Noname manuscript No.
(will be inserted by the editor)

Secure Peer-to-Peer 3D Streaming

Mo-Che Chan · **Shun-Yun Hu** · **Jehn-Ruey Jiang**

Received: date / Accepted: date

Abstract In recent years, interactive virtual environments such as Second Life, and virtual globe applications such as Google Earth, have become very popular. However, delivering massive amounts of interactive content to millions of potential users brings enormous challenges to content providers. Distributed peer-to-peer (P2P) approaches have thus been proposed to increase the system scalability in affordable ways. Building content delivery systems based on P2P approaches nevertheless creates security concerns for commercial vendors. This paper presents a generic system model for subscription-based service providers to adopt P2P-based, non-linear streaming for interactive content. We also propose solutions to the issue of content authentication, such that paying customers can be sure of the authenticity of the content retrieved from other users. Other practical security issues in an extended system model are also identified to allow further investigations in this problem space.

Keywords Peer-to-Peer · Virtual Environments · Nonlinear Media · 3D Streaming · Security · Online Games

1 Introduction

In recent years, a number of interactive multi-user virtual worlds, such as *World of Warcraft* and *Second Life*, have proliferated. Millions of people are now paying subscribers to such services, engaging in epic adventures or the creation and trading of virtual items worthy of millions of dollars. Interactive, alternative life-styles are possible in these networked *virtual*

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.

Mo-Che Chan
Department of Computer Science and Information Engineering National Central University, Taiwan, R.O.C.
E-mail: chrysler@acnlab.csie.ncu.edu.tw

Shun-Yun Hu
Department of Computer Science and Information Engineering National Central University, Taiwan, R.O.C.
E-mail: syhu@csie.ncu.edu.tw

Jehn-Ruey Jiang
Department of Computer Science and Information Engineering National Central University, Taiwan, R.O.C.
E-mail: jrjiang@csie.ncu.edu.tw

environments (VEs) [31], and new ones are introduced almost every month (e.g., *Barbie Girls*, Sony's *Home*, *Entropia*, *IMVU*, *Metaplace*, *VastPark*, and so on). In these VEs, users adopt a virtual self representation called the *avatar* to interact with a limited number of other users within his or her view (known as the *area of interest*, or AOI). Two recent trends in VEs have been *larger scale*, where both the world size and peak concurrent users are growing, and the emergence of *user-generated content* as a part of the user experience. Coincidentally, we have also seen a number of virtual globe applications (most notably *Google Earth*) that allow users to navigate a planet-scale environment, with detailed satellite imagery at the ground level.

Although tailored to different needs, both virtual world and virtual globe applications share two common traits: the adoption of 3D content and the content's growth to massive scale (e.g., Google Earth has over 70 terabyte of data, while Second Life has over 34 terabyte of content in 2007 [1]). When content becomes larger and more dynamic, *content streaming* will be an integral part for virtual worlds or globes, as already seen in Google Earth and Second Life. Streaming provides better user experience when users can immediately visualize and interact with the content. This effectively avoids the long wait for download or installation, which becomes prohibitive and unpractical when the content is massive.

The streaming of 3D content (i.e., *3D streaming* [12]) has been proposed and adopted for over a decade since *progressive meshes* [10] were introduced. Unlike the popular Internet audio or video streaming, 3D content is served in highly interactive manners, and is hence more *latency-sensitive* than video streams. Also, as the content access pattern often depends on real-time behaviors (i.e., movements within a virtual world, or navigation on a virtual globe), 3D streaming is also *non-linear* in nature. These characteristics create unique challenges for designing efficient streaming mechanisms.

As we look towards virtual worlds with millions of concurrent users in a single environment, the *scalability* of streaming becomes a challenge, while the *affordability* of streaming will impact its adoption. *Peer-to-peer* (P2P) 3D streaming [4, 14, 28] thus has recently been proposed, in hope to provide highly scalable, yet affordable streaming for interactive 3D content. Using P2P approaches nevertheless creates new issues to address. Among the top concerns for commercial adoption is security guarantee, as the content is now obtained from not just the authentic publisher, but also from other user machines (i.e., peers). In this paper, we will identify practical obstacles that must be overcome, in order for subscription-based services to adopt P2P, interactive 3D streaming. We then present solutions for authenticating different types of interactive 3D content streaming. Several schemes have been proposed for authenticating 3D mesh data, texture data and multimedia streaming data [18, 26, 39, 40]. To the best of our knowledge, no research focuses on the authentication of 3D streaming, though. This paper is one that concentrates on protocols for verifying the authenticity and integrity of 3D streaming.

The rest of the paper is organized as follows. Section 2 provides some background on P2P 3D streaming and the main system model for a commercial P2P 3D streaming system. Section 3 describes our proposed authenticated content streaming schemes, and Section 4 presents the security and performance analysis for the proposed schemes. In Section 5, we present an extended system model and future topics worthy of investigations. Finally, we conclude the paper in Section 6.

