タイムリリース暗号によるタイムスタンプ署名

白勢 政明 †　　吉田 洸輝 †

† 公立はこだて未来大学
041-8655 北海道函館市亀田中野町 116-2
{shirase,b1008125}@fun.ac.jp

あらまし タイムリリース暗号では，受信者の公開鍵 upk と送信者が指定する復号時刻 T の情報を暗号化鍵として平文を暗号化し，受信者の秘密鍵 usk と時刻鍵 s_T を復号鍵として暗号文を復号する．なお，時刻鍵は時刻 T に時刻サーバから発行される．本稿は，タイムリリース暗号に基づき，時刻情報による署名検証で「受理」となることが，署名生成時刻が T であることを保証するタイムスタンプ署名を提案する．本方式はタイムリリース暗号の署名版である．提案方式におけるタイムスタンプ・サーバは，タイムリリース暗号と同じく時刻鍵 s_T を生成する機能があり，s_T を使ってタイムスタンプ生成を行う．

Timestamp Signature from Timed-Release Encryption

Masaaki Shirase†　　Koki Yoshida‡

†Future University Hakodate
116-2 Kamedanakano, Hakodate, Hokkaido, 041-8655, JAPAN
{shirase,b1008125}@fun.ac.jp

Abstract In a timed-release encryption, a sender encrypts a plane text using a a receiver's public key and decryption time T, and the receiver decrypts a cipher text using a secret key and a time key, where the time key is issued by a time server at T. This paper proposes a timestamp signature system based on timed-release encryption, in which that verification of a timestamp signature using by T and a public key is accepted guarantees that the timestamp signature was generated at T.

1　はじめに

ペアリングと呼ばれる有限群上の双線型写像 $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ を用いることで，ID ベース暗号 [12, 4]，ID ベース署名 [9]，タイムリリース暗号 [7]，リング署名 [14]，墨塗り署名 [5]，証明書不要な公開鍵暗号 [1]，キーワード検索暗号 [3]，放送型暗号 [6]，属性ベース暗号 [13]，といった様々な暗号プロトコルを効率的に実現できるため，ペアリング暗号の研究が盛んになっている．本稿の目的は，ペアリングを用いるタイムリリース暗号の技術を使って，新しいタイムスタンプ署名方式を提案することである．

タイムリリース暗号とは，復号時刻を指定できる暗号方式であり，タイムリリース暗号では，時刻 T に時刻鍵 s_T を発行する機能を持つ時刻サーバを用いる．暗号生成者は受信者の公開鍵と復号時刻 T のストリング $\mathtt{T}$ を暗号化鍵として平文を暗号化し，受信者は自身の秘密鍵と時刻鍵 s_T を使って暗号文を復号する．タイムリリース暗号を実現する手法の一つに ID ベース暗号の応用がある．この場合，タイムリリース暗号における時刻サーバは ID ベース暗号における鍵発行サーバと同じ機能を有し．タイムリリース暗号における (時刻ストリング $\mathtt{T}$, 時刻鍵 s_T) の関係は ID ベース暗号における (ユーザの ID, ユーザの秘密鍵) の関係と同じである．

タイムスタンプは，文書がタイムスタンプ生成時に存在していたこと (存在性) と文書の状態がその時から変わっていないこと (完全性) を証明する技術であり，タイムスタンプはタイムスタンプ・サーバによって生成される．[8] に書かれているように，

タイムスタンプは電子入札，電子申請，オンライントレード，電子カルテ，特許，メールシステム等に有用である．タイムスタンプの種類にはシンプル・プロトコルとリンキング・プロトコルがある．シンプル・プロトコルは電子署名技術を用い，リンキング・プロトコルでは新たなタイムスタンプは以前のタイムスタンプ全てに依存するようにハッシュ関数を用いて構成される．本稿で提案するタイムスタンプはシンプル・プロトコルに属する．

シンプル・プロトコルのタイムスタンプ・システムでは，タイムスタンプ・サーバは公開鍵を公開し秘密鍵を秘密に保持しており，ユーザの文書 m のハッシュ値または署名値から秘密鍵を使ってタイムスタンプを生成する．(署名値からタイムスタンプが生成され，ユーザが署名とタイムスタンプを受信者に送信する方式はタイムスタンプ署名と呼ばれる．) タイムスタンプの検証はタイムスタンプ・サーバの公開鍵を使ってなされる．

