VCR and PEO revised

Emiliano Kargieman <ek@core-sdi.com>
Ariel Futoransky <futo@core-sdi.com>

CORE SDI S.A.
<http://www.core-sdi.com>

October 1998

Keywords: Cryptography, Hash functions, Security, VCR, PEO

Abstract

This paper focus on the security analysis of the cryptographic protocols PEO and VCR first introduced in [1] and with important applications in information security and auditing[1]. The analysis is oriented to prove the security of the protocols based only on general properties of cryptographic hash functions and independently of the functions used for the implementation. Towards this goal a modification to the original protocols is proposed that would strengthen the security[2].

[1] as of this wirting, the protocols have been succesfully implemented and tested in real-life applications such as a secure syslog daemon for UNIX, secure Event Logger for Windows NT and database auditing

[2] This paper is a mid-term result from an ongoing reaserch project on **new methods for auditing and information security**, currently under developement on the CORE Labs at CORE SDI S.A. in Buenos Aires, Argentina.

1 INTRODUCTION

A commonly used technique among computer hackers, and experimented thieves as well, is to erase their fingerprints from the crime scene. This means, usually, erasing or modifying the logs stored in the computer that will show them off when carefully looked at. This basic action, if done well, will make security auditing an imposible task in most cases. For when an intruder gains complete access to the system, she also gains the ability to read, modify or erase any such log. Let us define SECURE LOGGING as the ability to record a given amount of information on a given storage media and be able to check the authenticity of that record later. This definition says nothing about the security of the storage media where the information is recorded, we should assume that everybody can read, modify or erase it. If a record is erased we will know as a fact that is not authentic, for there should be something recorded as we know we recorded something there. Something different happens if the record is modified, we should have a way to tell it was modified even if we don't know what was the information recorded.

When an attacker gains absolute control of all system's resources standard cryptographic techniques are usually compromised, she can allways browse through the system memory to retrieve any symmetric or public key used for encryption and with that information, proceed to modify the logs stored. The protocols introduced within this paper are oriented to determinate if the information logged in a system before an intrusion has been altered. That is, given a system that append records periodically to a database, and an attacker that gains access to the system in a given instant of time, an auditor can establish if the data logged before the intrusion has been modified. The protocols don't guarantee what happens with the information recorded after the intrusion, since that moment the intruder has control of all the inputs and outputs of the system. The security of these protocols lays on the fact that the state of the system in the instant a record is appended cannot be reproduced with the information present on the system thereafter.

2 DESCRIPTION OF THE ORIGINAL PROTOCOLS

In this section the protocols are outlined from a theoretical point of view.

The intervenient parts are:

- Auditor: is who audits the authenticity of the recorded information.
- Source: is the source of the information to be stored.
- System: the system that records the information.

In both protocols the security relies mostly onthe confidentiality of the initial state or initial key (which we will call K_0) and will be generated at random by the auditor. The instant in which K_0 is generated will be the initial instant (0), the moment when the auditor proceeds to verify the recorded data will be called with the letter n. This way the instants

in which the protocols take place will be ordered $0 < i <= n$. The states K_i will be erased by the system, who keeps only the last state K_n. K_0 is known only to the auditor. During the process of verifying, the auditor regenerates from K_0 and the recorded data every K_i. The one-way hash function $H(K, D)$ processes the data (D) with the initial state (K) to obtain a digest. The feedback is acomplished when the last iteration digest is taken as the initial state for the current iteration.

2.1 PEO

Table 1 describes PEO in detail.

INSTANT	ACTION	DESCRIPTION
0	INIT $K_0 = Random()$	The auditor generates a random K_0 and stores it in a secure place
i	STORE $K_I = H(K_{i-1}, D_i)$	The source generates D_i, the system stores it and computes K_i using D_i and K_{i-1}. K_{i-1} is destroyed.
n	VERIFY $V_0 = K_0$ $V_j = H(V_{j-1}, D_j)$ $V_n == K_n$???	The auditor verifies K_n computing V_n as a function of K_0 and the D_js

Table 1: PEO protocol

(From the spanish phrase "Primer estado oculto", meaning "hidden first state")

If V_n is the same as K_n then the data in storage wasn't modified. It is easy to relate the security of this protocol with the security of the choosen one-way hash function. We will deal with this in the next section.

2.2 VCR

The outline of VCR is given next, for this protocol $E(K, D)$ is a symmetric encryption block algorithm that encrypts the message D using the key K. $E^{-1}(K, C)$ is the inverse function, that decrypts the cyphertext C using the key K. Some precautions must be taken if the keyspace of E(K,-) is greater than the posible outputs of H. The idea of VCR (as shown in

Table 2) is feedbacking the key used in the symmetryc algorithm with the hash of the original plaintext. This scheme allows us to retain the functionality of PEO while providing privacy for the data.

INSTANT	ACTION	DESCRIPTION
0	INIT $K_0 = Random()$	The auditor generates a random K_0 and stores it in a secure place
i	STORE $K_I = H(K_{i-1}, D_i)$ $C_i = E(K_{i-1}, D_i)$	The source generates D_i, the system computes K_i from D_i and K_{i-1}, encrypts D_i with K_{i-1} and stores it. K_{i-1} and D_i are destroyed.
n	READ AND VERIFY $V_0 = K_0$ $D_j = E^{-1}(V_{j-1}, C_j)$ $V_j = H(V_{j-1}, D_j)$ $V_n == K_n$???	The auditor reconstructs D_j as a function of C_j and K_0 and verifies K_n

Table 2: VCR protocol

(From the spanish phrase "Vector de claves remontante" meaning "Remounting Key Vector")

3 ANALYSIS

First we are going to formalize the definition of a one-way hash function.