[1] http://www.informationweek.com/news/showArticle.jhtml?articleID=197800179

2 Background and Model

2.1 Background

There are roughly four types of 3D streaming in use today: object streaming, scene streaming, visualization streaming, and image-based streaming [12]. For virtual worlds and virtual globes, our main interests are in *scene streaming* [37] and in object streaming [10, 19]. We assume that some 3D objects are located at various places in the VE. A user navigates the scene and has a visibility area (i.e., AOI) that constantly covers new objects. Scene streaming consists of two main stages: 1) *object determination*, where some objects of interest are determined and prioritized according to a user's viewing angle or preference, and 2) *object transmission*, where the objects are downloaded using object streaming techniques such as progressive meshes [10]. Objects themselves are fragmented into a *base piece* and many *refinement pieces* to allow progressive download. A user renders a rough 3D view when the base pieces are obtained, and progressively improves the rendering when subsequent refinement pieces arrive.

Some recent works propose the use of P2P networks for content delivery to support 3D streaming on a large-scale. The main idea is that as users in the same VE often have overlapping visibility, certain content thus can be shared among users who have common views (i.e., interests). To support a large number of concurrent users, computations such as visibility determination or the prioritization of object requests, should also be handled by clients to ease the server loading [6].

Four main stages can be roughly identified for P2P 3D streaming: 1) *object discovery*, where a client learns of which objects are within its visibility; 2) *source discovery*, where a client learns about the potential content sources (including both other clients and the server); 3) *state exchange*, where the clients form interest groups to share information on content availabilities and network conditions; and 4) *content exchange*, where the actual content streaming occurs among the clients, such that local policies and preferences determine the proper selection of peers and pieces to request.

FLoD [14] 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) [2–4, 8, 11, 17, 29], where a 2D spatial overlay provides a list of nearby users within view (called the *AOI neighbors*), for the discovery of content sources. 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 content. The server is contacted only if no neighbors have the relevant content. As the query-response approach to inquire content availability may be slow, a follow-up work of FLoD [36] 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 content.

Royan et al. propose another design for P2P 3D streaming, where a *level of detail description tree* (LODDT) [28] organizes 3D buildings from a large city model into a hierarchical tree structure. A user can discover visible objects quickly using the tree structure and determine the priority of object requests. For source discovery, a P2P VE overlay is also assumed to provide AOI neighbors as potential sources. Clients exchange their network conditions privately and would request from each other based on estimates on both content availability and bandwidth loading. In the *HyperVerse* design [4], a collection of backbone servers keep the lists of objects and AOI neighbors, so clients are notified directly

by the servers for both object and source discoveries. Once the AOI neighbors are known, the clients also exchange availability states and request 3D objects among themselves to offload 3D content delivery from the servers.

2.2 System Model for P2P 3D Streaming

Before describing our schemes, we first present a system model for 3D virtual world or virtual globe applications that utilizes streaming content delivery. We note that while the above schemes on P2P 3D streaming (e.g., FLoD, LODDT, and HyperVerse) have described a general process, they have not described a complete system model that includes processes such as login, account management, state management, and content streaming, such that commercial vendors could adopt. Below we will present a basic outline for such a general model, upon which the security threats can be more clearly defined and solutions be more concisely described. Note that we will use *user* and *peer* interchangeably in our descriptions.

Our scenario is a commercial vendor who has some proprietary content to be delivered to paying customers on a monthly or hourly subscription, but would like to utilize the customers' computers for content delivery to improve scalability and to lower costs. We emphasize that although today's predominant model of VEs is for customers to download and install the VE application beforehand, such model will become inadequate if the content size becomes massive (e.g., terabytes) and dynamic (e.g., user-generated, or user-modifiable).

We assume that the world is a large 2D plane (for games) or a sphere (for globes), where users can navigate freely with manual controls. Various *content objects* are located around the plane, including *static objects* (e.g., trees and buildings), or *dynamic objects* (e.g., virtual people, movable tables or cars). There may also be *terrain* data that covers the whole ground. All these data are collectively called *content* (as opposed to object *states* such as a user's position, or an computer character's health points). We also assume the existence of the methods to fragment the different types of content into *pieces*. For example, 3D models may be represented as *progressive meshes* [10], and textures may be represented in progressive encodings. Even for content that appears to be continuous, such as terrain, we still assume that they can be divided into pieces (e.g., a terrain can be seen as a big texture dividable into square tiles). All content can be rendered once the relevant pieces are available (even if just the first, or *base piece*, is available to the user).