これに対して，本稿で提案するタイムスタンプ署名では，タイムスタンプ・サーバはタイムリリース暗号の時刻サーバと同じ機能，つまり時刻 T になると時刻鍵 s_T を生成できる機能を持ち，時刻 T のタイムスタンプとして s_T を使って文書の署名を生成する．そして，時刻 T を表すストリング T を鍵として署名検証をパスすることが，時刻 T に文書が生成されたことの保証となる．従って，検証時にタイムスタンプ・サーバの公開鍵が不要である．

本稿で用いる表記

本稿の以降では以下の表記を用いる．

TSS	: タイムスタンプ・サーバ
ID	: ユーザの ID を表すストリング
T	: 時刻 T の情報を表すストリング

2 準備

2.1 ペアリング

$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ を位数が素数 l の群とする．慣例的に $\mathbb{G}_1, \mathbb{G}_2$ は加群，$\mathbb{G}_3$ を乗法群とする．写像

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$$

が 双線型性 (すべての $P \in \mathbb{G}_1, Q_1, Q_2 \in \mathbb{G}_2$ に対して $e(P, Q_1 + Q_2) = e(P, Q_1) \cdot e(P, Q_2)$，及びすべての $P_1, P_2 \in \mathbb{G}_1, Q \in \mathbb{G}_2$ に対して $e(P_1 + P_2, Q) = e(P_1, Q) \cdot e(P_2, Q)$ が成り立つ)，非退化性 ($e(P, Q) \neq 1$ となる $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ が存在する)，計算可能性 (すべての $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ に対して $e(P, Q)$ を効率的に計算できる) を持つ時，e をペアリングという．なお，$\mathbb{G}_1 = \mathbb{G}_2$ のとき (あるいは計算が効率的な同型写像 $\mathbb{G}_1 \to \mathbb{G}_2$ が存在するとき)e を対称ペアリング，そうでないときは e を非対称ペアリングと呼ぶ．

ペアリングの具体例として，$\mathbb{G}_1$, $\mathbb{G}_2$ として有限体上の楕円曲線のなす群，$\mathbb{G}_3$ として有限体の乗法群を用いる Weil ペアリング，Tate ペアリング，η_T ペアリング [2]，Ate ペアリング [10] 等がある．

2.2 公開鍵暗号と電子署名

本節では，本稿で用いるアルゴリズムの表記の紹介を兼ねて，公開鍵暗号と電子署名を簡単に説明する．

公開鍵暗号は 3 つのアルゴリズム PKE.KeyGen, PKE.Enc，PKE.Dec から構成される．

$(upk, usk) \leftarrow$PKE.KeyGen(1^k): セキュリティパラメータ 1^k から鍵ペア (upk, usk) を生成.
$c \leftarrow$PKE.Enc(upk, m): 公開鍵 upk と平文 m から暗号文 c を生成.
$m \leftarrow$PKE.Dec(usk, c): 秘密鍵 usk と暗号文 c から復号文 m を生成.

ここで，PKE.Enc と PKE.Dec は

$$m = \text{PKE.Dec}(usk, \text{PKE.Enc}(upk, m))$$

を満たす.

また，電子署名も 3 つのアルゴリズム DS.KeyGen, DS.Sig，DS.Ver から構成される．

$(upk, usk) \leftarrow$DS.KeyGen(1^k): セキュリティパラメータ 1^k から鍵ペア (upk, usk) を生成.
$\sigma \leftarrow$DS.Sig(usk, m): 秘密鍵 usk と文書 m から署名 σ を生成.
受理 or 拒否 $\leftarrow$ DS.Ver(upk, m, σ): 公開鍵 upk，文書 m，署名 σ から文書の受理/拒否を判定.

ここで，DS.Sig と DS.Ver は

$$\text{受理} = \text{DS.Ver}(upk, m, \text{DS.Sig}(usk, m))$$

を満たす.

