Chapter 5
Cryptanalysis of an Enhanced Event Signature Protocols for Peer-to-Peer Massively Multiplayer Online Games

Wei Yuan, Liang Hu, Hongtu Li and Jianfeng Chu

Abstract In 2008, Chan et al. presented an event signature (EASES) protocol for peer-to-peer massively multiplayer online games (P2P MMOGs). The authors declare that the EASES protocol is efficient and secure, and could achieve non-repudiation, event commitment, save memory, bandwidth and reduce the complexity of the computations. In 2010, Li et al. found a replay attack on the EASES protocol and proposed an enhanced edition to improve it. However, our works show their enhancement is still not secure as well. Finally, we made a discussion about this problem and point the weakness existence in this protocol.

Keywords Hash-chain · Cryptanalysis · Massive multiplayer online games · One-time signature · Peer-to-peer networks · MMOG · Network security

5.1 Introduction

Multiplayer on line games are a rapidly growing segment of Internet applications in the recent years. By providing more entertainment and sociability than single-player games, is fast becoming a major form of digital entertainment. In this kind

W. Yuan · L. Hu · H. Li · J. Chu (✉)
Department of Computer Science and Technology, Jilin University, Changchun, China
e-mail: chujf@jlu.edu.cn

W. Yuan
e-mail: yuanwei1@126.com

L. Hu
e-mail: hul@mails.jlu.edu.cn

H. Li
e-mail: li_hongtu@hotmail.com

X. He et al. (eds.), *Computer, Informatics, Cybernetics and Applications*,
Lecture Notes in Electrical Engineering 107, DOI: 10.1007/978-94-007-1839-5_5,

of games, all players should connect with the server to send and receive event updates. An event update is cryptographic protocol by which a player generates an event message and sends it to the server for updating the game states. Traditional massively multiplayer online games (MMOGs) are conventional client–server models that do not scale with the number of simultaneous clients that need to be supported. To resolve conflicts in the simulation and act as a central repository for data, peer-to-peer (P2P) architecture is increasingly being considered as replacement for traditional client–server architecture in MMOGs. P2P MMOGs have many advantages over traditional client–server systems due to their network connectivity and basic network services in a self-organizing manner. Whenever a player wants to play the finger-guessing game, an event message is sent to the server and the server processes all the events and updates the game states to ensure a global ordering for game executions and fair plays. However, P2P MMOGs communicate on the Internet raise the security issues such as cheating that a dishonest player can get valuable virtual items and even be sold for moneymaking. Recently, there are more and more efforts mounted to focus on event update protocols for online games in respect to the protection of sensitive communication and the provision of fair play.

In 2004, Dickey et al. [1] proposed a low latency and cheat-proof event ordering based on digital signatures and voting mechanism for P2P games. However, Corman et al. [2] later show that Dickey et al.'s protocol is unable to prevent all cheats as claimed, and propose an improvement called secure event agreement protocol, As digital signature requires a large amount of computations. To reduce heavyweight computations in every round of a game session, in 2008, Chan et al. [3] proposed an efficient and secure event signature (EASES) protocol using one-time signature with hash-chain key and claimed that their protocol has low computation and bandwidth costs, and is thus applicable to P2P-based MMOGs. Then they proposed a dynamic EASES protocol to avoid the pre-generation of hash-chain keys. Unfortunately, the EASES protocol is not secure and attackers can easily forge a series of update event to replace the original one. In 2010, Li et al. [4] found a replay attack on the EASES protocol and suggested a simple enhanced edition. However, their enhanced protocol still suffered from our attack.

In this chapter, we briefly review the enhanced protocol proposed by Li et al. Further, we introduce attacking methods to crack this protocol. Finally, we make a discussion on why our attack does.

5.2 Review of Chan et al.'s Event Signature Protocol for P2P MMOGs

Chan et al.'s event signature protocol has four phases: the Initialization Phase, Signing Phase, Verification Phase, Re-initialization Phase.

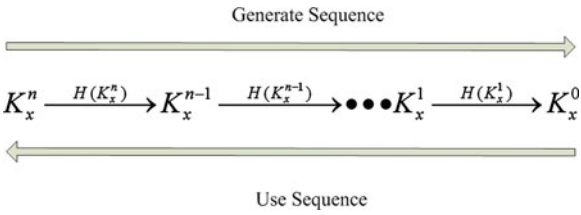

Fig. 5.1 Construction of hash-chain keys

5.2.1 Initialization Phase

In this phase, player P_i generates a series of one-time signature keys for a session and performs the following operations:

1. P_i chooses a master key MK_i to compute the nth one-time signature key $K_i^n = h(MK_i)$, where n represents the maximum number of rounds in a session.
2. P_i computes the other rth round one-time signature keys $K_i^{r-1} = H(K_i^r)$, where $r = (n-1), \ldots, 0$.
3. P_i signs the first one-time signature key by its private key to get the signature $\Delta_i = S_{sk_i}(K_i^0)$. The hash-chain keys K_i^r will be used in the reverse order of their production during the subsequent rth rounds, where $r = 0, 1, 2, \ldots, n-1$. Figure 5.1 shows the production of hash-chain keys.