The data retrieval procedures can then be summarized in the following steps, by adopting the common components from both the FLoD [14] and HyperVerse [4] designs:

- Each peer contacts a *login server* to authenticate its join.
- The peer obtains the necessary game states and meta-information about the objects within the region from the server (i.e., object discovery). The server also notifies the peer of a list of AOI neighbors currently within the peer's view (i.e., source discovery).
- The peer contacts each one of its AOI neighbors, to exchange states regarding content availability and network conditions. This procedure continues periodically so that peers always have refreshed states about their neighbors.
- The peer then initiates *content exchange* with certain other peers to obtain the content of interest (probably those within its AOI).

For virtual globe applications, as often there would be several *layers* of content data, each representing a different level of detail (LOD) of the content viewed from a different altitude. We thus also assume that different layers may be partitioned into different sizes (the higher the altitude, the wider the region size). The actual game state management and the

Table 1 Classification and Properties of 3D Content Types

	Content Type (example)	Signature Scheme	Cost / Benefit
1.	whole model (regular meshes)	general digital signature	requires full download / lowest overhead
2.	linear stream (progressive meshes)	hash chain	more overhead / one-the-fly rendering
3.	independent stream (point cloud)	Rabin-based	highest overhead / fast forward supported
4.	partially linear stream (view-dependent meshes)	hash DAG	hybrid of 2. linear and 3. independent

content exchange methods between peers are all outside our scope and we assume scenarios as described by FLoD [14] and HyperVerse [4]. In this paper, we are mostly interested in the security aspects to support such a scenario.

3 Authentication of 3D Content Streaming

The basic problem in content authentication is that users obtain published content from possibly a large number of other users in P2P-based streaming, instead of the authoritative content publisher. How to ensure that the users still receive the proper content, without malicious content modifications or replacements, thus is of importance to both the legitimate users and the publisher.

To provide such security guarantees for users, the content should be checked for its authenticity and integrity whenever a user receives it. Digital signature and message authentication code (MAC) are often used to verify the authenticity and integrity of the retrieved digital content. In a typical scenario, the publisher first generates a MAC based on the published content by the publisher's private key. The user then takes the publisher's public key to verify the received content, to ensure that the content is unmodified from the publisher. Cryptographic hash functions also are often used along with digital signatures because of their computational efficiency and ability to prevent existential forge [24]. However, digital signatures are costly if applied continuously, and would defeat the real-time requirement of 3D streaming. Finding efficient content authentication methods for the interactive 3D content thus is the main problem we want to tackle.

This section first classifies the properties of different types of 3D content, and then presents the authentication protocols suitable to efficiently verify the authenticity and integrity of a given 3D stream. Table 1 shows the different types of 3D streaming content, followed by their descriptions.

3.1 Content Classifications

1. The *whole model* content type is most basic form that needs to be fully downloaded before using. For example, for a normal mesh model, the user needs to download the whole mesh, before rendering can take place. This is a typical model format for certain small mesh models, or large models that need to be transferred between the publisher or some content serving super-peers.
2. In a *linear stream* such as *progressive meshes* [10], users can download and render the model progressively. Content of this type usually consists of a linear stream. The main

Table 2 Notations

Δ	the digital signature signed by a private key
$S_{sk}(M)$	the signature of a message M signed by secret key sk
M_i	the i^{th} piece of a data content
$\mathbb{M}_i$	the hash value of the i^{th} piece of a data content
$H(M)$	the hash value of message M

benefit (compared to the whole model) is that a rough sketch can be rendered first to allow users to quickly have a preview, and decide whether to stay or go somewhere else. The restriction here is that each piece of the stream depends on the previous one. This linear format also exists for content whose streaming does not depend on view-position (e.g., terrain or texture data).

3. An *independent stream* contains pieces that do not depend on each other. *Point cloud* models [23] are examples of independent streams, where patches of points form this model. Points can be downloaded in any ordering to reconstruct models simply based on the user's viewing preference.
4. *Partially linear stream* means that the dependency among pieces may follow a complex structure [7]. This format can often be found for *view-dependent* models, where the streaming sequence consists of linearly-dependent streams that are themselves dependent on only certain previous pieces. In such a case, a particular patch of mesh data may have higher priority and need to be downloaded first.

Figure 1 displays four types of 3D content. The transmission overhead is higher if data dependency is lower (i.e., the overhead is highest for independent stream, followed by either linear or partially linear, and whole model with the lowest overhead). However, lower dependency allows more flexible transmission for the content.

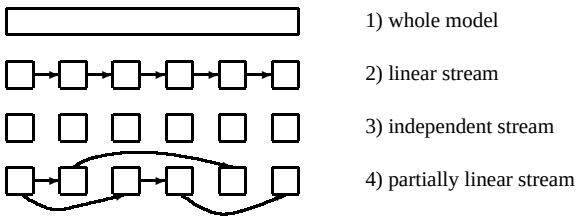

Fig. 1 Four basic dependency structure types of 3D content. 1) whole model 2) linear stream 3) independent stream 4) partially linear stream, e.g., directed acyclic graph (DAG)-like dependency

3.2 Proposed Authentication Protocols

We now present the authentication protocols that can verify MAC efficiently for the above stream types. Table 2 describes the notations used in the protocols.