本稿では，DS.Sig が文書 m のハッシュ値 $h(m)$ からサブアルゴリズム DS.SigSub を呼び出して

$$\sigma \leftarrow \text{DS.SigSub}(usk, h(m))$$

により署名 σ を生成し，DS.Ver もサブアルゴリズム DS.VerSub を使って

$$\text{DS.VerSub}(upk, \sigma) = h(m)$$

が成り立つ時受理，そうでない時拒否を出力するような電子署名をタイプ I，そうでないものをタイプ II と呼ぶことにする．タイプ I の電子署名では，文書 m' が適切な集合の元でサイズが十分に小さいならば

$$m' = \text{DS.VerSub}(upk, \text{DS.SigSub}(usk, m'))$$

が成り立つ．タイプ I の電子署名に RSA 署名がある．ElGamal 署名や Schnorr 署名はタイプ II である．

2.3 ID ベース暗号

ID ベース暗号とは，ユーザの ID を公開鍵とできる公開鍵暗号方式のことであり，(対称) ペアリングを用いることで効率的に ID ベース暗号を実現できる．ここでは，[4] の手法を紹介する．ID ベース暗号は 4 つのアルゴリズム，IBE.Setup，IBE.Extract，IBE.Enc，IBE.Dec から構成される．$\{0,1\}^n$ を平文空間とする．なお，対称ペアリング $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_3$ ($\mathbb{G}_1$ と $\mathbb{G}_3$ の位数は素数 l) と，2 つのハッシュ関数 $H : \mathbb{G}_1 \to \{0,1\}^n$，$G : \{0,1\}^* \to \mathbb{G}_1$ を要し，これらに関する情報は公開パラメータ prm として公開される．

$(prm, msk) \leftarrow$IBE.Setup(1^k): セキュリティパラメータ 1^k から公開パラメータ prm やマスター鍵 msk を生成．

1. $P \in \mathbb{G}_1$ を選択．P を prm に追加．
2. $msk \in \{1, 2, \cdots, l-1\}$ を選び，$P_{pub} = mskP \in \mathbb{G}_1$ を計算．P_{pub} を prm に追加．

$d_{ID} \leftarrow$IBE.Extract$(msk, \mathtt{ID})$: ユーザの ID とマスター鍵 msk からユーザの秘密鍵 d_{ID} を生成．

1. $Q_{ID} = G(\mathtt{ID}) \in \mathbb{G}_1$ を計算．
2. $d_{ID} = mskQ_{ID} \in \mathbb{G}_1$ を計算．

$c \leftarrow$IBE.Enc$(\mathtt{ID}, m)$: ユーザの ID と平文 m から暗号文 c を生成．

1. $Q_{ID} = G(\mathtt{ID}) \in \mathbb{G}_1$ を計算．
2. 乱数 $r \in \{1, 2, \cdots, l-1\}$ を生成．
3. 暗号文
   $c \leftarrow (rP, m \oplus H(e(Q_{ID}, P_{pub}))) \in \mathbb{G}_1 \times \{0,1\}^*$
   を計算．

$m \leftarrow$IBE.Dec(d_{ID}, c): 秘密鍵 d_{ID} と暗号文 $c = (u, v)$ から復号文 m を生成．

1. $m = v \oplus H(e(d_{ID}, u))$ を計算．

ここで，IBE.Enc と IBE.Dec は

$$m = \text{IBE.Dec}(d_{ID}, \text{IBE.Enc}(\mathtt{ID}, m))$$

を満たす．

このID ベース暗号システムで重要なことは，マスター鍵 msk を秘密に保持しユーザの ID から秘密鍵 d_{ID} を生成する鍵発行サーバが必要なことである．

2.4 タイムリリース暗号

複号時刻を指定できる公開鍵暗号方式をタイムリリース暗号という．タイムリリース暗号は5つのアルゴリズム TRE.Setup，TRE.KeyGen，TRE.Release，TRE.Enc，TRE.Dec から構成される．ここでは，[11] による一般的構成法，つまり既存の ID ベース暗号 (IBE.Setup, IBE.Extract, IBE.Enc, IBE.Dec) と公開鍵暗号 (PKE.KeyGen, PKE.Enc, PKE.Dec) を利用して構成する方法を紹介する．但し，ここでは簡単のため，適応的選択暗号文攻撃に対する安全性を考慮しないものを考える．適応的選択暗号文攻撃に対する安全性を考慮した方式は [11] を参照．

