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

Secure Peer-to-Peer 3D Streaming

Mo-Che Chan · **Shun-Yun Hu** · **Chien-Hao Chien** ·
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 pose challenges for the processing and network capacities of the content providers. Distributed peer-to-peer (P2P) approach has thus been proposed to provide more affordable scalability. However, building content delivery systems based on P2P approaches create security concerns for commercial adoptions. This paper identifies three practical obstacles that must be addressed, in order for subscription-based service providers to adopt P2P-based, nonlinear interactive streaming: 1) the detection of double playing, ~~where~~ a subscriber of the service can only login with one presence anywhere in the system; 2) the authentication of content, ~~where~~ paying customers can be sure that the content they retrieve from other users are authentic; and 3) the update of content, ~~where~~ new versions of the content can be distributed to users timely and securely. We then present the basic solutions to each of these challenges, and also present the respective security analysis to our approach.

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

This research was supported in part by the National Science Council of the Republic of China under the grant NSC95-2221-E-008-048-MY3.

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

Chien-Hao Chien
Department of Computer Science and Information Engineering National Central University, Taiwan, R.O.C.
E-mail: chienhao1@gmail.com

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

1 Introduction

In recent years, there has been a proliferation of interactive, multi-user virtual worlds such as *World of Warcraft* and *Second Life*. Many millions of people have become paying subscribers to such services, or are engaged in the creation and trading of virtual items worthy of millions of dollars every month. Highly interactive, alternative life-styles are possible in these networked *virtual environments* (VEs) [28], and we are only seeing more adoptions with new ones being introduced almost every month (e.g., Barbie Girls, Google's Lively, Entropia, IMVU, Metaplace, to name a few). Users enter a particular VE via a virtual self representation called *avatar*, and interact with a limited number of total online users within the user's view, or *area of interest* (AOI). Two emerging trends for VEs have been *large-scale*, with larger space and more people, and *dynamic*, with user-generated content becoming an integral part of the experience. Coincidentally, we have also seen in recent years a number of virtual globe applications, most notably *Google Earth*, which allows users to easily navigate an entire planet-scale environment, with detailed imagery at the ground level.

Although tailored to different usage scenarios, both virtual world and virtual globe applications share one thing in common: the adoption of 3D content, that includes at the most basic level, 3D models and textures, and the content's potential growth to massive scale (e.g., Google Earth has over 70 TB of data, while Second Life has over 34 TB of content [1]). As content grows and becomes more dynamic, *content streaming* will become an integral part for virtual worlds or virtual globes, as already evidently seen in Google Earth and Second Life. Streaming is a promising way to provide better user experience as users can immediately visualize and interact with the content, without having to wait for a full download, which becomes prohibitive and unpractical when the size is massive.

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

However, 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~~ the scale of~~ its adoption. *Peer-to-peer* (P2P) 3D streaming [14, 25] thus has recently been proposed, in hope to provide highly scalable, yet affordable streaming for interactive 3D content. P2P 3D streaming nevertheless creates new issues to address. Among the top concerns for commercial adoption is the security guarantees of the streaming, 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 identify three practical obstacles that must be addressed, in order for subscription-based service providers to adopt P2P-based, nonlinear 3D streaming:

User Authentication. In order to permit only paid subscribers to use the service, user authentication is necessary upon login. However, in a P2P environment, authentications among peers are also necessary to ensure that only service subscribers can exchange content. As authenticating or querying continuously with the server is cumbersome, a *single sign-on* mechanism may be more~~ preferred~~ [31]. A user will be authenticated only once with the

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

server, then can navigate freely ~~among~~ various peers. Proper accounting requires that each user to have only one login. Although this requirement is easy to achieve in a client-server architecture, it becomes an issue in P2P environments, where the server may not necessarily know the current status of all online users after authentication. A mechanism to detect *double playing* thus is needed.

Content Authentication. The content published by a vendor should be be checked for its authenticity and integrity whenever received by users. Digital signature and message authentication code (MAC) are the basic tools usually applied to achieve such. A digital signature scheme is usually applied to verify the authenticity and integrity of an obtained digital content. There are at least two primary algorithms, one for signing the user's private key to deliver the digital signature, and the other for verifying signatures with ~~previous~~ user's public key. Under the difficulty of some mathematical problems, the security of digital signatures can be held. For example, the signature cannot be forged without the private key. 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, nonlinear 3D content thus is a relevant topic.