5.2.2 Signing Phase

In this phase, if P_i wants to submit event update messages to other online players in a game session with n rounds, he/she performs the following operations:

1. P_i computes the 1st round one-time signature key δ_i^1 by computing $\delta_i^1 = H(K_i^1 \parallel U_i^1), \Delta_i, K_i^0$. Then, P_x submits the first round message to other online players.
2. P_i computes the second round one-time signature key $\delta_i^2 = H(K_i^2 \parallel U_i^2), U_i^1, K_i^1$ and submits it to other online players.
3. P_i computes the rth round one-time signature key $\delta_i^r = H(K_i^r \parallel U_i^r), U_i^{r-1}, K_i^{r-1}$ and submits it to other online players in the subsequent rth round, where $r = 3, 4, \ldots, n$.

5.2.3 Verification Phase

In this phase, each online player P_j receives the event update message $\delta_i^1 = H(K_i^1 \parallel U_i^1), \Delta_i, K_i^0$ from the player P_i and performs the following operations:

1. In the first round, P_j first verifies $\Delta_i \stackrel{?}{=} D_{pk_i}(K_i^0)$. If it holds, P_j confirms that the key K_x^0 is legitimate.
2. In the subsequent rth round, P_j verifies $K_i^{r-2} \stackrel{?}{=} H(K_i^{r-1})$ to check if the signature key K_i^{r-1} is legitimate, where $r = 2, 3, 4, \ldots, n$.
3. If above holds, P_j verifies $\delta_i^{r-1} \stackrel{?}{=} H(K_i^{r-1} \parallel U_i^{r-1})$ to check whether the update has been altered or not. If it passes verification, P_j convinces that no player has tampered with the update from P_i.

5.2.4 Re-Initialization Phase

If P_i wants to extend his/her game session for a few rounds, P_i regenerates a new master key and performs the following operations:

1. In the nth round, P_i chooses a new master key MK_i' and generates the new one-time signature keys $NewK_i^0, \ldots, NewK_i^n$. Then P_i has the new signature key $NewK_x^0$ with the key $K_i^n = H(MK_i')$ to generate $\delta_i^n = H(K_i^n \parallel U_i^n \parallel NewK_i^0), U_i^{n-1}, K_i^{n-1}$. P_i sends $\delta_i^n, U_i^{n-1}, K_i^{n-1}$ to other players as usual.
2. In the (n + 1)th round, P_i sends $\delta_i^{n+1} = H(NewK_i^1 \parallel U_i^{n+1}), U_i^n K_i^n NewK_i^0$ to other players.
3. In the (n + 2)th round, P_i sends $\delta_i^{n+2} = H(NewK_i^2 \parallel U_i^{n+2}), U_i^{n+1}, NewK_i^1, MK_i$ to other players.

Upon receiving new one-time signature keys from P_i, the other player, P_j, should perform the following verifiable operations:

1. In the (n + 1)th round, P_j verifies $\delta_i^n \stackrel{?}{=} H(K_i^n \parallel U_i^n \parallel NewK_i^0)$ to check if the new signature key $NewK_x^0$ is legitimate.
2. In the (n + 2)th round, in addition to the regular verifications, P_j must also verify $K_i^n \stackrel{?}{=} H(MK_i)$ If the above passes verification, P_j confirms the validity of $NewK_x^0$ The series of new one-time signature keys $NewK_i^0, \ldots, NewK_i^n$ can be used after the (n + 2)th rounds.

5.3 A Replay Attack on Chan et al.'s Protocol

In 2010, Li et al. has presented a replay attack to Chan et al.'s protocol then they made an enhanced one. In this section, we review the replay attack on Chan et al.'s protocol.

After one session ends, the attacker can get the one-time signature keys $K_i^0, K_i^1, \ldots, K_i^n$, the signature $\Delta_i = S_{sk_i}(K_i^0)$ and the master key, MK_i, which is transmitted from legal player P_i to other players. Then he can forge event updates

$U_k^0, U_k^1, \ldots, U_k^n$ with the valid signature keys $K_i^0, K_i^1, \ldots, K_i^n$ to cheat other players in P2P based MMOG like formula (5.1):

$$\begin{cases} \partial_i^{1'} = H\left(K_i^1 \parallel U_k^1\right), \Delta_i, K_i^0 & \textit{in the first round} \\ \partial_i^{r'} = H\left(K_i^r k^r\right), \Delta_i^{r-1}, K_i^0 & \textit{in the rth round} \end{cases} \tag{5.1}$$

Upon receiving event messages from the attacker, the other player computes the hash value of a given signature key K_i^r and fake event updates U_i^r to verify its equality to the previously received signature key $\delta_i^{r'}$, by computing $\delta_i^{r'} = H(K_i^r || U_k^r)$. Thus, the replay attack can not be prevented in Chan et al.'s protocol. Then Li et al. proposed their enhanced protocol.

