# 分散環境で保存されるログファイルにおける各ログエントリー間の順序関係保証方法に関する考察

安東　学　　松浦　幹太　　馬場　章

東京大学大学院 情報学環・学際情報学府
〒 113-0033 東京都文京区本郷 7-3-1

E-mail: qq16206@iii.u-tokyo.ac.jp, kanta@iis.u-tokyo.ac.jp, baba@hi.u-tokyo.ac.jp

概要: ログは不正行為の証拠として重要な情報であり，改ざんされないよう保護すべきものである．Schneier らは第 3 者機関との通信を最小限に抑えつつ，暗号学的に安全なログ保管方式を提案した．この方式は有用であるが，時刻情報に関して十分に考慮されていない．特に，複数エンティティが参加する場合，各ログエントリー間の順序関係を保証する仕組みが十分には考察されていない．本論文では，ログエントリー間の順序関係を保証するための方法を数種類採り上げ，有効性・実現性などを総合的に分析する．また，分析結果を受けて，順序関係保証方法に必要であると考えられる方向性について指摘する．

# An Analysis of Ensuring Order of Log Entries in Distributed Envirnment

Manabu ANDO　　Kanta MATSUURA　　Akira BABA

Interfaculty Initiative in Information Studies,
Graduate School of Interdisciplinary Information Studies,
The University of Tokyo,
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-0033, Japan

E-mail: qq16206@iii.u-tokyo.ac.jp, kanta@iis.u-tokyo.ac.jp, baba@hi.u-tokyo.ac.jp

Abstract: Log is important information for detecting attacks to computer or network systems. Thus, log needs to be protected against tampering, especially alteration of logs. Schneier et al. proposed secure audit log system, in which log is protected using cryptographic primitives and little amount of the communication between each entity and Trusted Third Party is needed. Their system is usful, but it is insufficient to consider time information. Especially, when many entities in a distributed environment paticipates in this system, the orders of log entries are not ensured. In this paper, we analyze several methods to ensure the order of log entries.

## 1　はじめに

ログはネットワーク環境における不正行為の証拠物として，極めて重要な役割を担っている．システムを安全に管理・運用するには，ログによる監査が必要となる．例えば，ログを監査することにより，攻撃者の場所や侵入経路，攻撃対象などを特定し，次の攻撃に備えたり，攻撃者を特定したりすることができる．また，ログをリアルタイムに監査し，異常な動作があったときにはシステム管理者へ通報する，といった侵入検知システム (Intrusion Detection) としての使用も考えられる．

ログを有効に活用するには，ログの改ざんを防止する必要がある．不正侵入者は，不正の痕跡を隠滅するためにログへのアクセス権を取得し，都合の悪いログエントリーを改ざん・消去しようとする．こ

れではログの監査をしても不正行為を検出することができない．従って，ログを安全に保管し，攻撃者が改ざんしたときには，それを検知できるような仕組みが必要となる．

Schneier らは，信頼できる第 3 者機関 (TTP: Trusted Third Party) を設け，ログを安全に保管する方式を提案した ([1][2])．この方式では，実際にログを保管するエンティティと TTP との通信は最小限度に抑えられ，TTP 以外の検証者によるログの検証が可能となっている．また，TTP が各エンティティのログを保管する，という設計にはなっていないため，多数の参加者があったとしても，実現性の高い方式である．以後，本論文ではこの方式を「SK 方式」と表記する．

SK 方式は極めて有効性の高いシステムであると考えられるが，各ログエントリーの時刻情報に関しては十分に考察されていない．特に，TTP が 1 に対し，複数のエンティティが参加するとき，各ログエントリー間の順序関係を明確にすることが困難である．しかし，[3] で分散環境における時刻情報の保証方法が議論されているように，ログにおいても同様の議論が必要である．

ログエントリーの順序関係を保証する仕組みは様々考えられる．例えば，デジタルタイムスタンプを定期的に申請する，というような方法も考えられる．また，[1] では，各エンティティ間で通信を行い，ある時点 $t$ において 2 者が同期している，というログエントリーを残す方式が提案されている．この記録を基準とし，相対的順序関係をある程度保証することもできる．

本論文では，複数エンティティが参加する SK 方式に基づくログシステムにおいて，各ログエントリーの順序関係を保証するための仕組みについて分析する．仮定や前提条件を様々に設定し，各方式がどのような長所・短所を有しているかを明確にすることが本論文の目的である．

# 2 SK 方式の概要

## 2.1 エンティティの定義

SK 方式では，以下の 3 者のエンティティが存在する．

- *Untrusted Machine: U*　ログを保管するエンティティであるが，対タンパ性が保証されていない．従って，不正侵入され，ログを改ざんされる可能性がある．
- *Trusted Machine: T*　完全に信頼できるエンティティであり，攻撃されることはない．
- *Verifier: V*　ログ（の一部）を検証することができる．

## 2.2 ログ保管プロトコル

プロトコルの詳細を記す前に，表記について整理する．

$D_j$: $j$ 番目のログデータ.