3.2.1 Authenticating the Whole Model

A general digital signature protocol is adequate for authenticating simple content. The publisher first signs the hash value of the whole mesh model (or a texture) then publishes both the content and signature to users. Formula 1 describes the signature.

$$\Delta = S_{private\ key}\{H(whole\ content)\}. \tag{1}$$

After a peer receives the whole 3D content and its signature, it computes the hash value of the received content and uses the value and the publisher's public key to verify the signature. However, traditional digital signature protocol can not verify the authenticity and integrity of the received content if the content is only partially available. In other words, the data format can not support one-the-fly downloading and verifying.

3.2.2 Authenticating the Linear Stream

To support verification of the received content on-the-fly, a trivial solution is to generate digital signatures for each of the many pieces consisting a particular content. However, this trivial idea ignores the fact that public key cryptosystem requires a lot of computing power because of its many modular exponentiation computations. A stream signing mechanism [1, 9] thus can be adopted for better efficiency. As a 3D object, consisting of both mesh and texture data, can be treated logically as a *base piece* plus many *refinement pieces* [12], where each refinement piece depends on the previous piece. We can thus exploit such linear dependency in designing the proper authentication protocol.

When a provider publishes a 3D content, the model is first divided into $M_0, M_1, M_2, \ldots, M_n$, and texture divided into $T_0, T_1, T_2, \ldots, T_m$, where M_0 and T_0 are the base pieces and $M_1, M_2, M_3, \ldots, M_n$ and $T_1, T_2, T_3, \ldots, T_m$ are refinement pieces. The publisher then computes the hash values of those pieces (e.g., $\mathbb{M}_0$, $\mathbb{M}_1$), and sign them by using the formulas in Figure 2 (please see Table 2 for notations). The publisher then disseminates the digital signature of the hash of the base piece, Δ_M, some metadata of the object, *meta* (e.g., the object's ID, owner, size, number of pieces, etc.), and both the object pieces and their hash values, $\mathbb{M}_0$, M_0, $\mathbb{M}_1$, M_1, ... as a data stream. In order to verify the i-th message M_i immediately, a technique is to send the hash value $\mathbb{M}_i$ first before the message M_i. Because $\mathbb{M}_{i+1}$ is required when verifying M_i.

$$\begin{array}{ll}
\mathbb{M}_n = H(M_n) & \mathbb{T}_m = H(T_m) \\
\mathbb{M}_{n-1} = H(M_{n-1}|\mathbb{M}_n) & \mathbb{T}_{m-1} = H(T_{m-1}|\mathbb{T}_m) \\
\mathbb{M}_{n-2} = H(M_{n-2}|\mathbb{M}_{n-1}) & \mathbb{T}_{m-2} = H(T_{m-2}|\mathbb{T}_{m-1}) \\
\ldots = \ldots & \ldots = \ldots \\
\mathbb{M}_1 = H(M_1|\mathbb{M}_2) & \mathbb{T}_1 = H(T_1|\mathbb{T}_2) \\
\mathbb{M}_0 = H(meta|M_0|\mathbb{M}_1) & \mathbb{T}_0 = H(meta|T_0|\mathbb{T}_1) \\
\Delta_M = S_{sk}(\mathbb{M}_0) & \Delta_T = S_{sk}(\mathbb{T}_0)
\end{array}$$

Fig. 2 Production rules of authenticated mesh and texture data

The receiving peer can progressively verify the received content and render the mesh and texture. An example using progressive meshes is described as follows. The peer first

receives the stream Δ'_M, $meta'$, $\mathbb{M}'_0$, M'_0, $\mathbb{M}'_1$, M'_1, ... It can verify the authenticity of the main signature $\Delta'_M (= S_{sk}(\mathbb{M}'_0))$ by the publisher's public key. The base piece M'_0 is verified if $\mathbb{M}'_0 \stackrel{?}{=} H(meta'|M'_0|\mathbb{M}'_1)$, and the first refinement piece M'_1 is verified if $\mathbb{M}'_1 \stackrel{?}{=} H(M'_1|\mathbb{M}'_2)$. Likewise, the peer can verify subsequent pieces with just one hash operator. To further save bandwidth, the major signatures can be combined $\Delta = S_{sk}(\mathbb{M}_0|\mathbb{T}_0)$ if mesh and texture are transferred together. This protocol thus allows a peer to efficiently verify a linear content. However, neither the delivery nor verification for intermediate pieces can be skipped.

