International Journal of Security and Its Applications
Vol. 3, No. 3, July, 2009

Cryptanalysis of an efficient and secure event signature protocol for peer-to-peer massively multiplayer online games

C. H. Wei and Y. H. Chin
Department of Computer Science
National Tsing Hua University, Hsinchu, Taiwan, R.O.C.
E-mail: chwei@cs.nthu.edu.tw

Abstract

To prevent the problem of cheating, Chan et al. proposed an efficient and secure event signature (EASES) protocol for peer-to-peer massively multiplayer online games. However, we show that the masquerade attacker not only can successfully collect valuable information about the messages being transmitted but also impersonate any player to replay the message cheating the innocent player. Chan et al.'s proposed EASES protocol fails to defend fair play for everyone in peer-to-peer massively multiplayer online games. Our cryptanalysis results are important for security engineers, who are responsible for the design and development in peer-to-peer massively multiplayer online games.

Keywords: *peer-to-peer, massively multiplayer online games, masquerade attacks, cheating*

1. Introduction

Event update protocol is cryptographic protocol by which a player performs an event message sent to the server that subsequently processes the events and updates the game states [2]. In comparison with massively multiplayer online games (MMOGs), traditional 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) architectures are increasingly being considered as replacements for traditional client-server architectures in MMOGs. In P2P architectures, game state maintenance and execution could be distributed to individual players; therefore, unfair cheating has been increasing in this environment.

The concept of a distributed architecture for a multiplayer in Internet is due to Diot, Gautier, and Kurose in 1999 [7][8] when they presented a technique called buckets, used to maintain the distributed synchronization mechanism that guarantees the state consistency of the game; however, this technique does not regard the problem of cheating. The cheating problem was considered by Baughman and Levine. The protocol was called the lockstep protocol [1] that has a probable anti-cheating guarantee and ensures fair play out of events under distribution. The concept, based on rounds for time, is called time buckets. A player reliably sends a cryptographic one-way hash of his state update is called a commitment, and then sends the plain-text update to reveal the commitment in the next round. Therefore, the protocol can prevent any player from knowing the other player's actions at any time. Lockstep protocol is a major advance in distributed protocols because it is probably secure against the fixed-delay. Unfortunately, the lockstep protocol has some defects. Among others, a new event ordering NEO protocol [6] based on MMOGs has been presented. In 2004, Gauthier Dickey et al. [6] proposed a low-latency to prevent cheating. NEO protocol provides a much lower latency than the previous event ordering protocols, which are limited by the

latency of the slowest player to any other player in the game. However, in 2006, Corman et al. [5] pointed out that NEO is unable to prevent cheating attacks and further proposed an improved protocol called a secure event agreement (SEA). In order to secure updates, the security of SEA protocol is based on one-way hash functions [9] and traditional digital signatures [3] to sign updates in every round. In 2008, Chan et al. [2] proposed an efficient and secure event signature (EASES) protocol, with non-repudiation for P2P-based MMOGs. Unlike previous protocols, such as NEO and SEA that use heavyweight digital signatures in every round of a game session, EASES protocol is an extension of the NEO protocol and SEA protocol. In EASES protocol, they combine the cryptographic techniques of one-way hash-chain functions and digital signatures. On the one hand, EASES only adopts a digital signature in the first round of the signing phase and computes lightweight hash-chain keys in all subsequent rounds; and on the other hand, EASES not only reduces computation/bandwidth/memory usage overhead but also integrates the dynamic topologies into algorithms and it is quite applicable to P2P-based MMOGs. Nevertheless, in this paper some security loopholes of their scheme will be pointed out, and the corresponding attacks will be described.

The rest of this paper is organized as follows. In the following section, we describe and review EASES protocol and show our attack in Section 3. Concluding remarks are given in Section 4.