Content Update. ~~Other than~~ the original server, published content may locate at arbitrary peers after some content exchange. However, the service provider could update its content to reflect a change in the environment or theme. Different versions of the content therefore may scatter around the P2P networks. Ensuring that content updates would reach relevant users in a timely and securely fashion thus is another important problem.

In this paper, we investigate three topics: 1) the detection of double playing, where a subscriber of the service can only login with one presence anywhere in the system; 2) the authentication of content, ~~where~~ paying customers can be sure that the content retrieved from other users are authentic; and 3) the update of content, ~~where~~ new versions of the content can be distributed to users timely and securely. The rest of the paper is organized as follows. Section 2 provides some background on P2P 3D streaming and the main scenarios of our discussions. Section 3 describes our proposed schemes, and Section 4 presents the security analysis and simulations for the proposed schemes. Section 5 describes related work on topics in secure P2P streaming. Finally, we conclude the paper in section 6.

2 Background and Scenarios

2.1 Background

There are roughly four types of 3D streaming: object streaming, scene streaming, visualization streaming, and image-based streaming [12]. For virtual worlds and virtual globes, we are mainly interested in *scene streaming* [34], where some 3D objects are located at various places in the VE. A user navigates the scene and has a visibility sphere that constantly covers new objects. Scene streaming generally consists of two stages: *object determination*, where a list of objects are first obtained and prioritized according to viewing angle or preference; and *object transmission*, where the objects are downloaded using object streaming techniques such as progressive meshes. To allow progressive download, objects are often fragmented into various pieces, consisting of a *base piece* and many *refinement pieces*. A user renders a rough 3D view once the base piece is downloaded. Progressively better graphics can then be rendered as the subsequent refinements arrive.

To support 3D streaming on a large-scale, some recent work propose the use of P2P networks for content delivery. The main idea is that as users in the same VE often have overlapping visibility, certain content thus may be shared among users who have common interests. The brief system architecture is sketched as Figure 1. To minimize server resource usage, it is also preferred if computations such as visibility determination, or the prioritization of object requests can be delegated to clients [6]. FLoD [14] is one such framework that classifies the stages of P2P 3D streaming into the partitioning of the scene, the fragmentation of objects, the pre-fetching of content, the prioritization of requests, and the selection of peers for content exchange. FLoD relies on the recent research of P2P virtual environments (P2P VE) [2–4,8,11,17,26], where spatial overlays may provide a list of nearby users within view (called *AOI neighbors*). So once a navigating user obtains a list of AOI neighbors from the overlay, the user can then send queries and requests to neighbors to exchange content. The server is contacted only if no neighbors has the relevant content. A follow-up work of FLoD investigates other strategies than query-response to maintain the content availability information among peers [33]. Peers would actively push content availability to their AOI neighbors so to save the round-trip query time, additional AOI neighbors are also maintained to increase the potential pool of source peers who may provide content. Cavagna et al. propose a *level of detail description tree* (LOD-DT) [25] that organizes all the 3D objects into a hierarchical, spatial tree structure. A user can use the structure to quickly determine which objects are visible and can be request from nearby peers. A P2P VE overlay is also assume to provide AOI neighbors for content exchange. In the *HyperVerse* project [4], the list of AOI neighbors are kept by a collection of backbone servers. But once neighbors are learned from the servers, the peers also request and exchange 3D objects among themselves to offload the delivery of 3D content for the server.