3.2.3 Authenticating the Independent Stream

For content that can be retrieved and used in any order, each piece has to be signed individually since the pieces are independent to each other. In such a scenario, the number of atomic digital signature operators cannot be reduced, so the fastest digital signature algorithm is needed. The Rabin public key cryptographic algorithm [25] can be applied to such a content type, as only one modular multiplication is needed to verify the authenticity and integrity of a Rabin signature. Such efficiency is achieved at a cost of a much slower signing process than other digital signature algorithms. However, we note that preprocessing of the content can often be employed by the publisher, so this additional cost should not negatively affect the system's run-time performance. Formula 1 is not efficient for verification now because each piece needs an additional hash operation. We describe the two main stages below:

Signing. When the publisher wants to publish a new content datum, each piece M_i has to be signed. A random number R_i is first picked and then the signature $S_i = \sqrt{(M_i|R_i|ObjectID)}\ (mod\ n)$ is computed for each piece M_i, where $n = p \times q$ is public, and p and q are two large primes which only the publisher knows privately. Formula 2 describes the signing of signature. Note that the random number R_i is used to make $(M_i|R_i|ObjectID)$ to be in QR_n (i.e., the *quadratic residue* under modular n [5]), and $ObjectID$ is an appointed string used to prevent existential forge attack, where the attack would not work if $ObjectID$ contains more than 80 bits. Some fixed padding is necessary if the length is shorter than this specific size. The publisher then sends the signatures S_i. Note that the piece M_i should be replaced by hash value of this piece if this piece is larger than a Rabin signature can hold. In other words, the publisher should sign $S_i = \sqrt{(H(M_i)|R_i|ObjectID)}\ (mod\ n)$ as the signature of this piece, and sends the signature along with M_i, R_i and $ObjectID$.

Extraction and Verification. To check whether the message is authentic, a peer can take the following steps when the publisher's signatures S_i are received. To extract M_i, R_i and $ObjectID$, the peer can compute $(M_i|R_i|ObjectID) = S_i^2\ (mod\ n)$, and then check whether $ObjectID$ is correct. The content can then be rendered after it is verified. Formula 3 describes the verifying of the signature.

With the above protocol, hash operators can be saved when the refinement pieces are small. When refinement pieces are large enough, it can still be hashed to smaller hash value to be signed efficiently. There is thus a tradeoff between message overhead and computation.

$$S_i = \sqrt{(M_i|R_i|ObjectID)}\ (mod\ n), \tag{2}$$

$$(M_i|R_i|ObjectID) = S_i^2\ (mod\ n). \tag{3}$$

3.2.4 Authenticating the Partially Linear Stream

The last type of 3D stream has irregular dependency. Here the dependency can be described as a *directed acyclic graph* (DAG) [7], where a piece may depend on several parent pieces, and may also impact the rendering of several child pieces. So we propose a *"hash DAG"* scheme to generate MAC for such streams. In Figure 3, the direction of arrow indicates "is parent of"(e.g., piece 2 depends on piece 1). The hash value of piece M_i can be generated if all the hash values of the predecessors of piece M_i are generated. The MAC generation procedure for this example is described as follows. The hash value of piece 3, $\mathbb{M}_3 = H(M_3)$, is generated first, then $\mathbb{M}_2 = H(\mathbb{M}_3|M_2)$ and $\mathbb{M}_5 = H(\mathbb{M}_3|M_5)$ can be generated. $\mathbb{M}_0 = H(meta|\mathbb{M}_1|\mathbb{M}_4)$ can be generated after $\mathbb{M}_1 = H(\mathbb{M}_2|\mathbb{M}_5|M_1)$ and $\mathbb{M}_4 = H(\mathbb{M}_2|M_4)$ are generated. Finally, the publisher signs $\mathbb{M}_0$ to generate the major signature $\Delta_M = S_{sk}(\mathbb{M}_0)$. The publisher can publish this dependency relation graph, then the receiver can know which pieces are needed. Receiver can first verify the signature $\Delta_M = S_{sk}(\mathbb{M}_0)$, then verify the pieces that follow. For example, anyone can verify piece 1 $\mathbb{M}_1 = H(\mathbb{M}_2|\mathbb{M}_5|M_1)$ if piece 1's M_1 and the hash values $\mathbb{M}_2, \mathbb{M}_5$ are received. All MAC can be generated, transmitted, and verified easily according to the dependency graph.

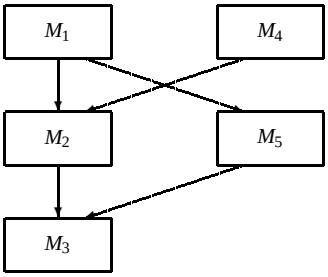

Fig. 3 Example dependency relation of partially linear stream

4 Evaluation

4.1 Security Analysis

We analyze the security for each of the four content types as follows.

Whole model. For the complete mesh or texture model, the traditional digital signature protocol is applied. Both the authenticity and integrity stand because it is difficult to find collisions of cryptographic hash functions or deliver a valid digital signature verifiable by the publisher's public key from a specific hash value. Any attacker faces two computationally infeasible problems without the publisher's private key, so this digital signature protocol cannot be forged easily.

