes imperceptibly. It assumes that each vertex of a 3D mesh is a coordinate of x, y and z values represented by floating point numbers. According to the specification of the single-precision binary floating-point number format [14], a floating-point number consists of three fields, namely the *Sign* with the most significant bit (with index 31), the *Exponent* with the middle 8 bits (with index from 23 to 30) and the *Mantissa* with the last 23 bits (with index from 0 to 22). The actual value of a floating-point number is computed by $(-1)^{Sign} \times 2^{Exponent-127} \times 1.Mantissa$. The least significant bit is the last bit of Mantissa (i.e., the bit with the index 0). If the least significant bit is replaced, the introduced relative error is no more than 2^{-23}. The distortion of the mesh is therefore negligible even if the LSB of each vertex is changed arbitrarily. Using LSBs to store the digital signature, which is adopted by Wu and Cheung's scheme, is thus imperceptible. In this way, the file format does not need to be changed. This means whether the digital signature is inserted to a piece or not, the previous rendering engine which cannot verify the digital signature can still parse and render the mesh content correctly.

E. Secret Key Exchange

In secure group communication, a pair of the group members need to negotiate a secret key for encryption/decryption between them. Many key exchange protocols can be used for the purpose. In this subsection, we introduce a password-based key exchange protocol [15] proposed by Bellare, Pointcheval and Rogaway. The protocol can be used to exchange a secret key shared between users A and B with a pre-specified large prime p, a pre-specified *primitive root* (or *generator*) g of p, and a shared password pw known only by A and B. It's security is based on the complexity of solving the discrete logarithm problem and can be proved in a formal model even though the password is chosen from a small range. It has the following steps:

1) User A first picks a random value x, then calculates $g^x\ (mod\ p)$ and $E_{pw}[g^x\ (mod\ p)]$. Note that for the sake of representation convenience, we omit "$(mod\ p)$" in the rest of the paper.
   In this step, A sends his/her identity A and $E_{pw}[g^x]$ to B. The procedure can be abstracted as follows.
   $A \rightarrow B : A, E_{pw}[g^x]$
2) B picks a random number y and calculates g^y. B also uses the shared password pw to decrypted $E_{pw}[g^x]$ to obtain the value g^x, and calculates the secret key $sk' = H(A\|B\|g^x\|g^y\|g^{xy})$. Finally, B transmits $E_{pw}[g^y]$ and $H(sk'\|1)$ to A. After A receives this message, A also uses the share password pw to decrypt $E_{pw}[g^y]$ to obtain the value g^y, and calculates the secret key $sk'' = H(A\|B\|g^x\|g^y\|g^{yx})$. Note that the value g^{xy} is equivalent to g^{yx}, so $sk' = sk''$. A ensures that B already knows the secret key by checking $H(sk'\|1) \overset{?}{=} H(sk''\|1)$. The procedure can be abstracted as follows.
   $A \leftarrow B : E_{pw}[g^y], H(sk'\|1)$
3) User A transmits $H(sk'\|2)$ to B, and B ensures that A already knows the session key by checking $H(sk'\|2) \overset{?}{=} H(sk''\|2)$. The procedure can be abstracted as follows.
   $A \rightarrow B : H(sk'\|2)$

III. The Proposed Scheme

Observing that users in virtual environments usually form a group to perform specific tasks, such as visiting a shopping mall together or fighting monsters cooperatively in an MMOG, we propose to take advantage of secure group communication to further reduce the cost of 3D streaming authentication. The basic idea is for members of a group, formed by social network facilities or MMOG team formation tools, to elect a leader according to the reputation, computing power or connection stability of members. The leader will verify the authenticity and integrity for 3D contents of interest by a public authentication mechanism, such as the hash chain-based scheme proposed in [8], and will then relay the authenticity by encrypted checksum values to all members of this group via secure group communications. We also suggest embedding the authentication information into the LSB of floating point numbers representing 3D contents to make the authentication information imperceptible.

In this section, we show a scheme achieving group-based P2P 3D streaming authentication for the case of progressive meshes. The scheme has three phases: 1) secure channel initialization, 2) authenticity and integrity relay, and 3) verification. We describe the details of every phase in the following subsections.

A. Secure Channel Initialization Phase

In the first phase, group members elect a leader, and then each member establishes a *secure channel* with the leader by a secret key exchange mechanism similar to Bellare et al.'s [15]. Besides, a *padding value* (PV) is exchanged after the secure channel is established. The padding value PV is used to pad on 3D contents before contents checksum is calculated. This phase has the following steps for initializing the secure channel and exchanging PV. We assume all users have verified/trusted others' public keys before continuing the steps below. Note that the public key infrastructure (PKI) can meet this requirement.