Table 1. The notations of EASES protocol

P_i	the *i*th player in the game.
sk_i	player *i*'s private key.
pk_i	player *i*'s public key.
$(x\|\|y)$	concatenation of message x and y.
$H(x)$	a public one-way hashing function with input message x.
TP_i^r	a timestamp generated by P*i* in the *r*th round.
$S_{sk_i}(x)$	a message x signed by P*i*'s private key.
$D_{pk_i}(x)$	a message x decrypted by P*i*'s public key.
K_i^r	one-time signature key of P*i* in the rth round.
U_i^r	an event update of P*i* in the *r*th round.
δ_i^r	a signature signed by P*i*'s *r*th one-time signature key.
Δ_i	a signature signed by P*i*'s private key.

2. Review of EASES protocol

The notations of EASES protocol will be used in Table 1. In the EASES protocol proposed by Chan, Hu, and Jiang [2], there are four phases, namely: phase 1: before

starting the game; the player generates a series of keys in initialization for signing event updates. Phase 2: the player signs her/his event update in the signing phase; phase 3: the player can verify their accepted event updates; and phase 4: the player can re-generate her/his new signature in the re-initialization phase, if the key is not enough to use. In the following subsections we briefly review Chan et al.'s EASES protocol.

2.1. Phase 1: Initialization Phase.

After connection establishment and before starting the game, each player P_i generates a series of one-time signature keys for a session. Figure 1 shows the production of one-way hash-chain keys. P_i then performs the following operations:

(1) Choose 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) Compute the other rth round one-time signature keys $K_i^{r-1} = H(K_i^r)$, where r=(n−1), (n-2), ..., 0.

Sign the first one-time signature key by its private key to get the signature $\Delta_i = S_{sk_i}(K_i^0)$. Note that 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, …, n−1.

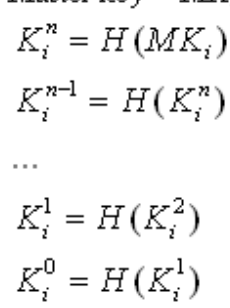

Figure 1. One-way hash-chain keys.

2.2. Phase 2: Signing Phase.

In this phase, P_i first computes the first round one-time signature key δ_i^1 by computing $\delta_i^1 = H(K_i^1 \| U_i^1), \Delta_i, K_i^0$, where U_i^1 is P_i's first event update. Then, P_i sends out the first round message δ_i^1. In the second round, P_i sends out $\delta_i^2 = H(K_i^2 \| U_i^2), U_i^1, K_i^1$. In the subsequent rth round, P_i sends out $\delta_i^r = H(K_i^r \| U_i^r), U_i^{r-1}, K_i^{r-1}$, where r=3, 4, ..., n. Due to space limitations, remaining rounds may be deduced by analogy to simplify the exposition of the procedure. Equation (1), below, shows the event message sent by P_i.

$$\begin{cases} \delta_i^1 = H(K_i^1 \| U_i^1), \Delta_i, K_i^0 & \textit{is the first round}; \\ \delta_i^2 = H(K_i^2 \| U_i^2), U_i^1, K_i^1 & \textit{is the second round}; \\ \delta_i^r = H(K_i^r \| U_i^r), U_i^{r-1}, K_i^{r-1} & \textit{in the subsequent round}. \end{cases} \quad (1)$$

2.3. Phase 3: Verification Phase.

In this phase, the player P_j after receiving updates from P_i, P_j first verifies $K_i^0 = D_{pk_i}(\delta_i)$. If it holds, P_j confirms that the key K_i^0 is legitimate; if not, it stops. In the subsequent *r*th round, P_j verifies $K_i^{r-2} = H(K_i^{r-1})$ to check if the signature key K_i^{r-1} is legitimate, where r=2, ..., n. If it holds, P_j verifies $\delta_i^{r-1} = H(K_i^{r-1} \| U_i^{r-1})$ to check whether the update has been altered or not. If it passes verification, P_j is convinced that no player has tampered with the update from P_i

2.4. Phase 4: Re-initialization Phase.

When the one-time signature keys have been used up in a session, P_i must re-generate the new one-time signature keys for further use. However, it is not necessary for P_i to start over from the initialization phase. A simpler way for P_i to perform the following operations in the nth and (n+1)th round exists.

(1) Choose a new master key, MK_i', to generate the new one-time signature keys $NewK_i^0, ..., NewK_i^n$. P_i then hash the new signature key $NewK_i^0$ with the key $K_i^n = H(MK_i)$ to generate $\delta_i^n = H(K_i^n \| U_i^n \| NewK_i^0)$. P_i send δ_i^n, U_i^{n-1}, and K_i^{n-1} to other players in the *n*th round.