Linear stream. The principle is the same here as in the previous traditional digital signature – it is difficult to generate a valid signature Δ_M that can be verified by the publisher's public key for a specific hash value. In addition, to find out M_{i+1}, which differs from the pre-image

of the original content M_i, is also infeasible. Unless the collision avoidance property of the adopted cryptographic hash function fails, the pieces from M_0 to M_n cannot be forged.

Independent stream. Independent signatures are used for each piece of the streaming content M_i, so it is necessary to achieve unforgeability for each piece M_i. According to the strength of the Rabin cryptosystem [25], it is infeasible to obtain the square root under module n for a specific value without knowing p and q, where $n = p \times q$, and p and q are two large primes. The essential security is described in Lemma 1. To generate the legitimate signature S_i is ideally infeasible without the correct private key p and q. However, the message is not exactly a specific value before computing the square root. As M_i is a variable, more advantages are given to the adversary. The remaining security is based on inability for the adversary to generate a valid signature such that the least bits of the signature match the fixed message $ObjectID$. For today's computers, about 2^{100} enumerate operators is computationally infeasible [30]. In other words, length of padding $ObjectID$ must be at least 100 bits. Therefore, the generated MAC for each piece is unforgeable.

Lemma 1 *To find out the signature of a specified message is infeasible if factoring n is intractable.*

Proof Reduction can be used to provide the proof of the intractability of the Rabin signature scheme. Let problem **A** be to factor n to p and q and problem **B** be to find out S from M, where $S = \sqrt{M} \ (mod\ n)$. The goal is to show that problem **B** has at least the same hardness as problem **A**. We know that if we can find two distinct square roots of a message M, we can factor the modulus n. Suppose an attacker is attempting to solve problem **A** and an oracle can response correct answers for problem **B**. The attacker first chooses a random value s and lets $m = s^2$. Now s is a valid signature of m. The attacker then submits m to the oracle. There is a one in two chance that it will produce the same signature s. If so, repeat this process. If not, the attacker has both square roots of m and can recover the factors of n. To be more precise, an attacker randomly computes $C = m_1^2 \ (mod\ n)$ and then sends C to the oracle. If the oracle responds $m_2 = SQRT_n(C)$ to the attacker, then the attacker successfully computes $gcd(m_1 - m_2, n)$ to give p or q. That is, the attacker can figure out p and q with 50% probability for each oracle querying round. To solve the GCD problem is computational feasible, so the attacker can factor n to p and q. Therefore, problem **B** is intractable if problem **A** is intractable.

Partially linear stream. The security principle is the same as the linear stream scheme.

4.2 Performance Analysis

Computation Overhead. Different 3D streaming delivery schemes, such as FLoD [14], have been presented for P2P VEs. The performance of the P2P delivery scheme has also been compared with client-server architectures via simulations [14]. We can see that P2P-based delivery mechanisms bring significant advantages in terms of scalability. On the other hand, benchmark results between different public key and cryptographic hash schemes have shown that public key cryptographic schemes are significantly slower than hashes [2]. As far as we know, Rabin is faster than other public key cryptographic schemes in terms of its speed to verify the signature. Although signing Rabin signatures is slow, the signing procedure can be done off-line in advance.

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

Transmission Overhead. The communication overhead of secure content streaming consists of the sizes of the hash values and signature. Formula 4 describes the ratio between communication overhead to content size. The overhead is relatively small if the original stream is not divided into pieces that are too small.

$$\frac{number\ of\ pieces * hash\ value\ size + signature\ size}{original\ stream\ size} * 100\% \quad (4)$$

5 Extended System Model and Future Topics

While our scenario provides a basic example on content authentication for P2P 3D streaming, the scalability of the system may still be limited by the centralized aspects for state management (i.e., object and source discovery), which may be a bottleneck if the system grows in the range of millions of concurrent users. To improve scalability, the whole VE may be partitioned into many disjoint *regions* to distribute the loads of content and state management [17, 27], using various partitioning methods (e.g., by squares, hexagons, or Voronoi diagrams [13]). A selected *super-peer* – a more trustworthy and capable machine – is responsible to manage the game states and content meta-data within each region. Super-peers are in general trustworthy, and may be selected based on their hardware capacities or the owners' reputations [15,20]. They also may be in constant contact with other super-peers managing neighboring regions. Two more steps are thus added to our system model:

- After authentication, the joining peer is directed to one of the super-peers that currently manages the region the user is interested to explore.
- Peers may move across different regions, at which point they would switch the super-peer to contact.