$W_j$: $D_j$ のタイプ．パーミッションマスクとして機能し，$V$ による検証が可能かどうかを規定.

$E_{K_j}(D_j)$: 共通鍵暗号方式（鍵は $K_j$）による暗号化された $D_j$.

$Y_j$: ハッシュチェーン
$(Y_j = hash(Y_{j-1}, E_{K_j}(D_j), W_j))$ .

$A_j$: Message Authentication Code(MAC) 生成鍵.

$Z_j$: $Y_j$ の MAC 値（鍵は $A_j$）.

$L_j$: $j$ 番目のログエントリー
$(L_j = (W_j, E_{K_j}(D_j), Y_j, Z_j))$ .

SK 方式のログ保管プロトコルを図 1 に示す．鍵が

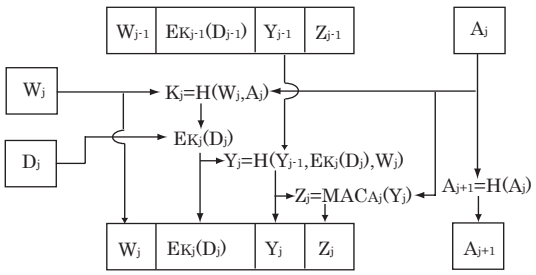


図 1: ログ保管プロトコル

更新されたとき，前回使用された鍵は完全に消去される．図 1 では，$K_j$ と $A_{j+1}$ が生成されると，$K_{j-1}$ と $A_j$ は消去される．鍵更新作業にハッシュチェーンを用いていることにより，以前に使用した鍵を生成することが不可能となる．従って，攻撃者が $U$ の制

| | $U$ | | | $T$ |
|---|---|---|---|---|
| forms | $K_0, d, d+, ID_{log}, C_U, A_0$ | | | |
| | $X_0 = p, d, C_U, A_0$ | | | |
| | $M_0 = ID_U, PKE_{PK_T}(K_0),$ $E_{K_0}(X_0, SIGN_{SK_U}(X_0))$ | $\rightarrow$ | verifies | $M_0$ |
| forms | $L_0$, $D_0 = d, d+, ID_{log}, M_0$ | | | |
| | | | forms | $X_1 = p, ID_{log}, hash(X_0)$ and $K_1$ |
| verifies | $M_1$ | $\leftarrow$ | | $M_1 = p, ID_T, PKE_{PK_U}(K_1),$ $E_{K_1}(X_1, SIGN_{SK_T}(X_1))$ |
| forms | $L_1$, $D_1 = M_1$ | | | |

図 2: 初期設定プロトコル　$K$: セッション鍵, $d$: 現在時刻, $d+$: 期限切れ時刻, $ID_{log}$: ログファイル識別子, $C_U$: $U$の公開鍵証明書, $A_0$: $A$ の初期値，$p$: プロトコルバージョンやナンス (nonce) などプロトコル実装で組み込まれるパラメータ，$E_K$: 共通鍵暗号方式，$PKE_{PK}$: 公開鍵暗号方式，$SIGN_{SK}$: 電子署名

御権を得たとしても，それ以前に記録されたログエントリーを閲覧することも，改ざんすることもできない．

また，$A_0$ は $T$が保管し，以後誰の手にも渡ることはない．もし，$A_0$ が搾取されれば，全ての鍵を生成することができ，全てのログファイルを閲覧，または改ざんすることができてしまう．

### 2.3 ログファイルの初期設定

$U$と $T$が通信するのは，初期設定時と検証作業時のみである．ここでは，ログファイルの初期設定について概説する．初期設定で重要なのは，$A_0$ の生成と受け渡しである．[1] では，$U$が $A_0$ を生成し，$T$ へ送る方式が採用されている．初期設定プロトコルを図 2 に示す．このようなプロトコル動作に従って，$A_0$ が $T$へと受け渡しされる．以降，$U$と $T$による通信は行われない．$U$は $L_2$ 以降を前節のプロトコルに則って作成し，$T$は $U$から受けた $A_0$（と $U$の識別子）を保管する．