1) Let $P = \{peer_1, peer_2, peer_3, ..., peer_n\}$ be a group of members (peers). Firstly, peers in P elect a leader, denoted by $leader$, where $leader \in P$.
2) Every peer picks an individual random number. Suppose $leader$ picks x_i and $peer_i$ picks y_i. Then $leader$ computes g^{x_i} and $peer_i$ computes g^{y_i}, where p is a sufficient large prime and g is p's generator.
3) $leader$ sends $\{g^{x_i}\}_{PK_i}$ to $peer_i$, where g^{x_i} is encrypted by $peer_i$'s public key. The procedure can be abstracted as follows.
   $leader \rightarrow peer_i : \{g^{x_i}\}_{PK_i}$, where $peer_i \in P \backslash \{leader\}$.
4) $peer_i$ sends g^{y_i} and $H(g^{x_i y_i}\|1)$ to $leader$, where g^{y_i} and $H(g^{x_i y_i}\|1)$ are encrypted by $leader$'s public key. The procedure can be described as follows.
   $leader \leftarrow peer_i : \{g^{y_i}\}_{PK_{leader}}, H(g^{x_i y_i}\|1)$
5) $leader$ decrypts the received message in the previous step to get g^{y_i} and $H(g^{x_i y_i}\|1)$. Then $leader$ computes $g^{y_i x_i}$ and the hash value of $(g^{y_i x_i}\|1)$ (note that the value $g^{x_i y_i}$ is equivalent to $g^{y_i x_i}$). $leader$ accepts the message if the hash value matches with $H(g^{y_i x_i}\|1)$; otherwise $leader$ rejects it. Likewise, $peer_i$ also does the same match checking.
6) Now, $leader$ and $peer_i$ share a pairwise secret key sk_i computed by $sk_i = H(g^{x_i y_i}\|2)$. $leader$ chooses a random padding value PV_i, encrypts PV_i and its checksum by sk_i and transmits the encrypted PV_i and checksum to $peer_i$. The procedure is abstracted as follows.
   $leader \rightarrow peer_i : E_{sk_i}[PV_i\|checksum\ of\ PV_i]$
7) $peer_i$ decrypts the received message to get PV_i and its checksum, and then checks the legitimation of PV_i. $peer_i$ accepts the secret key sk_i if PV_i is legitimate; otherwise, $peer_i$ rejects the secret key and $leader$ and $peer_i$ should go through all steps in the first phase. (However, we omit the star-over process in this paper to save space.)

B. Authenticity and Integrity Relay Phase

In this phase, the leader performs 3D streaming authentication and relays the authenticity and integrity of 3D contents. We assume the leader has the ability to quickly download the 3D contents of interest and verify the authenticity and integrity of the contents by a pre-specified mechanism, such as hash-chain signature scheme proposed in [8]. The leader computes checksum values for each piece of 3D content, and then sends checksum values to every group member. In order to prevent an insider from forging 3D contents and their corresponding fake checksum values to cheat normal group members, a padding value PV_i for $peer_i$ is added to each piece of 3D contents before calculating the checksum values. Moreover, a timestamp is also delivered along with checksum values to make the scheme more secure.

Many methods can be used for calculating the checksum value of a piece of data. Among them, the cyclic redundancy check (CRC) is the most well known method. However, CRC uses the division operation, so its computation cost is not as low as one may expect, especially when the chosen divisor is large. Actually, the computation cost of CRC is as high as the MD5 hash function when the divisor is longer than 32 bits according to a report in [16]. So, picking a fast checksum algorithm, such as Adler 32, is essential for our proposed scheme.

To sum up, the leader verifies the authenticity and integrity of the received 3D contents, computes for each $peer_i$ the checksum of each piece of contents by adding the padding value PV_i, and then sends each $peer_i$ the checksum encrypted by secret key sk_i. The procedure is abstracted as follows.

$leader \rightarrow peer_i : E_{sk_i}[c_i \| t_i \| ct_i]$

Note that c_j, t_j, ct_j in the above abstraction stands for the checksum of data piece D_j, the checksum of the time stamp of c_j, and the checksum of c_j and t_j according to Table I.