5.4 Review of Chun-Ta Li et al.'s Enhanced Event Signature Protocol for P2P MMOGs

Li et al.'s enhanced EASES protocol also has four phases: the initialization phase, signing phase, verification phase, re-initialization phase.

5.4.1 Initialization Phase

In the initialization phase, player P_x generates a series of one-time signature keys for a session and performs the following operations:

1. P_x chooses a master key MK_x to compute the nth one-time signature key $K_x^n = h(MK_x)$, where n represents the maximum number of rounds in a session.
2. P_x computes the other rth round one-time signature keys $K_x^{r-1} = H(K_x^r)$, where $r = (n-1), \ldots, 0$.
3. P_x signs the first one-time signature key by its private key to get the signature $\Delta_x = S_{sk_x}(K_x^0 \parallel gno\#)$. Note that hash-chain keys K_x^r will be used in the reverse order of their production during the subsequent rth rounds, where $r = 0, 1, 2, \ldots, n-1$.

5.4.2 Signing Phase

If P_x wants to submit event update messages to other online players in a game session with n rounds, he/she performs the following operations:

1. P_x computes the first round one-time signature key δ_x^1 by computing $\delta_x^1 = H(K_x^1 \parallel U_x^1 \parallel gno\#), \Delta x, K_x^0, gno\#$. Then, P_x submits the first round message it to other online players.

2. P_x computes the second round one-time signature key $\delta_x^2 = H(K_x^2 \parallel U_x^2 \parallel gno\#), U_x^1, K_x^1, gno\#$ and submits it to other online players.
3. P_x computes the rth round one-time signature key $\delta_x^r = H(K_x^r \parallel U_x^r \parallel gno\#), U_x^{r-1}, K_x^{r-1}, gno\#$ and submits it to other online players in the subsequent rth round, where $r = 3, 4, \ldots, n$

5.4.3 Verification Phase

In the verification phase, each online player P_y receives the event update message from the player P_x and performs the following operations:

1. In the first round, P_y first verifies $\Delta_x \stackrel{?}{=} D_{pk_x}(K_x^0 \parallel gno\#)$. If it holds, P_y confirms that the key K_x^0 is legitimate and $gno\#$ is not a duplicate value; if not, it stops.
2. In the subsequent rth round, P_y verifies $K_x^{r-2} \stackrel{?}{=} H(K_x^{r-1})$ to check if the signature key $K_x^{r-1}$is legitimate, where $r = 2, 3, 4, \ldots, n$.
3. If above holds, P_y verifies $\delta_x^{r-1} \stackrel{?}{=} H(K_x^{r-1} \parallel U_x^{r-1} \parallel gno\#)$ to check whether the update has been altered or not. If it passes verification, P_y convinces that no player has tampered with the update from P_y.

5.4.4 Re-Initialization Phase

Whenever P_x wants to extend his/her game session for a few rounds, P_x regenerates a new master key and performs the following operations:

1. In the nth round, P_x chooses a new master key MK_x' and generates the new one-time signature keys $NewK_x^0, \ldots, NewK_x^n$. Then P_x has the new signature key $NewK_x^0$ with the key $K_x^n = H(MK_x')$ to generate $\delta_x^n = H(K_x^n \parallel U_x^n \parallel NewK_x^0 \parallel gno\#)$. P_x sends δ_x^n, U_x^{n-1}, K_x^{n-1} and $gno\#$ to other players in this round.
2. In the (n + 1)th round, P_x computes $\delta_x^{n+1} = H(NewK_x^1 \parallel U_x^{n+1} \parallel gno\#), U_x^n K_x^x$ $NewK_x^0$ and sends δ_x^{n+1} to other players.
3. In the (n + 2)th round, P_x sends $\delta_x^{n+2} = H(NewK_x^2 \parallel U_x^{n+2} \parallel gno\#)$, U_x^{n+1}, $NewK_x^1, gno\#, MK_x$ to other players.

Upon receiving new one-time signature keys from P_x, the other player, P_y, should perform the following verifiable operations:

1. In the (n + 1)th round, P_y verifies $\delta_x^n \stackrel{?}{=} H(K_x^n \parallel U_x^n \parallel NewK_x^0 \parallel gno\#)$ to check if the new signature key $NewK_x^0$ is legitimate.