## 3 SK 方式の考察と論点整理

### 3.1 SK 方式における時刻情報について

SK 方式では，ログファイル初期設定時に現在時刻 $d$ を $T$へ送る．$T$は自身が所持するクロックを参照し，$d$が正当なものと認められるか否かを決定する．このことから，[1] では明文化されていないが，以下の仮定が存在する．

**仮定 1.** $T$のクロックが示す時刻情報を基準とし，各 $U$はこれに同期している．また，各 $U$はこの基準に従うことに同意している．

$T$のクロックが示す時刻情報が，完全に正確なものである必要はない．基準となり，$U$がこの基準時刻と同期していれば，初期設定プロトコルは正確に動作する．従って，複数の $U$が存在する場合，仮定 1 より各 $U$が保存する $L_0$ エントリーの順序関係を保証することができる．

しかし，$L_1$ 以降のログエントリーに関しては，時刻情報を保証することができない．$T$は $L_0$ の時刻を知っているため，それより遡って時刻を偽造することはできないが，$L_0$ 以降であればどのようにでも偽造することができる．SK 方式には以上のような時刻情報に関する問題が発生すると考えられる．

### 3.2 論点整理

$T$が 1 に対し，複数の $U$が存在するようなシステムを考える．$T$は各 $U$から得た $A_0$ を保管し，$U$は $L_0$ 作成時のみ $T$と通信する．前節の初期設定プロトコルにおいて，現在時刻 $d$が $T$により検証されているため，ログファイル作成開始時刻については保証されている．しかし，以降のログエントリーについては，時刻情報が保証されない．このとき，異なる $U$が保存するログエントリーの順序関係を保証することができない（図 3）．ログデータに付与される時刻情報は信頼できるものではない．従って，何らかの方法によりログエントリーの時刻情報，または順序関係を保証する必要がある．

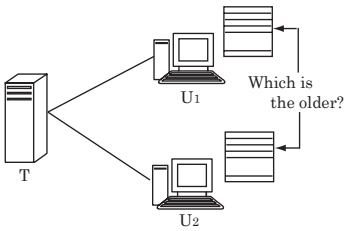


図 3: 順序関係に関する問題

# 4 ログエントリーの順序関係保証

本章では，図 3 にみられる問題を解決するための方法を数種類採り上げ，比較・分析する．

## 4.1 $T$による認証

各 $U$が頻繁に $T$と通信できるような状態を考える．すなわち，

**仮定 2．** $U$と $T$との通信経路は完備されている．また，$T$は莫大な計算能力を有する．

このような環境では，

**方法 1．** ログエントリーが追加されるときは，必ず $T$と通信し，現在時刻を保証してもらう．

という方法が考えられる．つまり，ログエントリー追加時に $U$は現在時刻 $d$を $T$へ送る．$T$は $d$を正当な現在時刻と認めることができれば，ログエントリー追加を認める．ログエントリー追加時刻を $T$が認証することにより各ログエントリーの順序関係を明確にする，という方法である．特徴を以下にまとめる．

- $U$から送られる時刻情報を $T$が常に認証作業をするため，精度の高い順序関係保証機構を実現できる．
- $U$同士による通信がないため，特定のエンティティに対する妨害攻撃ができない（この種の攻撃については後述する）．
- 仮定 1・仮定 2 が大前提となる．また，$T$は各 $U$のログを保存しないが，$U$から受け取った時刻情報に関するログを保存しなければならない．
- $T$への信頼が絶対的なものとなる．$T$がある $U$と結託した場合，特定の $U$によるログエントリー追加申請を認めない，という妨害攻撃が可能となる．

## 4.2 タイムスタンプの利用

$T$による時刻情報保証の代わりに，タイムスタンプサービスを利用する，という方法が考えられる．すなわち，

**方法 2．** ログエントリーを追加するとき，タイムスタンプ生成機関にログエントリー（のハッシュ値）を送信し，タイムスタンプを生成してもらう．

という方法である．ここでは，

**仮定 3．** $U$とタイムスタンプ生成機関との通信路は完備されている．また，使用されるタイムスタンプ方式は安全なものである．

と仮定する．この方法では，各ログエントリーがタイムスタンプによって示される時刻に存在していた，ということを保証できる．従って，タイムスタンプの検証により，各ログエントリーの順序関係を明確にすることができる．方法 2 の特徴をまとめる．

- $U$同士による結託や $U$と $T$との結託があった場合でも，相対的順序関係が保証される．
- 時刻情報に関して，$T$への信頼度を下げることができる．すなわち，仮定 1 を取り除くことができる．
- $U$とタイムスタンプ生成機関との通信量が膨大なものになってしまう．
- タイムスタンプ方式の安全性に依存している．