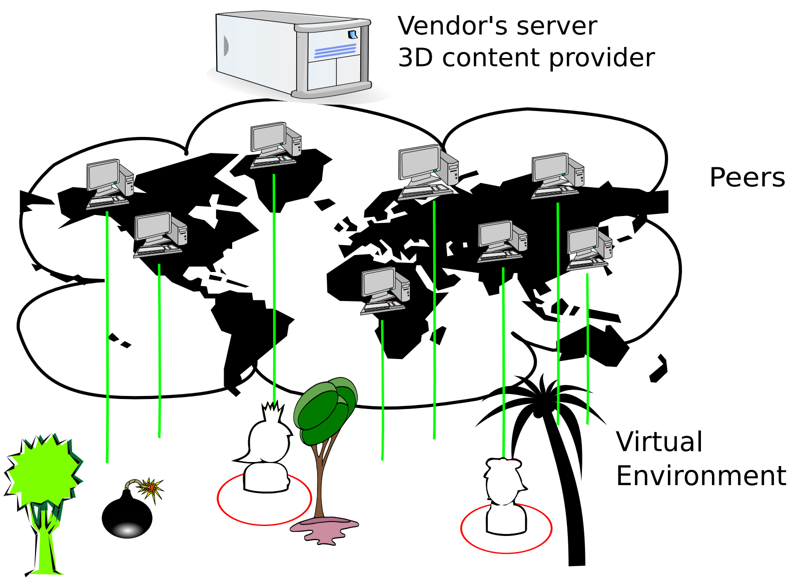

Fig. 1 The brief system architecture of presented scenario

2.2 Scenario Description

Before describing our schemes, we first present the scenario of the 3D virtual world or virtual globe that utilizes streaming for content delivery. Note that we may use *user* and *peer* interchangeably. Our scenario is a commercial vendor who has some proprietary content to be delivered to paying customers (e.g., via subscriptions), but would like to utilize the customers' hosts to improve scalability and lower costs. We assume that the world is a large 2D plane (for games) or a sphere (for globes), where a user can navigate freely with manual controls. There are various *content objects* located around the plane. For example, *static objects* such as trees and buildings, *dynamic objects* such as virtual people or movable ones such as tables or carts. There may also be *terrain* data that cover the whole ground. All these data are collectively called *content* (as opposed to object *states* such as a user's position, or an enemy character's health points), and we assume that there are existing methods to divide the different types of content into *pieces*. For example, 3D meshes may be represented as *progressive meshes* [10], and textures may be represented in progressive encodings. Even for content that looks continuously, such as terrain, we still assume that they can be divided into pieces (e.g., terrain as a big texture dividable into square tiles). All content can be rendered once the relevant pieces are available.

To manage and distribute content, we assume that the whole world can be partitioned into many *regions* [17,24]. While the specific partitioning is beyond our scope (e.g., different shapes such as squares, hexagons, or Voronoi diagrams can be used [13]), we assume that a selected *super-peer* – a more trustworthy and capable user node – is responsible to manage the game states and content meta-data within each region. The super-peers may also be in contact with other super-peers managing neighboring regions. We assume that super-peers are in general trustworthy, and existing methods for their selections [15, 18] based on capacities or owner reputations are available.

In general, the data retrieval procedures works similar to what has been described by FLoD [14] or HyperVerse [4], and can be summarized in the following steps:

– Each peer contacts a *login server* to authenticate its join.
– After authentication, the joining peer is directed to one of the super-peers that currently manages the region the user is interested to explore.
– The peer obtains the necessary game states (a topic beyond our scope) and also meta-information about the objects within the region from the super-peer.
– The peer then initiates an *exchange procedure* with certain other peers to obtain the content of interest (probably those within its AOI).
– Peers may move across different regions, at which points they would switch the super-peer to contact.

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 at a different height. We thus also assume that different layers may have partitioning of different sizes (the higher, the wider the region size), but each region is still managed by an assigned super-peer. The actual partitioning method for the world, the selection for super-peers, and the 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 environments.

In order to provide a convenient experience for users, existing single sign-on mechanisms are used in the following way. The vendor provides an authentication server to validate a logging user based on 1) user's private knowledge (i.e., password), 2) possession of a token (i.e., smart card), or 3) user's biometric marks (i.e., finger prints) [19]. Each of them has different advantages and disadvantages. If the authentication passes, the server will issue the user a short-term *ticket*, which the user can then use to navigate within the system. Each collaborating peer in the system can validate the user by this ticket, so the user is authenticated just once and then can navigate everywhere in the VE.

3 The Proposed Scheme

Our proposed framework mainly deals with the issues of double playing detection, authenticated content streaming, and secure content update. Below we describe the proposed scheme individually.

3.1 Double Playing Detection

Our vendor would like to allow paying subscribers to login and use its service. However, we prefer that once a user has login, the user need not contact the authentication server again until logging off, while navigating and contacting various super-peers and peers for content download (i.e., a *single sign-on* is ~~used~~). The vendor thus is concerned with whether a user logins ~~to~~ an account exactly once, without double playing (i.e., an account has multiple concurrent presences). A double playing detection mechanism not only maintains vendor's profits, but also prevents cheating ~~problems~~.