C. Verification Phase

In this phase, group members authenticate 3D streaming data with the help of the leader. By peer-to-peer 3D streaming techniques, such as FLoD [4], [5], a group member $peer_i$ can download 3D contents from any source in the network. After downloading a data piece, $peer_i$ can verify the authenticity and integrity by the checksum values sent by the leader. The steps for the verification taken by $peer_i$ are described below.

1) $peer_i$ receives the data piece D_j of 3D contents from any other peer, the leader or the server, and receives the authentication message form the leader.
2) $peer_i$ decrypts the authentication message sent by the leader to obtain $c_i \| t_i \| ct_i$.
3) $peer_i$ first checks if the authentication message is legitimate by comparing ct_i with the checksum of c_i and t_i.
4) $peer_i$ checks whether the timestamp is obsolete or not by the equation: $current\_time - t_i \overset{?}{<} TH$, where TH is a pre-specified threshold of time.
5) $peer_i$ computes the checksum value c_i' of the received data piece D_j and perform the following checking: $c_j' \overset{?}{=} c_j$.
   If the result of the checking is positive, then the data piece D_j is authenticated.

IV. Security Analysis

The security of the proposed group-based 3D streaming authentication scheme is based on 1) the trust of the leader and 2) the authenticity and integrity relay by encrypted checksum values sent by the leader. Below, we perform security analysis in these two aspects.

A. The Trust of the Leader

The group leader is designated to verify the authenticity and the integrity of 3D contents and to relay the verification to other group members. A totally trusted group leader is ideal. However, motivations for the leader to cheat indeed exist in spite that the leader was usually chosen with his/her higher reputation than others. Fortunately, the proposed scheme uses the leader's public key to securely exchange the secret key, so that a peer A can always identify another peer B when B plays the role of the leader and cheats. To detect cheating, we can designate more than one leaders to separately relay authenticity and integrity to group members. The cheating of an individual leader (or a minority of leaders) can be detected unless a majority of group leaders collude to cheat.

B. Authenticity and Integrity Relay

The group leader relays the authenticity and integrity of 3D contents to all other group members after 3D contents are successfully verified. The authenticity and integrity of 3D contents can be confirmed by group members because it is infeasible to modify 3D contents to match the encrypted checksum without the knowledge of the secret key used to encrypt the checksum. Moreover, the proposed scheme embeds timestamps in checksum authentication messages, so a replayed checksum message will be discarded if the embedded timestamp is obsolete. This can prevent adversaries from disseminating outdated 3D contents to group members. The relay process is therefore secure.

V. Evaluation

In this section, we evaluate the proposed scheme in terms of 1) the computation cost saved, and 2) the rendering quality when imperceptible 3D mesh authentication scheme is integrated with the proposed scheme. As we will show, about 2/3 computation is saved and the rendering quality is acceptable. Below we begin with evaluating the computation cost.

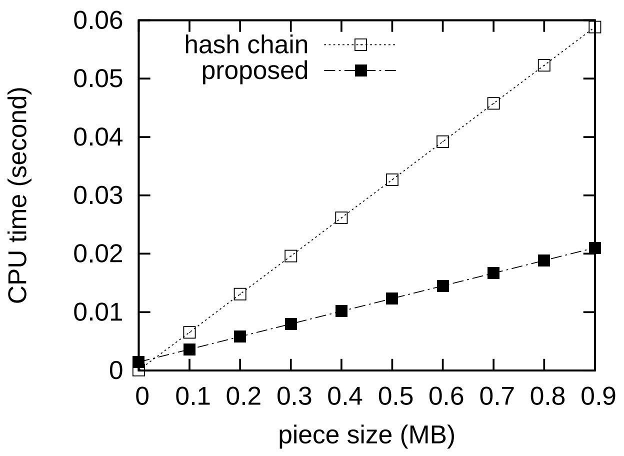

Figure 2. The CPU time consumption of the hash chain-based and the proposed authentication schemes

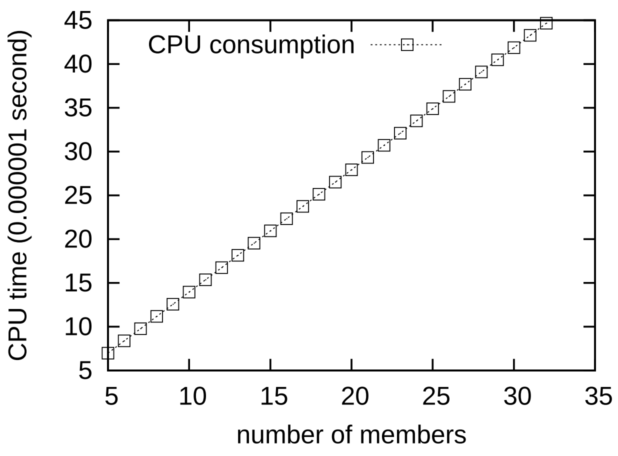

Figure 3. Relationship between the number of group members and the leader's CPU time consumption