## 4.3 Cross-Peer Linkage の利用

### 4.3.1 Cross-Peer Linkage について

Cross-Peer Linkage とは，異なる $T$に管理されている $U$間で相互認証することによって，ログ改ざんの時期を検出するための方法である ([1])．2 者の $U$が通信を行い，その証拠をログエントリーとして記録しておけば，どの時点以降のログエントリーが改ざんされているのかを明確にすることができる．また，$T$が攻撃されていたとしても，ログの改ざんを検出することができる．Cross-Peer Linkage プロトコルを図 4 に示す．検証時には，$Y$ の値を検証することにより，改ざんを検知することができる．

| | $U_1$ | | | $U_2$ |
|---|---|---|---|---|
| forms | $L_j, D_j = ID_{U_2}, d_0$ | | | |
| | $M_1 = p, Y_j, d_0$ | $\rightarrow$ | verifies | $M_0$ |
| | | | forms | $L_i, D_i = ID_{U_1}, d_0, Y_j$ |
| verifies | $M_2$ | $\leftarrow$ | | $M_2 = p, Y_i$ |
| forms | $L_{j+1}, D_{j+1} = ID_{U_2}, Y_i$ | | | |

図 4: Cross-Peer Linkage プロトコル $d_0$: 現在時刻，$Y$: ハッシュチェーン，$p$: プロトコルバージョンやナンス (nonce) などプロトコル実装で組み込まれるパラメータ．なお，図では省略したが，各ログエントリーには Cross-Peer Linkage を実行しているというフラグが含まれる．

#### 4.3.2 順序関係保証機構への応用

Cross-Peer Linkage は，ログエントリーの順序関係保証機構として応用することができる．Cross-Peer Linkage は，任意の $U$ 同士がある時刻 $d$ に通信したという記録がログエントリーとして残る．つまり，同期した証拠がログエントリーとして記録されていることになる．従って，

**方法 3．** Cross-Peer Linkage によって記録されるログエントリーを同期した証拠として扱い，相対的順序関係を保証するための基準とする．

という方法が考えられる．また，

**仮定 4．** $U$ 同士の通信には，安全な通信路が用いられる．

と仮定する．また，仮定 1 より $L_0$ エントリーに関しては順序関係を明確にすることができるため，各 $L_0$ も基準となる．従って，各 $U$ が頻繁に Cross-Peer Linkage を実行することによって，より精度の高い順序関係保証機構を実現することができる．以下に特徴をまとめる．

- $T$ を除く TTP を必要としない．また，$T$ と $U$ との通信量は SK 方式に等しい．
- 精度を高めるには，$U$ 同士の頻繁な通信が必要となる．つまり，Cross-Peer Linkage による基準が少なければ精度は低くなる．また，基準の間に記録されるログエントリーについては順序関係を明確にすることができない．
- 2 者以上の $U$ が結託することにより，特定の $U$ に対する妨害攻撃が可能となる．

# 5 考察

## 5.1 各方式の比較・分析

方法 1 は，時刻情報をオンラインで認証するため，極めて高い精度でログエントリーの順序関係を保証することができる．また，$U$ と $T$ との一対一通信であるため，他者による，または他者への妨害攻撃も困難になる．一方で，$T$ への多大な負担が懸念される．これは計算量や通信量の負担だけではなく，信頼度という意味においても $T$ への負担が増大してしまう．方法 1 では $T$ への負担が増大してしまい，SK 方式の長所を保つことができない．このような問題を解決するには，一定期間毎にログエントリー追加要求を $T$ へ送り，これを基準として相対的順序関係を保証する，という方式も考えられる．しかし，精度を高めるためには「一定期間」を短くする必要が生じるため，根本的な解決には至らない．従って，方法 1 は仮定 2 のもとでのみ実現可能な方法であると考えられる．

方法 2 は $T$ の負担が従来の方式と等しい[1]．その代わりに，新たな TTP を導入する必要が生じる．更に，$U$ の負担が増大する．また，タイムスタンプ方式に完全に依存することになるため，その構成を熟慮する必要がある ([4] 等を参照)．この方法についても，前述したように，一定期間毎にタイムスタンプ発行要求を送信する，という方法が考えられる．問題点も方法 1 と同様である．