In order to provide a convenient experience for users, existing single sign-on mechanisms can be used as follows. The vendor's authentication server first validates a logging user based on 1) user's private knowledge (e.g., password), 2) possession of a token (e.g., smart card), or 3) user's biometric marks (e.g., finger prints) [21], with different trade-off involved. If the authentication passes, the server will issue the user a short-term *ticket*, which the user can then use this ticket to navigate within the system. Each collaborating peer in the system can validate the user by this ticket, so the user is authenticated conveniently just once and then can navigate everywhere in the VE. However, it is preferred that once a user has logged in and starts to navigate (contacting various super-peers and peers for content exchange), the user need not contact the authentication server again until logging off (i.e., a *single sign-on* is mandated).

In such a scenario, the scalability may be improved, but distributed authentication or state management are then needed. We now identify the following additional issues, if a commercial P2P VE system also utilizes super-peer resources for state management.

User Authentication. User authentication allows only paid subscribers to login to use the service. However, when the service is provided by not just the server, authentications among peers becomes necessary to ensure that only subscribers can receive and exchange content. As authenticating or querying continuously with the server is cumbersome, single sign-on may be more preferred. Proper accounting also requires that each user has only one login. Although authentication, authorization and accounting requirements are straightforward to

implement in a client-server architecture, they become more sophisticated in a P2P environment as the server may not know the current statuses of all online users. *Double playing* thus is an issue when distributed user authentication mechanism is adopted (e.g., a user can login more than once during the same period, and creates unfairness to other users).

Content Update. Besides the original server, published content may be placed at arbitrary peers after some content exchange. However, a service provider may update the content to reflect a change in the virtual environment. Applications may also allow users to modify or create new content as they see fit (e.g., Second Life's content is entirely user-generated). Different versions of the same content object thus may scatter around on the P2P network. Ensuring that content updates would reach relevant users timely and securely therefore is another problem.

Virtual Goods Duplication. User can own valuable virtual goods in the virtual world that may be traded for money. Currently, the states of these virtual goods are stored in a central database in the client-server architecture, and all users need to login to the server to perform transactions. However, if the virtual goods and credits are also stored on the P2P network (or for example, the models and textures are stored at peers due to exchange purposes). The peer that keeps the goods may duplicate and resell the virtual items. How to prevent credit or virtual goods stealing thus would be an important issue for supporting virtual goods transactions.

Super-peer Reliability. Super-peers play an important role as they now manage the users' status and maintain the P2P VE overlay. Although super-peers may be selected to be more trustworthy than regular peers, possibilities still exist for them to take advantages on normal peers. Besides cheating, super-peers may also fail and lose its currently stored states. If we can ensure the reliability and trustworthiness of super-peers, then P2P VEs could become more distributed.

6 Conclusion

3D streaming will provide a better user experience for the growing number of virtual world and virtual globe applications. As it is difficult to support a massive number of users with traditional client-server architectures, peer-to-peer networks are a promising solution to share the central server's loading. However, how to ensure that 3D streaming is secure then becomes an important problem for commercial adoption. In this paper, we discuss P2P-based 3D streaming from the aspect of the authenticated content streaming, where we present the classification of the four content types for 3D streaming, and their respective authentication protocols. We also analyze the security and performance of these protocols.

As we look towards even larger-scale systems with distributed state management based on super-peers, there are other new topics worthy of exploration. For example, the detection of double-playing of users, performing proper accounting to charge users, and content update mechanisms that ensure the users are always notified of the latest content securely. How to achieve the above in a P2P environment securely presents interesting issues that we will investigate in the future.