(2) In the (n+1)th round, P_i computes $\delta_i^{n+1} = H(NewK_i^1 \| U_i^{n+1})$ and sends $\delta_i^{n+1}, U_i^n, K_i^n$, and $NewK_i^0$ to other players.

(3) In the (n+2)th round, P_i sends $\delta_i^{n+2} = H(NewK_i^2 \| U_i^{n+2}), U_i^{n+1}, NewK_i^1$, and the original master key, 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:

(4) In the (n+1)th round, P_j verifies $H(K_i^n \| U_i^n \| NewK_i^0) \stackrel{?}{=} \delta_i^n$ to check if the new signature key $NewK_i^0$ is legitimate.

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 accepts the validity of $NewK_i^0$.

The series of new one-time signature keys $NewK_i^0,...,NewK_i^n$ can be used after the (n+2)th rounds.

3. Cryptanalysis of the EASES protocol

The EASES protocol claimed to use an efficient and secure event signature, achieving a secure cheat-proof protocol. In this section, we show that passive attacks [10][11] cannot be fully prevented in the EASES protocol because the masquerade attacker (called Eve) not only can monitor communication channels between peer nodes in P2P networks but also can collect valuable information about the messages being transmitted. After collecting information from these communication channels, she can masquerade as any legal player to send a series of malicious updates to other players. The detailed steps of the cryptanalysis of EASES protocol are described as follows:

(1) In the initialization phase, the legal player chooses a master key MK_i to compute the nth one-time signature key $K_i^0,K_i^1,...,K_i^{n-1},K_i^n$, and the signature $\Delta_i = S_{sk_i}(K_i^0)$, which is transmitted from legal player P_i to other players P_j. Nevertheless, the signature information is collected by Eve who is an attacker and intercepts the signature as shown in Figure 2.

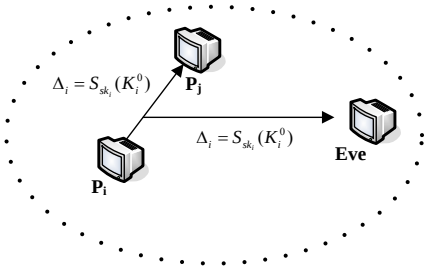

Figure 2. The signature information is collected in the initialization phase

(2) In the signing phase, Eve can arbitrarily masquerade as P_i to use valid signature keys $K_i^0,K_i^1,...,K_i^{n-1},K_i^n$, which are received from the initialization phase, and then Eve can send malicious event updates $U_i^{1'},U_i^{2'},...,U_i^{n-1'},U_i^{n'}$ to other victim players in P2P-based MMOGs. Eve masquerades as P_i, and sends the malicious message $\delta_i^{1'} = H(K_i^1 \| U_i^{1'}),\Delta_i,K_i^0$ to other players P_j in the first round as shown in Figure 3. In the subsequent rth round, Eve would send out $\delta_i^{r'} = H(K_i^r \| U_i^{r'}),U_i^{r-1'},K_i^{r-1}$ to the other players P_j. The equation (2) shows the malicious event message sent by Eve, where r = 2, ..., n.

$$\begin{cases} \delta_i^{1'} = H(K_i^1 \| U_i^{1'}), \Delta_i, K_i^0 & \textit{in the first round}; \\ \delta_i^{r'} = H(K_i^1 \| U_i^{1'}), \Delta_i, K_i^0 & \textit{in the subsequent rth round}. \end{cases} \quad (2)$$