A. Computation Cost

In the proposed scheme, the group leader needs to verify the authenticity and integrity of 3D contents with a pre-specified authentication mechanism, such as the hash chain-based mechanism proposed in [8], while group members only need to perform the verification with encrypted check-sum values sent by the leader. Consequently, the group members save a lot of computation, while the leader bears heavier computation load. We assume that 1) the pre-specified authentication mechanism is the hash chain-based mechanism proposed in [8], and 2) the mechanism uses SHA-1 hash function, and 3) the proposed scheme uses the Adler32 checksum function and the AES/CTR 128-bit secret key encryption/decryption function. In Figure 2, we plot, according to benchmarks reported in [16], the CPU consumption time of a group member for the hash chain-based authentication scheme and the proposed scheme under different data piece data sizes. We can easily observe that the proposed scheme consumes less computation time than its counterpart and the superiority is more significant when the data piece size is larger. When data piece size is about 0.9MB, about 2/3 computation is saved.

On the other hand, the leader has heavier computation load since it needs to calculate the checksum value of each data piece separately for each individual group member. Furthermore, the leader should encrypt the checksum values by a secret key separately for every group member before sending them to the member. The computation load of the leader thus increases with the number of group members. Fortunately, the checksum and the secret-key encryption do not cause too much computation and the number of group members is usually not too large. Note that when a group has too many members, we can demand the members to elect more leaders so that each leader will serve only a small number of members. Figure 3 plots the relationship between the group size and the leader's CPU consumption.

B. Rendering Quality for Imperceptible Authentication

As mentioned in Section III, authentication information is suggested to be embedded into the LSB of floating point numbers representing progressive meshes to make the information imperceptible. In this section, we evaluate the rendering quality for such information embedding when the hash-chained based authentication scheme proposed in [8] is adopted to authenticate progressive meshes. Figure 4(a) and Figure 4(b) show the rendering of the base mesh and the full mesh of a horse model represented by progressive meshes. Since a typical digital signature is about the size from 1024 to 2048 bits and the base mesh has 60 vertices in the horse model, each floating point number has to carry 12 bits of a 2048-bit digital signature. Figure 4(c) shows the base mesh of which every floating point number carries 12 bits of the digital signature. We can see that distortion exists between Figure 4(a) and Figure 4(c). However, the result is acceptable since the base mesh is just used to roughly show the simplified shape of the 3D model. On the other hand, since a SHA-1 hash value occupies 160 bits and every vertex split has about 15 vertices in the horse model, each floating point number has to carry 4 bits of the hash value. Figure 4(d) shows the rendering of the horse model with the complete mesh embedded with hash values. We can see that there is almost no difference between Figure 4(b) and Figure 4(d). This result indicates that the rendering quality of imperceptible hash chain-based 3D streaming authentication scheme is acceptable for the case of progressive meshes.

VI. Conclusion

We have presented an authentication scheme for group-based peer-to-peer 3D streaming using secure group communication to reduce CPU time consumption. The group

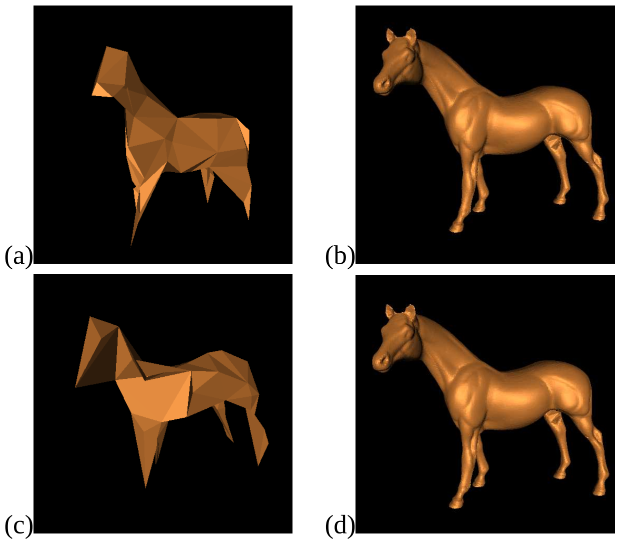

Figure 4. (a)Original base mesh (b)Original complete mesh (c)Base mesh in which 12 floating point number LSBs carry authentication information (d)Complete mesh in which 4 floating point number LSBs carry authentication information