$(prm, msk) \leftarrow$TRE.Setup(1^k): セキュリティパラメータ 1^k から公開パラメータ prm と時刻サーバのマスター鍵 msk を生成．

1. $(prm, msk) \leftarrow$IBE.Setup(1^k) を計算．

$(upk, usk) \leftarrow$TRE.KeyGen(1^k): セキュリティパラメータ 1^k からユーザの鍵ペア (upk, usk) を生成．

1. $(upk, usk) \leftarrow$PKE.KeyGen(1^k) を計算．

$s_T \leftarrow$TRE.Release$(msk, \mathtt{T})$: マスター鍵 msk と時刻情報 T から時刻 T に対応する時刻鍵 s_T を生成．

1. $s_T \leftarrow$IBE.Extract$(msk, \mathtt{T})$ を計算．

$c \leftarrow$TRE.Enc$(upk, \mathtt{T}, m)$: 公開鍵 upk，時刻情報 T，平文 m から暗号文 c を生成．

1. $c' \leftarrow$IBE.Enc$(\mathtt{T}, m)$ を計算．
2. $c \leftarrow$PKE.Enc(upk, c') を計算．

$m \leftarrow$TRE.Dec(usk, s_T, c): 秘密鍵 usk，時刻鍵 s_T，暗号文 c から平文 m を出力．

1. $c' \leftarrow$PKE.Dec(usk, c) を計算．
2. $m \leftarrow$IBE.Dec(s_T, c') を計算．

ここで，TRE.Enc と TRE.Dec は

$$m = \text{TRE.Dec}(usk, s_T, \text{TRE.Enc}(upk, \mathtt{T}, m))$$

を満たす．

時刻サーバはマスター鍵 msk を秘密に保持し，時刻 T になった時に時刻鍵 s_T を生成し公開する．従ってそれ以前では，T で暗号化された暗号文は復号できない．

2.5 ID ベース署名

署名検証の時に公開鍵としてユーザの ID を用いる ID ベース署名はいくつか提案されているが，ここでは [9] の方法を紹介する．この ID ベース署名は，4 つのアルゴリズム IBS.Setup，IBS.Extract，IBS.Sig，IBS.Ver から構成される．なお，この ID ベース署名では，対称ペアリング $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_3$ ($\mathbb{G}_1$ と $\mathbb{G}_3$ の位数は素数 l) と 2 つのハッシュ関数 $h : \{0,1\}^* \times \mathbb{G}_3 \to (\mathbb{Z}/l\mathbb{Z})^*$，$H : \{0,1\}^* \to \mathbb{G}_1 \setminus \{0\}$ を要し，これらに関する情報は公開パラメータ prm として公開される．

$(prm, msk) \leftarrow$IBS.Setup(1^k): セキュリティパラメータ 1^k から公開パラメータ prm やマスター鍵 msk を生成．

1. $P \in \mathbb{G}_1$ を選択．P を prm に追加．
2. $msk \in \{1, 2, \cdots, l-1\}$ を選び，$P_{pub} = mskP \in \mathbb{G}_1$ を計算．P_{pub} を prm に追加．

$d_{ID} \leftarrow$IBS.Extract(msk, ID): ユーザの ID から秘密鍵 d_{ID} を生成．

1. $Q_{ID} = H(\mathtt{ID}) \in \mathbb{G}_1$ を計算．
2. $d_{ID} = mskQ_{ID} \in \mathbb{G}_1$ を計算．

$\sigma \leftarrow$IBS.Sig(d_{ID}, m): 秘密鍵 usk と文書 m から署名 σ を生成．

1. $P_1 \in \mathbb{G}_1$ を選択．
2. 乱数 $r \in \{1, 2, \cdots, l-1\}$ を生成．
3. $r' = e(P_1, P)^r \in \mathbb{G}_3$ を計算．
4. $v = h(m, r') \in (\mathbb{Z}/l\mathbb{Z})^*$ を計算．
5. $u = vS_{ID} + kP_1 \in \mathbb{G}_1$ を計算．
6. $\sigma = (u, v) \in \mathbb{G}_1 \times (\mathbb{Z}/l\mathbb{Z})^*$ とする．