We have:

$$H\colon \{0,1\}^N X \{0,1\}^N \rightarrow \{0,1\}^N \quad \text{and} \quad I \in \{0,1\}^N$$

for each $k \in \mathcal{N}$ and $M = (m_1, m_2, ..., m_k)$ with $m_i \in \{0,1\}^N$ for $1 <= i <= k$ we define:

$$H'(M) := H(H(...H(H(I, m_1), m_2), ...), m_k)$$

we say H (H') is a one-way hash function (a message digest) if the following conditions are satisfied:

$$\text{given } M \text{ is easy to compute } H'(M) \qquad (1)$$

$$\text{given } H'(M) \text{ is hard to compute } M \qquad (2)$$

$$\text{given } H'(M) \text{ is h.c. a } M' \text{ such that } H'(M) = H'(M') \qquad (3)$$

additionally, we say that H (or H') is *collision resistant* if:

$$\text{is hard to compute } M \text{ and } M' \text{ such that } H'(M) = H'(M') \tag{4}$$

furthermore, we say H is *pseudo-collision resistant* if:

$$\text{is hard to compute } I, m, I', m' \text{ such that } H(I, m) = H(I', m') \tag{5}$$

We consider an implementation of PEO where an intruder gains access to the system in the instant $i+1$, attempting to substitute the record D_i with a faked record M. The intruder has access to the following data: K_i and D_1 to D_i, he doesen't know either K_0 or K_1 to K_{i-1} because they where destroyed by the system. If the intruder pretends to stay unnoticed he has to generate a new set of data $\{D'_1, \ldots, D'_i = M\}$ and a new K'_i such that

$$H(H(\ldots H(K_0, D'_1) \ldots, D'_i) = K'_i$$

The prove that this is hard to accomplish is easily reduced to prove that:

$$\text{given } H(A, D) \text{ and } D \text{ is hard to compute } H(A, M) \tag{6}$$

If A is not known to the attacker he cannot compute $H(A, M)$ directly, one choice he has is to study the propagation of differences between messages through the hash function H[3].

another attack on the protocol could be derived if A is computed, meaning the the following assumption is broken:

$$\text{given } H(A, D) \text{ and } D \text{ is hard to compute } A \tag{7}$$

Is easy to see that (6) implies (7), for if A can be computed then $H(A, M)$ is easy to compute too.

Lets assume that an algorithm is given that computes A from $H(A, D)$ and D with low computational cost, then the following procedure could be used to contradict (5):

Given $I, M \in \{0,1\}^N$ let $R \in \{0,1\}^N$ be a random message, then the algorithm could be used to compute A_R such that $H(I, M) = H(A_R, R)$, finding a pseudo-collision on the hash function H and contradicting (5).

[3]This "propagation of differences" resembles a differential cryptoanalysis, although the main attempt here is to prove the security of the protocol based only on the general properties of hash functions, the question of wether hash functions should or should not resist this particular kind of attack is a valid one.

4 PROTOCOL MODIFICATION

The security of PEO in it's original definition (specifically the validity of (7)) was based on the complexity of generating pseudo-colissions for H (property (5)), a modification of the protocol is proposed such that the equivalent of (7) is deduced from the validity of (2). The principles upon which PEO was based (the presence of a hidden state) remain unchanged, but the feedback is obtained on a different way. The following table describes the new protocol in detail, the symbol $\veebar$ describes the XOR binary operator.

INSTANT	ACTION	DESCRIPTION
0	INIT $K_0 = Random()$	The auditor generates a random K_0 and stores it in a secure place
i	STORE $K_I = H'(K_{i-1} \veebar D_i)$	The source generates D_i, the system stores it and computes K_i using D_i and K_{i-1}. K_{i-1} is destroyed.
n	VERIFY $V_0 = K_0$ $V_j = H'(V_{j-1} \veebar D_j)$ $V_n == K_n$???	The auditor verifies K_n computing V_n as a function of K_0 and the D_js

Table 3: PEO-1 protocol

(Modification to PEO)

properties (6) and (7) are rewritten as follows:

$$\text{given } H'(A \veebar D) \text{ and } D \text{ is hard to compute } H'(A \veebar M) \tag{8}$$

$$\text{given } H'(A \veebar D) \text{ and } D \text{ is hard to compute } A \tag{9}$$

Now again, (8) implies (9).

Assuming an algorithm that computes A from $H'(A \veebar D)$ and D, and substituting variables, the following is deduced:

given $H'(M)$ and a random message R, the algorithm returns $M \veebar R$, since R is known we can compute

$$M = M \veebar R \veebar R$$

thus contradicting (2). From this follows that **if H' is a secure crypto hash function property** (9) **is valid**.

Note. $M_K(A) := H'(K \veebar A)$ can be thought as a *keyed hash function* depending on the key K. The validity of (8) could be answered then by looking into the theory of Message Authentication Codes.

5 BIBLIOGRAPHY

References

[1] A. Futoransky and E. Kargieman, *VCR y PEO, dos protocolos criptográficos simples*, 25 Jornadas Argentinas de Informática e Investigación Operativa, *July 1995*.

[2] B. Schneier and J. Kelsey, *Support for secure logs on untrusted machines* Proceedings of the 7th USENIX Security Symposium, *January 1998*.

[3] S. Stubblebine, V. Gligor *On Message Integrity in Cryptographic Protocols*, IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, May, 1992, pp. 85-104.

[4] S. Stubblebine, V. Gligor *Protocol Design for Integrity Protection*, IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, May, 1993, pp. 41-53.

[5] CORE SDI S.A. *Secure Syslog, source code*, CORELABS, <http://www.core-sdi.com/ssyslog>, *December 1997*

[6] D. Reed, *New Syslog*, <ftp://coombs.anu.edu.au/pub/net/misc/nsyslogd.tar.gz>, *July 1998*.