コンピュータセキュリティ 11−10
（2000．9．8）

タイムスタンププロトコル Cuculus と PKITS の安全性に関する一考察

宇根 正志†* 松本 勉†
une@mlab.jks.ynu.ac.jp tsutomu@mlab.jks.ynu.ac.jp
†横浜国立大学大学院工学研究科人工環境システム学専攻
*日本銀行金融研究所

あらまし
本稿では，ディジタル署名やハッシュ関数の安全性低下がタイムスタンププロトコル Cuculus と PKITS の安全性に与える影響について考察する．この結果，Cuculus では，攻撃者がタイムスタンプ生成機関と結託可能，かつ，ハッシュ関数の second-preimage が容易に探索可能な場合，タイムスタンプが改ざん可能であることを明らかにした．一方，PKITS では，攻撃者がタイムスタンプ生成機関と結託可能な場合，ディジタル署名やハッシュ関数の安全性低下に関わらずタイムスタンプの改ざんが可能であることを明らかにした．

A Study on Security of Time Stamping Protocols Cuculus and PKITS

Masashi Une†* Tsutomu Matsumoto†
une@mlab.jks.ynu.ac.jp tsutomu@mlab.jks.ynu.ac.jp

† Division of Artificial Environment and Systems, Graduate School of Engineering
Yokohama National University
*Institute for Monetary and Economic Studies, Bank of Japan

Abstract
This paper reports results of our analysis of security of time stamping protocols Cuculus and PKITS under the condition that employed digital signature scheme and/or hash function became unreliable. We mainly obtained the following two findings. The first is that if collusion with the time-stamping authorities (TSA) is possible and if the hash function is not second-preimage resistant, attacker can manipulate time stamps of Cuculus. The second is that if collusion with TSA is possible, attacker can manipulate time stamps of PKITS no matter how secure cryptographic techniques are employed.

1. はじめに

近年，電子商取引や電子媒体での文書管理を進める動きが活発化しており，後々の係争や情報公開等に備えて，ディジタル文書やその取扱履歴を長期間保管する技術が必要となっている．こうした中，あるデータが特定時刻に存在し，その時刻以降データ改ざんされていないことを第三者に証明する機能を有するタイムスタンプ技術が注目されている．

これまでに様々なタイムスタンプのプロトコルが提案されている[8]が，その多くは安全性をディジタル署名等の暗号技術の安全性に依存している．これらの暗号技術は，その解読技術の進歩や事故等により，安全性が低下することがあるが，そうなった際にタイムスタンプの信頼性が損われてしまうことは望ましくない．このような問題意識に基づく安全性評価は，タイムスタンプ以外の分野でもいくつか発表されている[4, 9]．

本稿では，タイムスタンプの具体的なプロトコルとして Cuculus[1, 2]と PKITS[3]を取り上げ，暗号技術の安全性低下がプロトコル全体の安全性に与える影響を考察する．Cuculus では，攻撃者がタイムスタンプ生成機関と結託可能，かつ，ハッシュ関数の second-preimage（後述）を容易に探索可能な場合，タイムスタンプを容易に改ざんできることを示す．また，PKITS では，攻撃者がタイムスタンプ生成機関と結託可能な場合，ディジタル署名やハッシュ関

数の安全性によらずタイムスタンプを容易に改ざんできることを示す.

以下では，まず考察対象とする Linking Protocol について 2.で説明し，3.で攻撃の前提・目的を説明する．4.及び 5.では Cuculus と PKITS の概要と考察結果をそれぞれ説明し，6.では 2 つのプロトコルに関する考察結果の比較を行う.

2. タイムスタンププロトコルの種類

タイムスタンプのプロトコルは，タイムスタンプの生成方法により，Linking Protocol，Simple Protocol，Distributed Protocol に分類される[7, 8].

2.1 Linking Protocol

2.1.1 プロトコルの概要

Linking Protocol は，タイムスタンプ生成機関（TSA : Time-Stamping Authority）が複数のタイムスタンプを相互に関連付けるリンク情報を生成し，これを基にタイムスタンプを生成する方式である．プロトコルは，タイムスタンプ生成を要求する利用者，タイムスタンプの検証を行う検証者，タイムスタンプの生成や，その検証に必要な情報の提供を行う TSA から構成される．ある時点のリンク情報とハッシュ値を結合・ハッシュ化して次のリンク情報とし，リンク情報の連鎖を生成する方法や，木構造を用いて複数のハッシュ値を集約する方法等がある．本稿では前者の方式を考察対象とし，特に，詳細な仕様が公表されている Cuculus と PKITS を具体的評価対象とする.