受理 or 拒否 $\leftarrow$IBS.Ver(ID, m, σ): 送信者の ID と文書 m，署名 $\sigma = (u, v)$ から受理/拒否を判定．

1. $r = e(u, P) \cdot e(G(\mathtt{ID}), -Q_{TA})^v \in \mathbb{G}_3$ を計算．
2. $v = h(m, r)$ の時受理，そうでない時は拒否．

ここで，IBS.Sig と IBS.Ver は

$$\text{受理} = \text{IBS.Ver}(\mathtt{ID}, m, \text{IBS.Sig}(d_{ID}, m))$$

を満たす．

ID ベース署名では ID ベース暗号と同様に，マスター鍵 msk を保持しユーザの ID から秘密鍵 d_{ID} を生成する鍵発行サーバが必要である．

2.6 タイムスタンプ署名

ここでは，シンプル・プロトコルと呼ばれる，電子署名を用いるタイムスタンプ署名を紹介する．もう一つのタイプであるリンキング・プロトコルは [8] を参照．

2 つの電子署名 DS1=(DS1.KeyGen, DS1.Sig, DS1.Ver) と DS2=(DS2.KeyGen, DS2.Sig, DS2.Ver) を用いる．

鍵生成:

1. $(upk, usk) \leftarrow$DS1.KeyGen(1^k) により受信者の鍵ペアを生成．
2. $(tpk, tsk) \leftarrow$ DS2.KeyGen(1^k) により TSS の鍵ペアを生成．

タイムスタンプ署名の生成:

1. 送信者は自身の秘密鍵 usk を使って文書 m の署名 $\sigma_U \leftarrow$DS1.Sig(usk, m) を計算．
2. 送信者は σ_U を TSS に送信．
3. TSS が σ_U を受信した時刻が T の場合，TSS は $\sigma_T \leftarrow$ DS2.Sig($tsk, \sigma_U \circ \mathtt{T}$) を計算．
4. TSS は署名 σ_U のタイムスタンプとして $(\sigma_T, \mathtt{T})$ を送信者に返信[1]．
5. 送信者は $(m, \sigma_U, \sigma_T, \mathtt{T})$ をタイムスタンプ署名付き文書として受信者に送信．

タイムスタンプ署名の検証:

1. $(m, \sigma_U, \sigma_T, \mathtt{T})$ を受信．
2. 受理 =DS1.Ver(upk, m, σ_U) が成り立つかどうかチェック．
3. 受理 =DS2.Ver($tpk, \sigma_U \circ \mathtt{T}, \sigma_T$) が成り立つかどうかチェック．
4. 2, 3 の両方が受理の時文書 m を受理し，そうでない時は拒否．

従来のタイムスタンプ署名では，検証時にユーザの公開鍵 usk の他に TSS の公開鍵 tpk が必要であることに注意．また，タイムスタンプ署名付き文書のサイズは

$$|m| + |\sigma_U| + |\sigma_T| + |\mathtt{T}| \tag{1}$$

となる．

[1] 実用的なタイムスタンプ・システムでは更にタイムスタンプ・ポリシーに関する情報，時間の精度等も付加される [8].

3 提案タイムスタンプ署名

ここでは，2 つのタイプのタイムスタンプ署名法，提案手法 1，2 を提案する．提案手法 1 は，TSS にタイムリリース暗号の時刻サーバを用いる方法である．提案手法 2 は，[11] で提案されたタイムリリース暗号の署名版ともいうべきものであり，提案手法 1 と同様に TSS はタイムリリース暗号の時刻サーバの機能を有する．どちらのタイプも，TSS が時刻 T にユーザからタイムスタンプのリクエストを受信した場合，TSS はまず時刻鍵 s_T を生成し，それから s_T を使って署名を生成し，それをタイムスタンプとする．そして，タイムスタンプ署名付き文書を受信した受信者は時刻情報 T を鍵としてタイムスタンプを検証する．

3.1 提案手法 1

提案手法 1 では， ID ベース署名 (IBS.Setup, IBS.Extract, IBS.Enc, IBS.Dec) と電子署名 (DS.KeyGen, DS.Sig, DS.Ver) を用いる．なお，電子署名はタイプ I，タイプ II のどちらでも良い．(電子署名のタイプについては 2.2 節を参照．)

鍵生成

1. (upk, usk) ←DS.KeyGen(1^k) により受信者の鍵ペアを生成.
2. (prm, msk) ← IBE.KeyGen(1^k) により公開パラメータと TSS のマスター鍵 msk を生成.