References

1. Bergadano F, Cavagnino D, Crispo B (2000) Chained stream authentication. In: Proceedings of the 7th Annual International Workshop on Selected Areas in Cryptography, Springer-Verlag, Lecture Notes In Computer Science, vol 2012, pp 144–157
2. Bharambe A, Pang J, Seshan S (2006) Colyseus: a distributed architecture for online multiplayer games. In: Proceedings of the 3rd conference on 3rd Symposium on Networked Systems Design & Implementation, San Jose, CA, vol 3, pp 12–12
3. Bharambe A, et al (2008) Donnybrook: Enabling large-scale, high-speed, peer-to-peer games. In: Proceedings of SIGCOMM
4. Botev J, et al (2008) The hyperverse - concepts for a federated and torrent-based "3d web". In: Proceedings of MMVE
5. Burton DM (2005) Elementary Number Theory, 6th Edition. ACM Press
6. Cheng W, Ooi WT (2008) Receiver-driven view-dependent streaming of progressive mesh. In: Proceedings of NOSSDAV
7. Cheng W, Ooi WT, Mondet S, Grigoras R, Morin G (2007) An analytical model for progressive mesh streaming. In: Proceedings of the 15th international conference on Multimedia, pp 737–746
8. Frey D, et al (2008) Solipsis: A decentralized architecture for virtual environments. In: Proceedings of MMVE
9. Gennaro R, Rohatgi P (1997) How to sign digital streams. In: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology, Lecture Notes In Computer Science; Vol. 1294, pp 180 – 197
10. Hoppe H (1996) Progressive meshes. In: Proceedings of SIGGRAPH
11. Hu S, Chen J, Chen T (2006) VON: a scalable peer-to-peer network for virtual environments. IEEE Network 20(4):22–31
12. Hu SY (2006) A case for 3d streaming on peer-to-peer networks. In: Proceedings of the eleventh international conference on 3D web technology, pp 57–63
13. Hu SY, Chang SC, Jiang JR (2008) Voronoi state management for peer-to-peer massively multiplayer online games. In: Proceedings of NIME
14. Hu SY, et al (2008) Flod: A framework for peer-to-peer 3D streaming. In: Proceedings of IEEE INFOCOM
15. Huang GY, Hu SY, Jiang JR (2008) Scalable reputation management for p2p mmogs. In: Proceedings of MMVE
16. Josephson WK, Sirer EG, Schneider FB (2004) Peer-to-peer authentication with a distributed single sign-on service. In: Proceedings of the International Workshop on Peer-to-Peer Systems
17. Knutsson B, Lu H, Xu W, Hopkins B (2004) Peer-to-peer support for massively multiplayer games. In: Proceedings of IEEE INFOCOM
18. Li Z.-T.,Wang W.-D.,Zhang Y.-J.,Li W.-M. Source Authentication of Media Streaming Based on Chains of Iso-hash Clusters. In: Proceedings of the Third International Conference on Autonomic and Trusted Computing, pp, 398-407
19. Lin N.-S., Huang T.-H., Chen B.-Y. (2007) 3d model streaming based on jpeg 2000. IEEE Trans. Consumer Electronics, 53(1).
20. Lo V, Zhou D, Liu Y, GauthierDickey C, , Li J (2005) Scalable supernode selection in peer-to-peer overlay networks. In: Proceedings of HOT-P2P
21. O'Gorman L (2003) Comparing passwords, tokens, and biometrics for user authentication. In: Proceedings of the IEEE, vol 91, pp 2012–2040
22. Pathak V, Iftode L (2006) Byzantine fault tolerant public key authentication in peer-to-peer systems. Computer Networks: The International Journal of Computer and Telecommunications Networking 50(4):579–596
23. Pauly M, Gross M, Kobbelt LP (2002) Efficient simplification of point-sampled surfaces. In: Proceedings of IEEE Visualization, pp 163–170
24. Pointcheval D, Stern J (1996) Security proofs for signature schemes. Advances in Cryptology — EUROCRYPT '96 1070/1996:387–398
25. Rabin MO (1979) Digitalized signatures and public-key functions as intractable as factorization. MIT/LCS/TR-212, MIT Laboratory for Computer Science
26. Rey C, Dugelay J (2002) A survey of watermarking algorithms for image authentication. EURASIP Journal on Applied Signal Processing, 2002(6):613–621
27. Rosedale P, Ondrejka C (2003) Enabling player-created online worlds with grid computing and streaming. Gamasutra Resource Guide
28. Royan J, Gioia P, Cavagna R, Bouville C (2007) Network-based visualization of 3D landscapes and city models. IEEE CG&A 27(6):70–79

29. Schiele G, et al (2008) Consistency management for peer-to-peer-based massively multiuser virtual environments. In: Proc. IEEE Virtual Reality (IEEE VR) workshop Massively Multiuser Virtual Environment (MMVE)
30. Schneier B (1996) Applied Cryptography, 2nd edn, John Wiley & Sons, chap 7
31. Singhal S, Zyda M (1999) Networked Virtual Environments: Design and Implementation. ACM Press
32. Smit G, Havinga P, Helme A (1996) Survey of electronic payment methods and systems. In: Proceedings of Euromedia
33. Soriano E, Ballesteros FJ, Guardiola G (2007) Shad: A human-centered security architecture for the plan b operating system. In: Fifth Annual IEEE International Conference on Pervasive Computing and Communications, pp 272–282
34. Steiner JG, Neuman BC, Schiller JI (1988) An authentication service for open network system. Proceedings of the Winter 1988 Usenix Conference, pp 191–202
35. Stoica I, Morris R, Karger D, Kaashoek F, Balakrishnan H (2001) Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of SIGCOMM, pp 149–160
36. Sung WL, Hu SY, Jiang JR (2008) Selection strategies for peer-to-peer 3d streaming. In: Proceedings of NOSSDAV
37. Teler E, Lischinski D (2001) Streaming of complex 3d scenes for remote walkthroughs. CGF (EG 2001) 20(3)
38. The MIT Kerberos Team (1980) The network authentication protocol. `http://web.mit.edu/Kerberos/`
39. Wong P.-W. (1998) A public key watermark for image verification and authentication, In: Proceedings of International Conference on Image Processing (ICIP 98), pp. 455-459
40. Wu H, Cheung Y (2006) Public authentication of 3d mesh models. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, pp 940-948