Linking Protocol は，TSA を信頼できなくても，リンク情報を新聞等のメディアを通じて公表すること等によってプロトコル全体の安全性を確保できる．ただし，タイムスタンプ生成・検証時の計算量が比較的多くなる.

2.1.2 タイムスタンプの生成・検証（図 1）

2.1.2.1 説明に利用する記号

- h_u, h_{tsa} : 利用者及び TSA のハッシュ関数
- SIG_u, SIG_{tsa} : 利用者及び TSA のディジタル署名関数（署名生成鍵は一定）．出力 Y は入力 X と X に対するディジタル署名 $S_i(X)$ (ただし i=u, tsa)から成り，$Y=SIG_i(X)=[X, S_i(X)]$とする.
- H, H_n : 利用者が TSA に送付するタイムスタンプ対象データのハッシュ値を H とし，それが TSA において n 番目に受信した場合に H_n とする.
- ID : H を TSA に送信した利用者の識別情報
- ID_n : H_n を TSA に送信した利用者の識別情報
- T_n : TSA における H_n の受信時刻
- L_n : n 番目のリンク情報
- TS_n : H_n に対するタイムスタンプ

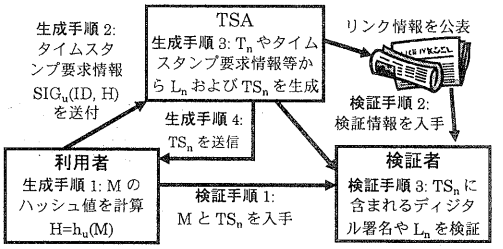

図 1. Linking Protocol のタイムスタンプ生成・検証

2.1.2.2 生成フェーズ

生成手順 1：利用者は，タイムスタンプ対象データ M のハッシュ値 H (= h_u(M))を生成.

生成手順 2：利用者は，タイムスタンプ要求情報として SIG_u(ID, H)を TSA へ送付.

生成手順 3：TSA は，L_{n-1} や時刻情報 T_n 等を用いて L_n や TS_n を生成（リンク情報の一部を新聞等に掲載する方式もある）.

生成手順 4：TSA は，利用者に TS_n を送付.

2.1.2.3 検証フェーズ

検証手順 1：検証者は，データ M とタイムスタンプ TS_n を利用者から入手.

検証手順 2：検証者は，TSA や新聞等からタイムスタンプの検証に必要な情報を入手.

検証手順 3：検証者は，TS_n に含まれる TSA のディジタル署名や L_n を検証.

2.2 その他のプロトコル

Simple Protocol は，1 つの TSA がタイムスタンプ対象データのハッシュ値に時刻情報等を加え，これらに対するディジタル署名を生成してタイムスタンプとする方法である．同プロトコルは，安全性をディジタル署名に大きく依存しており，ディジタル署名の安全性に問題が生じた場合，タイムスタンプの改ざんが容易であることは自明であることから，今回考察対象としない.

また，Distributed Protocol は，タイムスタンプを複数の TSA が共同で生成する方式である．複数の TSA が生成したディジタル署名を結合する方式や，秘密分散技術を用いる方式等がある．同プロトコルは今後の考察対象とし，本稿では取り上げない.

3. 攻撃の前提・目的

3.1 攻撃の前提

本稿では，攻撃の前提として以下の 3 点を考慮し，全体で 8 つのケースを想定した（表 1）.

- **TSA のハッシュ関数において，攻撃者が選択した特定のハッシュ値の** second-preimage（SP）**探索が容易か否か.**

ハッシュ関数 F が second-preimage 探索容

表 1. 攻撃の前提（8 つのケース）

	TSA との結託	TSA のハッシュ関数の SP 探索	TSA のディジタル署名の選択的偽造
1	不可能	困難	困難
2			容易
3		容易	困難
4			容易
5	可能	困難	困難
6			容易
7		容易	困難
8			容易

易であるとは，ある特定の入力値 x が与えられた場合，F(x)=F(y)を満たす入力値 y(≠x)を検索できる多項式時間アルゴリズムが存在し，攻撃者が利用可能であることを意味する．