~~Double playing is easy to detect in client-server architecture, as the server maintains the list of online users, so it is trivial to check if a particular user has login.~~ However, as we adopt a single sign-on mechanism to improve scalability, each authenticated user is issued a ticket that is presentable to navigate within the VE. A legitimate user thus may copy the ticket to friends so that they could also use the service at the same time. Even if the authentication server keeps a list of login users, as it is not in constant contact with the users, it will not be able to tell whether a login user is still online. Our problem thus is to avoid double playing while still keeping the benefits of a single sign-on mechanism. The nature of this problem is that subscription records are kept at a central location, but online status may be kept distributively. The basic idea of the solution thus is to either centralize online status records in some scalable way, or to allow distributive queries of the subscription records. There are two policies to detect some double playing behaviors — DHT-based and Trusted super peer-based.

DHT-based. To detect whether a user is currently online during authentication, we create a user status record based on *distributed hash table* (DHT) [32]. DHT allows a pair of (key, value) to be inserted and queried from a network of n nodes in $O(log\ n)$ time. We build a DHT composed of all super-peers so that they can collectively store or query certain important information. Only the super-peers have the rights to access and modify information on this DHT, which mainly records each user's current online status, including for example, the user's identity, avatar type and last position. A rough last position is necessary, as we can estimate which super-peer currently manages this user. It is also updated whenever handoff between super-peers occurs. The DHT record is updated whenever an existing user leaves the current region managed by a previous super-peer. When an existing super-peer is contacted by a user to join its region, the super-peer refers to the DHT status record to check whether the joining user is double playing. If the joining user's record shows that it is

Table 1 Classification of 3D Content Types and Their Properties

	Content Type / Piece Dependency	Signature Scheme	Cost / Benefit
1.	complete mesh (none supported)	general digital signature	parse until loaded lowest overhead
2.	progressive mesh (linear stream)	hash chain	more overhead one-the-fly rendering
3.	point cloud independent piece	Rabin-based	highest overhead fast forward supported
4.	view-dependent mesh partially linear (DAG)	hash DAG	hybrid of 2. progressive and 3. independent

already in another region (but not the neighboring region), double playing can be detected. The super-peer that admits a user into its region then updates the DHT for the user's new status. Similar updates are performed by the super-peer when the user leaves the system.

Trusted super peer-based. An alternative method is based on trusted super peer hand-off procedures. An user can be served by one of the super peers since he is authenticated. A user will be migrated to another super peer management when his avatar move to another region. The super peers help the migrating user to executing hand-off procedure, so that the newer super peer believes the coming user is just playing once.

3.2 Authenticated Content Streaming

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

Digital signature is a widely used method to verify the authenticity and integrity of a given digital content. The publisher first generates a message authentication code (MAC) for publishing content by its private key. The user then takes the publisher's public key to verify the received content, to ensure that the content comes without modifications from the publisher. Cryptographic hash functions are also often used along with digital signatures because of their computational efficiency and ability to prevent existential forge [22]. In this section, we first classify the different potential 3D content by the~~ir~~ property, and then design the proper digital signature protocols to efficiently verify authenticity and integrity of a given 3D stream. The content in 3D streaming can be classified [illegible]hown in Table 1, and we describe them as follows.

1. The most basic form of content is one that needs to be fully downloaded before usage, and we call this the *whole model* content type. For example, for a normal mesh model, the user needs to download the whole mesh, before rendering can take place. For certain small mesh models, or large models that need to be transferred between the publisher and some content serving super-peers, this is a typical model format.
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, and the main benefit compared with the whole model is that a rough sketch can be rendered first, so that users can quickly have a preview to decide whether to stay or go somewhere else.

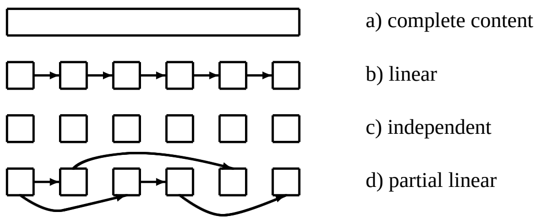

Fig. 2 Four basic types of 3D content. a) a whole content b) linear dependency c) fully independent d) directed acyclic graph (DAG) dependency

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 [21] are examples of independent stream, where the model is formed by patches of points. 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 particular patch of mesh data may have higher priority and need to be downloaded first.