タイムスタンプ署名の生成:

1. 送信者は自身の秘密鍵 usk を使って文書 m の署名 σ_U ←DS.Sig(usk, m) を計算.
2. 送信者は σ_U を TSS に送信.
3. TSS が σ_U を受信した時刻が T の場合，TSS は $s_T \leftarrow$ IBS.Extract$(msk, \mathtt{T})$ により時刻鍵 s_T を生成.
4. TSS は $\sigma_T \leftarrow$ IBS.Sig(s_T, σ_U) を計算.
5. TSS は署名 σ_U のタイムスタンプとして $(\sigma_T, \mathtt{T})$ を送信者に返信.
6. 送信者は $(m, \sigma_U, \sigma_T, \mathtt{T})$ をタイムスタンプ署名付き文書として受信者に送信.

タイムスタンプ署名の検証:

1. $(m, \sigma_U, \sigma_T, \mathtt{T})$ を受信.
2. 受理 =DS.Ver(upk, m, σ_U) が成り立つかどうかチェック.
3. 受理 =IBS.Ver$(\mathtt{T}, \sigma_U, \sigma_T)$ が成り立つかどうかチェック.
4. 2, 3 の両方が受理の時文書 m を受理し，そうでない時は拒否.

提案手法 1 のタイムスタンプ署名付き文書のサイズは

$$|m| + |\sigma_U| + |\sigma_T| + |\mathtt{T}| \tag{2}$$

となる．

3.2 提案手法 2

提案タイムスタンプ署名の構成には，ID ベース署名 (IBS.Setup，IBS.Extract，IBS.Sig，IBS.Ver) とタイプ I の電子署名 (DS.KeyGen，DS.Sig，DS.Ver (及びサブアルゴリズム DS.SigSub，DS.VerSub)) を用いる．

鍵生成

1. (upk, usk) ←DS.KeyGen(1^k) により受信者の鍵ペアを生成.
2. (prm, msk) ← IBE.KeyGen(1^k) により公開パラメータと TSS のマスター鍵 msk を生成.

タイムスタンプ署名の生成

1. 送信者は文書 m のハッシュ値 $h(m)$ を TSS に送信.
2. TSS が $h(m)$ を受信した時刻が T の場合，TSS は時刻鍵 $s_T \leftarrow$ IBS.Extract$(msk, \mathtt{T})$ を計算し，それから σ_T ←IBS.Sig$(s_T, h(m))$ を計算.
3. TSS は $(\sigma_T, \mathtt{T})$ をタイムスタンプとして送信者に返信.
4. 送信者は自身の秘密鍵 usk を使って σ_T の署名 σ_U ←DS.Sig(usk, σ_T)(=DS.SigSub(usk, σ_T)) を計算.
5. 送信者は $(m, \sigma_U, \mathtt{T})$ をタイムスタンプ署名付き文書として受信者に送信.

タイムスタンプ署名の検証:

1. $(m, \sigma_T, \mathtt{T})$ を受信.
2. $\sigma' \leftarrow$ DS.VerSub(upk, σ_T) を計算.
3. 受理 =IBS.Ver$(\mathtt{T}, m, \sigma')$ が成り立つならば文書 m を受理し，そうでなければ拒否.

タイムスタンプ署名付き文書のサイズは

$$|m| + |\sigma_U| + |\mathtt{T}| \tag{3}$$

となる．

3.3 提案手法と既存手法との比較

式 (1) と式 (2) より従来のタイムスタンプ署名と提案手法 1 では，タイムスタンプ署名のサイズは同じである．しかしながら提案手法 1 では，従来のものと違い，検証時に TSS の公開鍵を必要としないメリットがある．

提案手法 2 では，ユーザの署名生成にタイプ I の電子署名を用いなければならないので，式 (2) と式 (3) からでは 2 つの提案手法におけるタイムスタンプ署名のサイズを簡単には比較できない．提案手法 2 も提案手法 1 と同様に検証時に TSS の公開鍵を必要としないメリットがある．