- **TSA のディジタル署名において，攻撃者が選択した特定のメッセージに対する署名の偽造（選択的偽造）が容易か否か．**

本稿では，ディジタル署名の偽造が容易となる状況として，署名偽造のための多項式時間アルゴリズムが存在し，攻撃者が利用できる状況や，攻撃者が何らかの形で TSA の署名生成鍵を入手できる状況を想定する．

- **攻撃者が TSA と結託可能（TSA の一部の内部者との結託も含む）か否か．**

3.2 攻撃の目的

攻撃の目的として次の 3 点を考慮した．

- **攻撃者が所有するタイムスタンプのハッシュ値や時刻情報の改ざん**
- **他者のタイムスタンプの検証困難化**
- **特定の TSA における業務規律に関する信頼低下**

4. Cuculus の安全性に関する考察

4.1 プロトコルの概要と特徴

Cuculus は，電子文書管理やディジタル署名に関するエストニアの国家研究プロジェクトの成果として提案された．主な特徴点は以下の 2 点である．

- **リンク情報の公表**：TSA は，リンク情報を定期的に新聞に掲載し，掲載後におけるリンク情報の改ざんを困難にしている．
- **検証情報の分散**：TSA は，各タイムスタンプの検証情報を Time Certificate（TC）として各利用者に送信する．TSA のデータベースが利用不能な場合でもタイムスタンプの検証が可能となる．

4.2 タイムスタンプの生成・検証

4.2.1 生成フェーズ（図 2）

生成手順 1: 利用者は，タイムスタンプ要求情報として，TSA に $SIG_u(H)$を送付．

生成手順 2: TSA は，以下の要領で L_n と TS_n を生成し，利用者に送付．

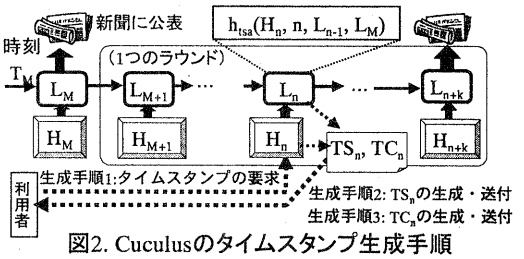

図2. Cuculusのタイムスタンプ生成手順

$$L_n = h_{tsa}(H_n, n, L_{n-1}, L_M)$$
$$TS_n = SIG_{tsa}(H_n, T_n, n, L_n)$$

L_M は直前に公表されたリンク情報であり，リンク情報は一定時間ごとに新聞に掲載される．掲載の時間的間隔はラウンドと呼ばれる．

生成手順 3: TSA は，現ラウンドにおける最後のリンク情報（L_{n+k} とする）を生成後，以下の TC_n を生成して利用者に送信．L_{n+k} は新聞に掲載．

$$TC_n = SIG_{tsa}(n, L_M, L_{n+k}, H_{M+1}, \ldots, H_{n+k})$$

4.2.2 検証フェーズ

検証手順 1: 検証者は，利用者から TS_n，TC_n を入手し，タイムスタンプ対象データのハッシュ値が TS_n と TC_n に含まれているかを検証．

検証手順 2: 検証者は，TS_n のディジタル署名を検証．

検証手順 3: 検証者は，新聞掲載の L_M と L_{n+k} が，TC_n に含まれるものと一致しているかを検証．その上で，TC_n の情報から当該ラウンドにおけるすべてのリンク情報系列 L_j (j=M+1,..., n+k)を生成．

検証手順 4: 検証者は，生成した L_{n+k} が新聞掲載のものと一致しているかを検証．

4.3 安全性に関する考察結果（表 2）

4.3.1 攻撃者が TSA と結託可能

ハッシュ関数が SP 探索容易，かつ，ディジタル署名が選択的偽造困難な場合（ケース 7, 8），攻撃者は Bypass Attack（後述）によって自分のタイムスタンプを容易に改ざんできる．また，ハッシュ関数が SP 探索容易，かつ，ディジタル署名が選択的偽造容易な場合（ケース 8），攻撃者は，自分のタイムスタンプの改ざんに加えて，他者のタイムスタンプの検証困難化や特定の TSA の信頼低下を目的とした攻撃を実行できる．