2. In the (n + 2)th round, in addition to the regular verifications, P_y must also verify $K_x^n \stackrel{?}{=} H(MK_x)$. If the above passes verification, P_y confirms the validity of $NewK_x^0$. The series of new one-time signature keys $NewK_x^0, \ldots, NewK_x^n$ can be used after the (n + 2)th rounds.

5.5 Cryptanalysis of Li et al.'s Enhanced Protocol

Li et al.'s Enhanced protocol is based on Chan et al.'s EASES, they found the EASES protocol easily suffered from the replay attack, and try to add a unique game number, "gno#", to solve this problem. Unfortunately, the problem they found is not the main issue, thus attacker can crack the enhanced protocol in the similar way. The detailed steps are described as follows:

1. When P_x starts to send the first message, $\delta_x^1 = H(K_x^1 \parallel U_x^1 \parallel gno\#), \Delta x, K_x^0$, $gno\#$, to P_y, attacker P_z intercepts it and records K_x^0.
2. When P_x sends the second message, $\delta_x^2 = H(K_x^2 \parallel U_x^2 \parallel gno\#), U_x^1, K_x^1, gno\#$, to P_y, P_z intercepts it and record K_x^1. Then P_z forges a new message $\delta_x^1 = H(K_x^1 \parallel U_x^{1*} \parallel gno\#), \Delta x, K_x^0, gno\#$ and sends it to P_y, P_y verifies $\Delta_x \stackrel{?}{=} D_{pk_x}(K_x^0 \parallel gno\#)$ and records $K_x^0$to his memory if the equation holds.
3. When P_x sends the third message, $\delta_x^r = H(K_x^r \parallel U_x^r \parallel gno\#), U_x^{r-1}, K_x^{r-1}, gno\#$, $r = 3, \ldots, n$, to P_y, P_z intercepts it and record K_x^{r-1}. Then P_z forges a new message $\delta_x^{r-1} = H(K_x^{r-1} \parallel U_x^{(r-1)*} \parallel gno\#), U_x^{(r-2)*}, K_x^{r-2}, gno\#$, and sends it to P_y, P_y verifies $K_x^{r-3} \stackrel{?}{=} H(K_x^{r-2})$ and $\delta_x^{r-2} \stackrel{?}{=} H(K_x^{r-2} \parallel U_x^{(r-2)*} \parallel gno\#)$ and record $U_x^{(r-2)*}$ and K_x^{r-2} to his memory if the two equations hold.
4. Finally, all update event $U_x^1, \ldots, U_x^{n-2}$ are replaced to $U_x^{1*}, \ldots, U_x^{(n-2)*}$. Thus, Li et al.'s enhanced protocol still suffers from our attack.

5.6 Discussions

The aim that Chan et al.'s protocol is to reduce the computational cost. They believe that public-key cryptosystem require a large amount of computations. Then they propose the EASES and the dynamic EASES protocol. In these two protocols, only the first signature needs to be based on public-key cryptography, while others are based on the relationship between the hash-chain keys. As the above attack shows, they do not achieve their aim. Because attackers can easily tamper the hash value based on the public message, like the associated key. Further, the update event can be forged and the receiver can not find any questions.

Thus, in Chan et al.'s protocol, the public key is necessary. Li et al. notice the replay attack on Chan et al.'s protocol and make corresponding enhancement. However, the core problems are still existence in their enhanced protocol, so our attack does as well.

Acknowledgments The authors would like to thank the editors and anonymous reviewers for their valuable comments. This work is supported by the National Natural Science Foundation of China under Grant No. 60873235 and 60473099, the National Grand Fundamental Research 973 Program of China (Grant No. 2009CB320706), Scientific and Technological Developing Scheme of Jilin Province (20080318), and Program of New Century Excellent Talents in University (NCET-06-0300).

References

1. Dickey C, Zappala D, Lo V, Marr J (2004) Low latency and cheat-proof event ordering for peer-to-peer games. In: Proceedings of ACM international workshop on network and operating system support for digital audio and video, pp 134–139
2. Corman A, Douglas S, Schachte P, Teague V (2006) A secure event agreement (SEA) protocol for peer-to-peer games. In: The first international conference on availability, reliability and security
3. Chan M-C, Hu S-Y, Jiang J-R (2008) An efficient and secure event signature (EASES) protocol for peer-to-peer massively multiplayer online games. Comput Netw 52(9):1838–1854
4. Li C-T, Lee C–C, Wang L-J (2010) On the security enhancement of an efficient and secure event signature protocol for P2P MMOGs. ICCSA 2010 LNCS 6016:599–609