We can imagine that 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 of the content. To generate and verify MAC efficiently for the above stream types, we present the following digital signature protocols. The notations used are described in Table 2.

3.2.1 Whole Model

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

$$\Delta = Signature_{private\ key}\{Hash(whole\ content)\}. \tag{1}$$

After a peer receives the content and its signature, it can first hash the whole content and then verify the signature by the publisher's public key. However, the authenticity and integrity of the received content cannot be verified with traditional digital signature protocol, if only partial content is available.

3.2.2 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

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 ith piece of a data content
$\mathbb{M}_i$	the hash value of the ith piece of a data content
$H(M)$	the hash value of message M

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 digital signature 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 3 (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.

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

Fig. 3 Production rules of authenticated mesh and texture data

The receiving peer can progressively verify the received content and then render the mesh and texture subsequently. An example using mesh data 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 $\mathbb{M}'_0 \overset{?}{=} H(meta'|M'_0|\mathbb{M}'_1)$, and the first refinement piece M'_1 is verified if $\mathbb{M}'_1 \overset{?}{=} H(M'_1|\mathbb{M}'_2)$. 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, here neither the delivery nor verification for intermediate pieces can be skipped.

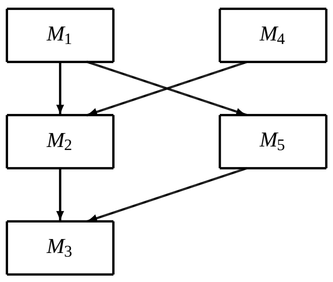

Fig. 4 Example dependency relation of partially linear stream

3.2.3 Independent Stream

For content that can be retrieved and used in any ordering, each data piece has to be signed individually since the pieces are independent to each other. In such [illegible]ario, the number of atomic digital signature operators cannot be reduced, so the fastest digital signature algorithm is needed. The Rabin public key cryptographic algorithm [23] can be applied to such [illegible]ent 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 an additional hash operator is needed for each piece. We describe the two main stages below:

Signing. When the publisher wants to publish a new content, 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. 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.

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

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.

With the above mentioned 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 are thus two tradeoff considerations between message overhead and computation.

3.2.4 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 3.2.3, 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.

3.3 Secure Content Update

3D content may be created, modified, or deleted during a user session, and in general, users need to obtain the newest version to stay synchronized with the world. Ensuring that new content is delivered timely to replace the old one thus is an important issue for commercial deployments. To provide guarantee on updates, each object needs to have its own ID, version number, expiration time and other related meta data. For example, the expiration time determines the duration that the object is valid~~,~~ it is provided so that peers can verify the validity of some content based on the content's signature. The publisher also has to sign these data before content publication. There are two cases to consider, respectively:

1. Updates for content that has been expired. All the publisher has to do is to re-sign and re-publish the new content. Retransmission of the content is not necessary if the content has not changed, the publisher just needs to sign the content with a new expiration time and publishes this digital signature.
2. Updates for content still within the expiration time. In this case, the publisher has to revoke the old version first. The publisher can broadcast a termination command with its signature to terminate the old content with high priority. Each peer has to eliminate the content from its storage if the authenticity of the signature is valid and the content exists in its storage. The revoke commands are kept in peers until a expiration time is reached. ~~Beside~~ revocation, the publisher also re-signs and re-publishes the new version at the same time.

However, to re-publish a newer version to interested ~~user~~ is important. In general, our consideration is that users will need the newer versions if they are currently looking at ~~this~~ object or will be browsing it in a short period. So we adopt two mechanisms here:

Through content delivery path. If each piece is downloaded from a different source, then content delivery path can be described as a DAG. If the predecessor nodes can record the successor nodes whenever some successor nodes download content from the predecessors, updates then can be broadcasted to the nodes who may need to see the new content.

Through super-peer. If each nodes check back with the super-peer whose region covers the objects periodically, then the super-peer can be aware of who is still alive and has the content. On one hand, if such information about who has certain content is kept, super-peers can then provide better contact lists for new nodes to find peers to exchange data. On the other hand, super-peers can notify peers who have downloaded previously with information about new versions. The peers can then either download the newer version if the object is still relevant, or erase the stored object otherwise.