一方，ハッシュ関数が SP 探索困難な場合（ケース 5, 6），3 種類のいずれの攻撃も適用困難である．

以下では，ケース 7, 8 における Bypass Attack の適用方法を示す．

4.3.1.1 ケース 7 における Bypass Attack

Bypass Attack は，偽造もしくは改ざんしたタイムスタンプの検証を成功させる目的で，真のリンク情報及びタイムスタンプの系列とは別に，Bypass とな

る不正なデータ系列を偽造する攻撃である．以下では，データ系列の Bypass を生成し，タイムスタンプの時刻情報を改ざんする方法を示す．

・**攻撃目的**：文書 M（ハッシュ値は Z）が過去の時刻 T_n に生成されたようにみせかけるため，M に対するタイムスタンプ TS_n'を偽造する．

・**タイムスタンプの偽造**

Step 1：攻撃者は，以下の等式を満たすハッシュ値系列[H_{M+1}', ..., H_{n-1}', H_{n+1}',..., H_{n+k}'] (Bypass となるハッシュ値系列) を探索．

$$L_{n+k}=h_{tsa}(H_{n+k}', n+k, h_{tsa}(H_{n+k-1}', n+k-1,..., h_{tsa}(Z, n,..., h_{tsa}(H_{M+1}', M+1, L_M, L_M), ...L_M), L_M)$$

Step 2：TSA は上記ハッシュ値系列を用いて不正なリンク情報 L_n'を生成し，TS_n'と TC_n'を偽造．

$$L_n' = h_{tsa}(Z, n, L_{n-1}, L_M)$$

$$TS_n' = SIG_{tsa}(Z, T_n, n, L_n')$$

$$TC_n' = SIG_{tsa}(n, L_M, L_{n+k}, H_{M+1}', ..., H_{n+k}')$$

・**偽造された TS_n の検証**

Step 1: 検証者は M のハッシュ値 Z を計算し，Z が TS_n'に含まれているかを検証（→検証成功）．

Step 2: 検証者は TS_n'のディジタル署名を検証（→検証成功）．

Step 3: 検証者は，TC_n'と新聞掲載の L_M から L_j' (j = M+1, ..., n+k)を生成し，新聞掲載の L_{n+k} が L_{n+k}'と同一かを検証（→検証成功）．

4.3.1.2 ケース 8 における Bypass Attack

以下では，Bypass Attack によって他者のタイムスタンプの検証を困難にする方法を示す．

・**攻撃目的**：攻撃対象のタイムスタンプ TS_n と同時刻に同一の TSA において生成されたと考えられる別のタイムスタンプ TS_n'を偽造し，両者のどちらが正当かを検証困難にする．

・**基本方針**：Bypass Attack によって不正なリンク情報系列や Time Certificate 等を生成し，それらを同一ラウンドの他の利用者（ID_n を除く ID_{M+1} から ID_{n+k} の利用者）に配付．その上で，真の TS_n の所有者と TSA が結託していると主張．

・**追加条件**：攻撃者が，攻撃対象となるタイムスタンプが含まれるラウンドの他のすべての利用者と結託可能である．

・**不正なタイムスタンプ系列と TC の生成**

Step 1: 攻撃者は，ある H_n'の下で，以下の等式を満たすハッシュ値系列 H_{n+i}'(i = 1,..., k)を探索．

$$L_{n+k}=h_{tsa}(H_{n+k}', n+k, h_{tsa}(H_{n+k-1}', n+k-1,..., h_{tsa}(H_n', n, L_{n-1}, L_M),..., L_M), L_M)$$

Step 2：攻撃者は，H_{n+j}' (j = 0,..., k)を用いて L_{n+j}', TS_{n+j}', TC_{n+j}'を以下の要領で偽造．

$$L_{n+j}' = h_{tsa}(H_{n+j}', n+j, L_{n+j-1}, L_M)$$

表 2. Cuculus の安全性に関する考察結果

	H_n や T_n の改ざん	他者の TS の検証困難化	特定の TSA の信頼低下
1	適用困難	適用困難	適用困難
2			
3			
4	Bypass Attack	「同一ラウンドにおける他のすべての利用者と結託可能」との条件が必要	
5	適用困難		
6			
7	Bypass Attack		
8		Bypass Attack	Bypass Attack

$$TS_{n+j}' = SIG_{tsa}(H_{n+j}', T_{n+j}, n+j, L_{n+j}')$$

$$TC_{n+j}' = SIG_{tsa}(n+j, L_M, L_{n+k}, H_{M+1},..., H_{n-1}, H_n',..., H_{n+k}')$$