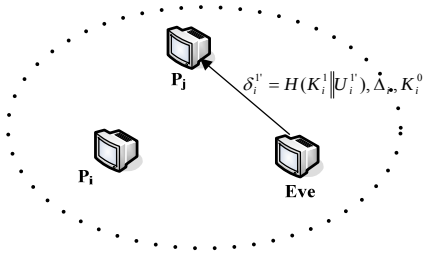

Figure 3. The malicious message is delivered in the signing phase

(3) In the verification phase, upon receiving event messages from Eve, the other players compute the hash value of a given signature key K_i^r and malicious message $U_i^{r'}$ to verify its equality to the previously received signature key $\delta_i^{r'}$, by computing $\delta_i^{r'} \stackrel{?}{=} H(K_i^r \| U_i^{r'})$. The masquerade attack is unpreventable in EASES protocol, because Eve adopts valid hash-chain signature keys to prevent other players from discovering the fraud. After collecting previous event update messages, $Old\delta_i^r$, sent by P_i, Eve may also replay them to other players P_j in a new session. Although P_i sends new event update messages $New\delta_i^r$ for a new session and Eve replaces $New\delta_i^r$ with $Old\delta_i^r$ in the signing phase, other players will continue to be fooled. Therefore, the verification of the event update message cannot be fulfilled using this protocol. EASES cannot resist a replay attack.

(4) In the re-initialization phase, Eve repeats the process from the initialization phase. Eve can receive one-time signature key $NewK_i^0, NewK_i^1, ..., NewK_i^{n-1}, NewK_i^n$, and the signature $\Delta_i = S_{sk_i}(NewK_i^0)$, and master key MK_i from an innocent player P_i.

4. Conclusion

Cheating is a key issue in online games. Whatever the rules that govern a game, some players will always be tempted to break or elude these rules so as to gain an unfair advantage over other players. In 2008, Chan et al. proposed an efficient and secure

event signature (EASES) protocol with non-repudiation for P2P-based MMOGs. However, in this paper, we have presented that there is a security leak inherent in EASES protocol, showing that their scheme violates the claim of achieving secure cheat-proof protocol. Eve can masquerade attacks and replays event update messages by the EASES protocol can succeed. For this reason, EASES protocol is insecure for P2P-based MMOGs. It is important that security engineers be made aware of this, if they are responsible for the design and development of P2P-based MMOGs.

Reference

[1] N. E. Baughman and B. N. Levine, Cheat-proof Playout for Centralized and Distributed Online Games, In IEEE Infocom, 2001.

[2] M. C. Chan, S. Y. Hu and J. R. Jiang, An efficient and secure event signature (EASES) protocol for.

[3] Y. F. Chang, and C. C. Chang, Robust t-out-of-n Proxy Signature Based on RSA Cryptosystems,International Journal of Innovative Computing, Information and Control, vol.4, no.2, pp.425-431,2008.

[4] M. C. Chan, S. Y. Hu and J. R. Jiang, An efficient and secure event signature (EASES) protocol for peer-to-peer massively multiplayer online games, Computer Networks, vol.52, no.9, pp.1838–1845, 2008.

[5] A. Corman, S. Douglas, P. Schachte, and V. Teague, A secure event agreement (SEA) protocol for peer-to-peer games, in: Proceedings of the First International Conference on Availability, Reliability and Security, 2006.

[6] C. Dickey, D. Zappala, V. Lo, J. Marr, Low latency and cheat-proof event ordering for peer-to-peer games, in: Proceedings of the ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Kinsale, County Cork, Ireland, 2004, pp. 134–139.

[7] C. Diot and L. Gautier, A Distributed Architecture for Multiplayer Interactive Applications on the Internet. IEEE Networks, 13(4), July/August 1999.

[8] L. Gautier, C. Diot, and J. Kurose, End-to-End Transmission Control Mechanisms for Multiparty Interactive Applications on the internet, In IEEE Infocom, 1999.

[9] M. S. Hwang, , C. C. Lee, S. K. Chong and J. W. Lo, A Key Management for Wireless Communications, International Journal of Innovative Computing, Information and Control, vol.4, no.8, pp.2045–2056, 2008.

[10] C. T. Li, and Y. P. Chu, Cryptanalysis of threshold password authentication against guessing attacks in ad hoc networks, International Journal of Network Security, vol.8, no.2, pp.166–168, 2009.

[11] C. T. Li, M. S. Hwang and Y. P. Chu, Improving the security of a secure anonymous routing protocol with authenticated key exchange for ad hoc networks, International Journal of Computer Systems Science and Engineering, vol.23, no.3, pp.225–232, 2008.