leader is assumed to be trusted and is designated to verify 3D contents' authenticity and integrity with a public authentication mechanism, and to send group members a encrypted checksum value for every content piece via group secure channels. By the encrypted checksum values, a group member can authenticate 3D contents downloaded from any source. Since checksum is cheap to compute, much computation is saved. One problem of the proposed scheme is that it requires every group member should establish a secret key shared with the leader, which incurs a lot of computations and communications. Fortunately, users in a virtual environment of a group usually establish such keys when joining the group. So, there is no extra cost of establishing secret keys caused by the proposed scheme. Another problem of the proposed scheme is that the leader bears a heavier load than others. To remedy the problem, we can allow multiple leaders to be elected so that a leader serves only a limited amount of members.

We have evaluated the computation saving of the proposed scheme for the case of progressive meshes using the hash chain signature 3D streaming authentication scheme. As we have shown, about 2/3 computation is saved for the group member using the proposed scheme when data piece is about 0.9MB. We have also evaluated the rendering quality when embedding hash chain signatures into the floating point number least significant bits (LSBs) of the mesh data to make the signatures imperceptible. As we have shown, the rendering quality is acceptable.

Acknowledgment

The authors are grateful to Cheng Wei for helpful discussions and ideas for 3D mesh model programming. This research was supported in part by the National Science Council of the Republic of China (Taiwan) under the grant NSC95-2221-E-008-048-MY3 and the grant NSC97-2221-E-008-060.

References

[1] http://www.virtualenvironments.info/.

[2] http://secondlife.com/.

[3] H. Hoppe, "Progressive meshes," in *Proceedings of SIGGRAPH 1996*, New Orleans, LA, 1996, pp. 99–108.

[4] S.-Y. Hu, "A case for 3d streaming on peer-to-peer networks," in *Web3D '06: Proceedings of the eleventh international conference on 3D web technology*, 2006, pp. 57–63.

[5] S.-Y. Hu, T.-H. Huang, S.-C. Chang, W.-L. Sung, J.-R. Jiang, and B.-Y. Chen, "Flod: A framework for peer-to-peer 3d streaming," in *INFOCOM 2008*, April 2008, pp. 1373–1381.

[6] H.-T. Wu and Y.-M. Cheung, "Public authentication of 3d mesh models," in *Proceedings of the 2006 International Conference on Web Intelligence*, 2006, pp. 940–948.

[7] ——, "A fragile watermarking scheme for 3d meshes," in *MM&Sec '05: Proceedings of the 7th workshop on Multimedia and security*, 2005, pp. 117–124.

[8] M.-C. Chan, S.-Y. Hu, and J.-R. Jiang, "Secure peer-to-peer 3d streaming," *Multimedia Tools and Applications*, May 2009.

[9] A. Bharambe, J. Pang, and S. Seshan, "Colyseus: a distributed architecture for online multiplayer games," in *Proceedings of the 3rd conference on 3rd Symposium on Networked Systems Design & Implementation*, vol. 3, 2006, pp. 12–12.

[10] S.-Y. Hu, J.-F. Chen, and T.-H. Chen, "Von: a scalable peer-to-peer network for virtual environments," *Network, IEEE*, vol. 20, no. 4, pp. 22–31, July-Aug. 2006.

[11] A. Bharambe *et al.*, "Donnybrook: Enabling large-scale, high-speed, peer-to-peer games," in *Proc. SIGCOMM*, 2008.

[12] W.-L. Sung, S.-Y. Hu, and J.-R. Jiang, "Selection strategies for peer-to-peer 3d streaming," in *NOSSDAV '08*, 2008, pp. 15–20.

[13] F. Bergadano, D. Cavagnino, and B. Crispo, "Chained stream authentication," in *Proceedings of the 7th Annual International Workshop on Selected Areas in Cryptography*, 2000.

[14] W. Cheng, W. T. Ooi, S. Mondet, R. Grigoras, and G. Morin, "An analytical model for progressive mesh streaming," in *Proceeding of ACM Multimedia 2007*, 2007, pp. 737–746.

[15] A. S. 754-1985, "Standard for binary floating point arithmetic."

[16] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," *Eurocrypt 2000 (LNCS 1807)*, pp. 139–155, 2000.

[17] http://www.cryptopp.com/benchmarks.html.