Step 3：攻撃者は，ID_j (j = M+1, ..., n-1)の利用者にはそれぞれ TC_j'を送信し，ID_m (m = n+1, ..., n+k)の利用者にはそれぞれ[TS_m', TC_m']を送信．

・**TS_n の検証**

Step 1: 検証者は，TC_n を用いて TS_n を検証し，TS_n の検証が成功することを確認．

Step 2: 攻撃者が，TS_n と同一の TSA で同一時刻に生成されたとする TS_n'を示し，TC_n'による TS_n'の検証を検証者に要求（→検証成功）．

Step 3: 検証者は，同一ラウンドの他の利用者が攻撃者と同一の TC_n'を所有していることを確認．また，TSA が TC_n と同一の情報を保管していることも確認．TSA は攻撃者と結託可能性を有し信頼できないため，検証者は TS_n の正当性を確認困難．

4.3.2 攻撃者が TSA と結託不可能（ケース 1～4）

ハッシュ関数が SP 探索容易，かつ，ディジタル署名が選択的偽造容易な場合（ケース 4），攻撃者は Bypass Attack によって自分のタイムスタンプを容易に改ざんできる．しかし，他のケース（ケース 1～3）では，いずれの攻撃も事後的に検知可能なため適用困難である．

5. PKITS の安全性に関する考察

5.1 プロトコルの概要と特徴

PKITS（Public Key Infrastructure with Time-Stamping Authority）は欧州委員会の下で実施された研究プロジェクトの成果として提案されたものである．主な特徴は以下の 2 点である．

・**リンク情報の相互交信**：複数の TSA が存在する環境下で，各 TSA はリンク情報を適宜他の TSA に送付し，リンク情報のタイムスタンプ生成を依頼．他の TSA によるタイムスタンプによってリンク情報の真正性を確保可能．

・**TSA による検証情報の一括管理**：TSA は，各タ

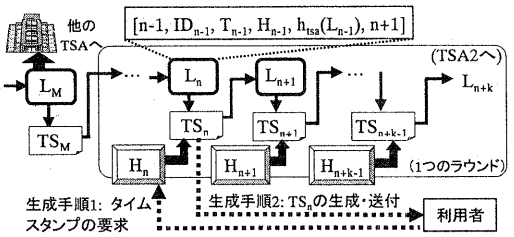

図3. PKITSにおけるタイムスタンプ生成手順

イムスタンプの検証情報を一括管理し，必要に応じて検証者に送信．

5.2 タイムスタンプの生成・検証

5.2.1 生成フェーズ

生成手順 1: 利用者は，タイムスタンプ要求情報として $SIG_u(H, ID)$を TSA（TSA1 と呼ぶ）に送付．

生成手順 2: TSA1 は，L_n と TS_n を生成してデータベースに保管した後，利用者に送付．

$L_n = [n-1, ID_{n-1}, T_{n-1}, H_{n-1}, h_{tsa}(L_{n-1}), n+1]$

$TS_n = SIG_{tsa}(n, ID_n, T_n, H_n, L_n)$

生成手順 3: TSA1 は，時刻 T_{n+k} に，L_{n+k} のハッシュ値 $h_{tsa}(L_{n+k})$を別の TSA（TSA2）に送付し，タイムスタンプ TS*を得る．

5.2.2 検証フェーズ

検証手順 1: 検証者は，タイムスタンプ対象データのハッシュ値を計算し，そのハッシュ値が TS_n に含まれているかを検証．

検証手順 2: 検証者は TS_n のディジタル署名を検証．

検証手順 3: 検証者は，TSA1 から TS_i (i=n+1,..., n+k-1)と TS*を入手し，各署名を検証．

検証手順 4: 検証者は，TS_i(i=n+1,..., n+k-1)からリンク情報系列 L_i(i=n+1,..., n+k)を生成し，その L_{n+k} が TS*の対象となっているかを確認．

5.3 安全性に関する考察結果（表 3）

5.3.1 攻撃者が TSA と結託可能

ハッシュ関数が SP 探索容易な場合（ケース 7, 8），Cuculus と同様，攻撃者は Bypass Attack を適用して自分のタイムスタンプを容易に改ざんできる．特に，ディジタル署名が選択的偽造容易な場合（ケース 8），攻撃者は，Bypass Attack によって他者のタイムスタンプの検証を困難化できるほか，特定の TSA の信頼を低下させることができる．