3.4 時刻鍵保存問題

提案手法において，TSS は定期的に時刻鍵 s_T を生成し公開する場合を考える．すると，各ユーザはタイムスタンプ署名を生成する時，文書の署名 (または文書のハッシュ値) を TSS に送信することなくタイムスタンプ署名を生成できる．しかしながら，ユーザが古い時刻鍵 $s_{T.old}$ を保存していると，$s_{T.old}$ を使って古い時刻 $T.old$ に文書が存在していたことを示すタイムスタンプ署名を生成できてしまう．従って，この問題 (時刻鍵保存問題) のために時刻鍵を公開することはできない．

しかしながら，何らかの方法で時刻鍵保存問題を解決できるならば，TSS は定期的に時刻鍵を生成し公開するだけでよくなり，TSS の負荷が非常に軽いタイムスタンプ署名が実現できそうである．

4 まとめと今後の課題

本稿はタイムリリース暗号における時刻サーバを TSS とすることで，タイムスタンプの検証に時刻 T を公開鍵として用いるタイムスタンプ署名を提案した．この提案手法では，従来のタイムスタンプ署名とは異なり，検証時に TSS の公開鍵を必要としない．今後は提案タイムスタンプ署名の実装を行い，計算時間やメモリ消費等の計算コストを評価したい．

また，時刻鍵保存問題を解決できるならば，提案手法を使って非常に効率的なタイムスタンプ署名システムを構築できるかもしれない．従って，時刻鍵保存問題の解決法を探っていきたい．

謝辞

本研究は科研費 (若手 B21700083) の助成を受けて行われた．

参考文献

[1] S. Al-Riyami, K. Peterson, “Certificateless public key cryptography,” *ASIACRYPT 2003*, LNCS 2894, pp. 452-474. Springer, 2003.

[2] P. Barreto, S. Galbraith, C. Ó hÉigeartaigh, and M. Scott, “Efficient pairing computation on supersingular abelian varieties,” *Designs, Codes and Cryptography*, Springer–Verlag, Vol. 42, No. 3, pp 239-271, 2007.

[3] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano, “Public key encryption with keyword search,” *EUROCRYPT 2004*, LNCS 3027, pp.506-522, 2004.

[4] D. Boneh and M. Franklin, “Identity based encryption from the Weil pairing,” *SIAM Journal of Computing*, Vol. 32, No. 3, pp. 586-615, 2003.

[5] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, H, “Aggregate and verifiably encrypted signatures from bilinear maps,” *EUROCRYPT 2003*, LNCS 2656, pp.416-432, 2003.

[6] D. Boneh, G. Gentry, and B. Waters, “Collusion resistant broadcast encryption with short ciphertexts and private keys,” *CRYPTO 2005*, LNCS 3621, pp.258-275, 2005.

[7] K. Chalkias and G. Stephanides, “Timed release cryptography from bilinear pairings using hash chains,” *CMS 2006*, LNCS 4237, pp. 130-140, 2006.

[8] 電子商取引推進協議会, “タイムスタンプサービスの利用ガイドライン,” `http://www.jipdec.or.jp/archives/ecom/results/h14seika/h14results-17.pdf`

[9] F. Hess, “Exponent group signature schemes and efficient identity based signature schemes based on pairings,” `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.2741&rep=rep1&type=pdf`

[10] F. Hess, N. P. Smart, and F. Vercauteren, “The Eta pairing revisited,” *IEEE Transactions on Information Theory*, Vol. 52, pp.4595-4602, 2006.

[11] 岡本義明, 齋藤泰一, 藤岡淳, “ワンタイム署名を使わない公開鍵型 Timed-Realease 暗号の一般的な構成法,” *SCIS2010*, 2C3-2, 2010.

[12] R. Sakai, K. Ohgishi, and M. Kasahara, “Cryptosystems based on pairing,” *SCIS 2000*, 2000.

[13] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” *EUROCRYPT 2005*, LNCS 3494, pp.457-473. 2008.

[14] F. Zhang and K. Kim, “ID-based blind signature and ring signature from pairings,” *ASIACRYPT 2002*, LNCS 2501, pp.629-637, 2002.