Each peer that receives an updated stream can compare the object with its kept version. If the kept version is newer, then it tells the source node to update it. If the kept version is older, then it would verify the authenticity and update the data. The peer in general keeps only the latest version, if several versions exist. As it is assumed that the user is always interested in the newest version of the content, it is not necessary to keep different versions of the same content in the environment. For each content object, just the newest version needs to be kept.

4 Evaluation

4.1 Security Analysis

4.1.1 Double Playing Detection

The security of the proposed double playing detection is based on the online status DHT. No single, or a small group of users, can seriously disrupt its normal operations due to the following reasons.

1. As online records are stored distributively on all the super-peers using a hash function. No single user can control exactly where an item will be stored (this depends on both the ID of the item, and the number of currently online super-peers). Thus, it would be difficult for any nodes to provide specific advantage for a another, colluding node. In a distributed environment, the probability is negligible for resources to be controlled by a specific or some colluding users. Therefore, the user who will be detected and ferreted out the NVE within a short period if he attempts to logging into NVE more than one times simultaneously by a single account.
2. No super-peer can have total control over a user's status. Even if a particular super-peer is willing to admit users to its region, as a new check for double playing is performed every time a user crosses regions, such violation has only limited effects and cannot be lasting, unless the user always stays within a cheating super-peer's region.

So due to property (1), there would be little incentives for super-peers to modify what they maintain in their hash tables, as such acts cannot benefit any specific user; and due to (2), any special advantage that a given super-peer is willing to provide cannot last across regions.

The super-peers are also likely not controlled by a specific user or a small number of users.

4.1.2 Authenticated Content Streaming

We describe 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. Anyone 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, so it is necessary to achieve unforgeability for each piece. According to the strength of the Rabin cryptosystem [23], it is infeasible to deliver the square root under module n for a specific value without knowing p and q. The essential security is described in Lemma 1. However, the message is not exactly a specific value before computing the square root. M_i is variable, so this gives more advantage to the adversary. The remaining security is based on the adversary cannot generate a valid signature such that the least bits of the signature match the fixed message $ObjectID$. For the computer's power today, about 2^{100} enumerate operators is computationally infeasible [27]. 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 The *reduction* can be used to accomplish this 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 $S = \sqrt{M} \ (mod\ n)$. We know that if we can find two distinct square roots of a message, we can factor the modulus. Choose a value s and let $m = s^2$. Now s is a valid signature of m. Submit m to the black box. There is a one in two chance that it will produce the same signature s. If so, repeat this process. If not, we have both square roots of m and can recover the factors of n. Explicitly, an attacker randomly computes $C = m_1^2 \ (mod\ n)$ and then sends C to the oracle. The oracle responds $m_2 = SQRT_n(C)$ to the attacker. The attacker successfully computes $gcd(m_1 - m_2, n)$ to give p or q with 50% probability. Therefore, problem **B** is intractable if problem **A** is intractable.

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

4.1.3 Secure Content Update

Here we explain why a user certainly will get newer versions for his or her possessed content if they become available. Every content includes an expiration time in its meta-data section, and all of the content are signed by the publisher's private key. The user believes that the content is still valid if the signed expiration time is not expired. The user has to download a signed new expiration time if the kept content has expired, otherwise, the user automatically considers a kept content out of date. The user has the opportunity to get a newer expiration time even though the kept content is not expired yet when the publisher releases new content. Because the download behavior of user is unpredictable, and the same content can be downloaded from many other peers, the honest user is hard to be fooled by others unless most of them collude. Moreover, the user will be notified by the source node of his content, and if the user is still interested in the content, he will check with the super-peer who manages this content periodically for newer versions. Therefore, it is negligible probability that the user still keeps the old version after the publication of newer versions.

4.2 Performance Analysis

Overhead of secured content streaming. The communication overhead of secured content streaming is hash value and signature sizes. Formula 4 describes the communication overhead and data ratio. The overhead is relatively low if original stream is not divided into too small pieces.

$$\frac{number\ of\ piece * hash\ value\ size + signature\ size}{original\ stream\ size} * 100\% \tag{4}$$

5 Related work

3D Streaming for P2P-NVE. Hu et al. surveyed the 3D knowledge and present a 3D object streaming for P2P networks [12]. Cavagna et al. present a tree-based data structure to describe level of detail for a 3D object [25]. The HyperVerse [4] project investigates how globalscale persistent virtual environments can be provided.