一方，ハッシュ関数が SP 探索困難な場合（ケース 5, 6），攻撃者は，Fake Chain Attack（後述）により，自分のタイムスタンプを容易に改ざんできる．

以下では，ケース 5, 6 における，Fake Chain Attack を利用したタイムスタンプの改ざん方法を示す．

Fake Chain Attack は，真のタイムスタンプ系列の特定時点から枝別れした別のタイムスタンプ系列

表 3. PKITS の安全性に関する考察結果

	H_n や T_n の改ざん	他者の TS の検証困難化	特定の TSA の信頼低下
1	適用困難	適用困難	適用困難
2			
3			
4		「同一ラウンドにおける他のすべての利用者と結託可能」との条件が必要	
5	Fake Chain Attack		
6			
7	Bypass Attack		
8		Bypass Attack	Bypass Attack

を偽造し，タイムスタンプの改ざん等の不正行為を行うものである[5]．以下では，Fake Chain Attack によってタイムスタンプを改ざんする方法を示す．

- **攻撃目的**：攻撃者（ID_n）は，H_n を H_n'に改ざんし，H_n に対する既存のタイムスタンプ TS_n を，H_n'に対するタイムスタンプ TS_n'に改ざんする．
- **前提**：TSA1 は，T_n 以降最初にタイムスタンプ要求情報として L_{n+k} を TSA2 に送信する．
- **不正なタイムスタンプ系列（Fake Chain）の生成**

Step 1：攻撃者はハッシュ値 H_n'を TSA1 に送付．

Step 2：TSA1 は，H_n'と正当なタイムスタンプ系列 TS_{n+i} (i = 1,...k-1)から，不正なリンク情報系列 L_{n+j}' (j=1,..., k)を以下の要領で偽造．

j = 1 の場合：$L_{n+1}' = [n, ID_n, T_n, H_n', h_{tsa1}(L_n), n+2]$

j ≠ 1 の場合：$L_{n+j}' = [n+j-1, ID_{n+j-1}, T_{n+j-1}, H_{n+j-1}, h_{tsa1}(L_{n+j-1}'), n+j+1]$

Step 3：TSA1 は，不正なタイムスタンプの系列 TS_{n+m}' (m = 0, ..., k-1)を以下の要領で偽造．

m=0 の場合：$TS_n' = SIG_{tsa1}(n, ID_n, T_n, H_n', L_n)$

m≠0 の場合：$TS_{n+m}' = SIG_{tsa1}(n+m, ID_{n+m}, T_{n+m}, H_{n+m}, L_{n+m}')$

Step 4：TSA1 は，TS_n'を攻撃者に送信するほか，L_{n+k}'のハッシュ値 $h_{tsa1}(L_{n+k}')$をタイムスタンプ要求情報として TSA2 に送信．

Step 5：TSA2 は $h_{tsa1}(L_{n+k}')$に対して時刻 T_{n+k} のタイムスタンプ TS*を偽造し，TSA1 に返信．

- **TS_n'の検証**

Step 1: 検証者は，H_n'が TS_n'に含まれているハッシュ値と同一か否かを検証（→検証成功）．

Step 2: 検証者は TS_n'のディジタル署名を検証（→検証成功）．

Step 3: 検証者は検証情報の送付を TSA1 に要求．TSA1 は TS_{n+m}' (m=0,..., k-1)と TS*を送付．

Step 4: 検証者は TS_{n+m}' (m=0,..., k-1)と TS*のディジタル署名を検証（→検証成功）．

Step 5: 検証者は，TS_{n+m}'(m=0,..., k-1)から L_{n+j}' (j=1,..., k)を生成し，L_{n+k}'のハッシュ値が TS*に

含まれるものと同一かを検証（→検証成功）.

なお，[6]は，PKITS にはリンク情報の相互交信によって別の検証パスが存在することを示しているが，そのような検証パスを利用する場合でも，Fake Chain Attack や Bypass Attack は適用可能となる.

5.3.2 攻撃者が TSA と結託不可能

攻撃者が TSA と結託不可能な場合（ケース 1～4），検証者はいつでも検証に必要な真の情報を TSA から入手可能なため，3 種類のいずれの攻撃も適用困難である.

6. 2 つの考察結果の比較

以上の考察より，プロトコルの安全性は Cuculus と PKITS で異なることを明らかにした（表 4）.