方法 3 は精度に関する問題がある．各 $U$ が基準となるログエントリーを頻繁に作成しなければならない．$U$ 同士が頻繁に通信しても順序関係の不明瞭なログエントリーは必ず存在するため，精度がある程度以上に高くなることはない．また，$T$ との通信な

[1] タイムスタンプ生成機関を $T$ が兼ねるとすれば，負担が増大するが，ログシステムにおける $T$ としては従来方式と等しい．

しで $U$ 同士が通信する場合，各 $U$ がどのようにして通信相手を認識するか，という問題がある．各 $U$ の ID とアドレスが配布されるということであれば，$U$ には誰と通信しようとしているかがわかる．このような状況では，特定の $U$ に対する妨害攻撃が可能になる．例えば，$U_1$ に対して他の $U$ が結託し，$U_1$ からの Cross-Peer Linkage 要求を無視する，ということで合意していた場合，$U_1$ のログエントリーに関して順序関係を不明瞭にすることができる．このような攻撃はある種の DoS(Denial of Service) 攻撃と考えられる．このような攻撃はタイムスタンプにおける攻撃例としても指摘されており ([5][6])，無視することのできない問題である．

### 5.2 問題解決の方向性

方法 3 の DoS 攻撃を防ぐには，通信相手を知ることができないような方法であればよい．つまり,「どのエンティティと通信して Cross-Peer Linkage を動作させようとしているのか」を各 $U$ が自分で決めることはできない，という方式にすればよい．例えば，Cross-Peer Linkage を $T$ 経由で行うことにより，$U$ 間の匿名性を保証することができる．しかし，$T$ 経由であるならば，方法 1 を採用すればよい．従って，$T$, 若しくは他の TTP を利用せずに匿名性をもたせることのできる方法を考察する必要がある．

また，他の $U$ や $T$, 異なる機能を提供する TTP との通信なしで順序関係保証の証拠を記録するという方法も考えられる．例えば，(1) 予測不可能であり，(2) 時刻情報との関連があり，(3) 公的に証明することが可能である，という性質の値（株価情報など）を使用し，ログエントリーに組み込む．ログエントリーとこの値を検証することにより，正当な時刻情報であるかを判断する．この方法では他者との介在がないため，DoS 攻撃も困難になる．更に，公的に提供される時刻が正確であるとするならば，絶対的時刻情報の保証にもなる．

## 6 おわりに

ログを安全に保管するためには，SK 方式が適当であると考えられる．しかし，SK 方式においては，時刻情報の扱いが十分に検討されていない．[1] で提案されている Cross-Peer Linkage プロトコルを使用することによって，ログエントリーの相対的順序関係をある程度の精度で保証することはできるが，特定のエンティティに対する DoS 攻撃が可能になるなど，問題もある．本論文では，Cross-Peer Linkage の他に，$T$ が時刻を保証する方法やタイムスタンプ発行機関を利用する方法を採り上げ，比較・分析を行った．実現性などを考慮した結果，(1) 通信状況等の条件によっては，$T$ が時刻情報を保証する方法やタイムスタンプ発行機関を利用する方法も実現できる，(2)Cross-Peer Linkage を利用するのであれば，匿名性のあるプロトコルを実現する，(3) 他者との通信を必要としない順序関係保証機構も考えられる，という結論を得た．(2) と (3) の方法に関する検討を今後の課題としたい．

## 参考文献


[1] B. Schneier and J. Kelsey:“Cryptographic Support for Secure Logs on Untrusted Machine,” In *Proc. of the 7th USENIX Security Symposium*, pp.53-62, Jan. 1998.

[2] J. Kelsey and B. Schneier:“Minimizing Bandwidth for Remote Access to Cryptographically Protected Audit Logs,” Second International Workshop on the Recent Advances in Intrusion Detection (RAID’99), Sep. 1999.

[3] P. Maniatis and M. Baker:“Secure History Preservation through Timeline Entanglement,” In *Proc. of the 11th USENIX Security Symposium*, Aug. 2002.

[4] M. Just:“Some Timestamping Protocol Failures,” In *Proc. of the Symposium on Network and Distributed Security (NDSS98)*, Mar. 1998.

[5] 宇根正志，松浦幹太，田倉昭:“デジタルタイムスタンプ技術の現状と課題,” IMES Discussion Paper Series No.99-J-36, 日本銀行金融研究所, Oct. 1999.

[6] 山田明, 清本晋作, 田中俊昭, 中尾康二:“発行サーバの不正に耐性のあるディジタルタイムスタンプ方式,” 2002 年暗号と情報セキュリティシンポジウム (SCIS2002) 予稿集, Vol.II, pp.621-626, Jan. 2002.