Single sign-on. Single sign-on mechanisms are widely applied to distributed environment. The user can use certain distributed resources after he or she has been authenticated by the authentication server. The authentication server issues a ticket to the user to use distributed resources at any collaborating hosts. Any peer that maintains the desired resources can verify the ticket easily to ensure that this user has been authenticated by server, and then the resources are provided to the user. Some popular single sign mechanisms that have been applied to distributed environments include Kerberos, which is the first single sign-on system [31], [35] for distributively authenticating a user. However, the Kerberos system is based on symmetric key encryption so that a central online authentication and authorization server is needed. It thus has a scalability problem due to this online server. Some peer-to-peer authentication schemes have been proposed [16, 20, 30] in recent years (e.g., the FreeRADIUS Server is a daemon for unix and unix-like operating systems, and allows one to set up a radius protocol server that can be used for Authentication and Accounting various types of network access.)

Authenticated 3D streaming. To authenticate 3D streaming content, several schemes have been proposed. Wu and Cheung proposed a scheme that the 3D mesh model can be authenticated by publisher's public key [36].

Double spending, double playing. Service providers are always concerned by whether the user has paid once, but used more than what has been paid. In fact, double spending of e-cash is a difficult issue in distributed network environment. To the best of our knowledge, avoiding double spending is a still an unsolved problem without robust devices in the distributed environment. So, there is theoretically no mechanism to avoid double spending in open environments so far. However, to detect double playing users is practically feasible. Service provider can also block an opportunist after detecting some abnormal behaviors. [29].

6 Conclusion

3D streaming will provide better user experience for the growing number of virtual world and virtual globe applications. However, as it is difficult to support[illegible]sive number of users with traditional client-server architectures, peer-to-peer network[illegible] potential solution to share the central server's loading. In order to adopt peer-to-peer techniques, a virtual world may be partitioned into many regions managed by some super-peers. But how to ensure that[illegible]treaming~~ may be~~ performed securely, is a topic of concern~~ if~~ such[illegible]nique~~s were~~ to be adopted by commercial vendors. In this paper, we ~~discuss how~~ for P2P-based 3D streaming~~, certain security considerations can be addressed, specifically~~: 1) The detection of double-playing users. 2) The classification of the four content types for 3D streaming, and their proper digital signature protocols. 3) Content update mechanism that ensures users are always notified of the latest content securely.

There are still other topics worthy of exploration. For example, ~~making~~ proper accounting to charge users is essential for vendors, so how to ~~perform~~ such[illegible] in a P2P environment securely is also an important topic, which we will investigate as future work.

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. 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
19. O'Gorman L (2003) Comparing passwords, tokens, and biometrics for user authentication. In: Proceedings of the IEEE, vol 91, pp 2012–2040
20. 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
21. Pauly M, Gross M, Kobbelt LP (2002) Efficient simplification of point-sampled surfaces. In: Proceedings of IEEE Visualization, pp 163–170
22. Pointcheval D, Stern J (1996) Security proofs for signature schemes. Advances in Cryptology — EUROCRYPT '96 1070/1996:387–398
23. Rabin MO (1979) Digitalized signatures and public-key functions as intractable as factorization. MIT/LCS/TR-212, MIT Laboratory for Computer Science
24. Rosedale P, Ondrejka C (2003) Enabling player-created online worlds with grid computing and streaming. Gamasutra Resource Guide
25. 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
26. Schiele G, et al (2008) Consistency management for peer-to-peer-based massively multiuser virtual environments. In: Proc. MMVE
27. Schneier B (1996) Applied Cryptography, 2nd edn, John Wiley & Sons, chap 7
28. Singhal S, Zyda M (1999) Networked Virtual Environments: Design and Implementation. ACM Press
29. Smit G, Havinga P, Helme A (1996) Survey of electronic payment methods and systems. In: Proceedings of Euromedia
30. 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
31. Steiner JG, Neuman BC, Schiller JI (1988) An authentication service for open network system. Proceedings of the Winter 1988 Usenix Conference, pp 191–202
32. 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
33. Sung WL, Hu SY, Jiang JR (2008) Selection strategies for peer-to-peer 3d streaming. In: Proceedings of NOSSDAV
34. Teler E, Lischinski D (2001) Streaming of complex 3d scenes for remote walkthroughs. CGF (EG 2001) 20(3)
35. The MIT Kerberos Team (1980) The network authentication protocol. `http://web.mit.edu/Kerberos/`
36. 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