こうした差異は，2 つのプロトコルにおける検証情報及びリンク情報の管理方法の差に起因する. PKITS では，TSA が検証情報を一括管理すると同時に，他の TSA のタイムスタンプによってリンク情報の真正性を確保する. このため，攻撃者が TSA と結託可能な場合，TSA は，検証情報だけでなく，他の TSA と結託してリンク情報のタイムスタンプも改ざんできる. この結果，暗号技術の安全性が低下しなくてもプロトコルの安全性が損われる可能性が生まれる. ただし，攻撃者が TSA と結託不可能な場合，TSA が検証情報やリンク情報を適切に管理可能であり，攻撃の適用が困難となる.

一方，Cuculus は，検証情報の分散化とリンク情報の新聞掲載によって，TSA による検証情報等の改ざんを困難にしている. この結果，攻撃者が TSA と結託可能でも，ハッシュ関数の安全性が損われない限りタイムスタンプの改ざんは困難となる. ただし，ハッシュ関数とディジタル署名の安全性が低下すると，TSA の結託可能性如何によらず，利用者が自分の検証情報やタイムスタンプを容易に改ざんできることになる.

7. おわりに

本稿では，利用されている暗号技術の安全性低下が Cuculus と PKITS の安全性に及ぼす影響について考察した. この結果，Cuculus では，攻撃者が TSA と結託容易，かつ，ハッシュ関数が SP 探索容易な場合，自分のタイムスタンプを容易に改ざんできることを示した. また，PKITS では，攻撃者が TSA と結託可能な場合，ディジタル署名やハッシュ関数の安全性如何によらず，タイムスタンプを容易に改ざんできることを示した. さらに，タイムスタンプの関連情報の管理方法がプロトコルの安全性を左右する要因であることも示した.

表 4. Cuculus と PKITS の考察結果の比較

	TSA と結託不可能な場合	TSA と結託可能な場合
Cuculus	ハッシュ関数とディジタル署名の両方の安全性低下： → **タイムスタンプの改ざん可能**	ハッシュ関数の安全性低下： → **タイムスタンプの改ざん可能**
PKITS	ハッシュ関数とディジタル署名の両方の安全性低下： → **タイムスタンプの改ざん困難**	暗号技術に安全性に問題なし： → **タイムスタンプの改ざん可能**

今後は，他の種類の Linking Protocol や Distributed Protocol を対象に本稿と同様の考察を行い，各プロトコルの安全性と暗号技術の安全性との関連を検討する方針である. さらに，それらの結果を基にして，暗号技術の安全性が低下してもプロトコル全体の安全性を確保するための方策について検討を進める方針である.

参考文献

[1] Buldas, A., P. Laud, H. Lipmaa and J. Villemson, "Time-Stamping with Binary Linking Schemes," Proceedings of CRYPTO'98, LNCS 1462, pp.486-501, 1999.

[2] Cybernetica, Time-Stamping Client, Version 1.1, User Manual, 2000. (http://www.cyber.ee/research/)

[3] Fabrica Nacional de Moneda y Timbre, PKITS: Deliverable D4 Time-Stamping Service Functional Specification and Protocols for Unstructured Data, Revision Number: 16, July 30, 1998. (http://www.fnmt.es/pkits/)

[4]藤永卓人，木村成伴，海老原義彦，"秘密鍵の漏洩を考慮した SSL 通信プロトコルの提案"，1999 年コンピュータセキュリティシンポジウム予稿集，pp.225-230，1999 年 10 月.

[5] Just, M., "Some Timestamping Protocol Failure," Proceedings of the Internet Society Symposium on Network and Distribute System Security, 1998.

[6]金谷篤郎，今井秀樹，"複数の検証パスを持つディジタルタイムスタンプ"，電子情報通信学会技術研究報告 ISEC2000-55，pp.235-242，2000 年 7 月.

[7] Massias, H. and J.-J. Quisquater, "Time and Cryptography," TIMESEC Technical Report, 1997.

[8]宇根正志，松浦幹太，田倉昭，"最近のディジタルタイムスタンプ技術の現状と課題"，金融研究第 19 巻別冊第 1 号，pp.105-154，2000 年 4 月.（http://www.imes.boj.or.jp/）

[9] Wen, W. and F. Mizoguchi, "Security of Authentication Protocols with Compromised Certificates," Proceedings of JW-ISC 2000, pp.135-150, 2000.