SCIS 2004 The 2004 Symposium on Cryptography and Information Security
Sendai, Japan, Jan.27-30, 2004
The Institute of Electronics, Information and Communication Engineers

ツリー構造による署名方式とその安全性証明
Signature schemes based on tree structures and their security proof

西垣 秀哉* 真鍋 義文*† 岡本 龍明*‡
Hideya Nishigaki Yoshifumi Manabe Tatsuaki Okamoto

あらまし 本稿では，ある一定回数の使用については安全であると考えられる署名方式に対して，ツリー構造を導入することで任意の使用回数に対して安全性が保証される署名方式の一般的構成を示し，その安全性証明を行う．特に，偽造確率が署名使用回数に対して線形以上に高くなる電子署名方式に対して，この方式は有効である．また具体的に，2つの署名方式への適用例を示す．まず，NTRUSignに対して適用することで，NTRUSignの安全性を向上させるとともにRSA暗号に基づく署名方式よりも処理性能が優れた方式が構成可能であることを示す．さらに，Fail-Stop署名の使い捨て署名方式に対して有効に適用できることを示す．

キーワード 電子署名方式，ツリー構造，安全性証明．

1 はじめに

現在までに様々な電子署名方式が提案されている．しかし，妥当な仮定の下でその安全性が証明されているものはそれほど多くない．本稿では，ある一定回数の使用については安全であると考えられる署名方式に対して，ツリー構造を導入することで任意の使用回数に対して安全性を保証できる方式を示し，その安全性証明を行う．また，最後に簡単に具体例を示し，適用例を検討する．

2 電子署名とその安全性

前提となる知識を以下にまとめる．

2.1 電子署名のモデル

電子署名スキームΠは，鍵生成アルゴリズム Gen，署名アルゴリズム $Sign$，検証アルゴリズム $Verify$ の3つのアルゴリズムから成る．

2.1.1 鍵生成アルゴリズム Gen

入力 1^k が与えられたとき，公開鍵と秘密鍵の組 (pk, sk) を出力する多項式時間確率的アルゴリズム．$(pk, sk) \leftarrow Gen(1^k)$．($k$ は，鍵の長さ．セキュリティパラメータとも考えられる．)

2.1.2 署名アルゴリズム $Sign$

メッセージ M と秘密鍵 sk の入力に対して，署名 σ を出力する多項式時間確率的アルゴリズム．$\sigma \leftarrow Sign_{sk}(M)$．

2.1.3 検証アルゴリズム $Verify$

メッセージ M，M の署名 σ，公開鍵 pk から，σ が正当な署名であるかを判定する多項式時間確定的アルゴリズム．その署名が検証に合格した場合は1を返し，不合格の場合は0を返す．$\{1, 0\} \leftarrow Verify_{pk}(M, \sigma)$．ここで，$Verify_{pk}(M, Sign_{sk}(M)) = 1$ である．

2.2 署名の安全性

署名に関する安全性は，2つの観点から考えることができる．1つはどのような偽造が可能であるかという観点であり，もう1つは攻撃者にどのような攻撃方法を許すかという観点である．

2.2.1 偽造の種類

偽造できるレベルによって，次のような安全性が存在する．

1. **total break**
   攻撃者が署名者の秘密鍵の情報を計算することができる．または，正当な署名アルゴリズムと等価な効率の良いアルゴリズムにより，偽造署名作成が可能となる．

* 京都大学〒 606-8501 京都市左京区吉田本町 Kyoto University Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501 JAPAN, nisigaki@kuis.kyoto-u.ac.jp
† NTT サイバースペース研究所，〒 239-0847 神奈川県横須賀市光の丘 1-1. NTT Cyber Space Laboratories, 1-1 Hikarinooka, Yokosuka-shi, Kanagawa-ken, 239-0847 JAPAN.
‡ NTT 情報流通プラットフォーム研究所，〒 239-0847 神奈川県横須賀市光の丘 1-1. NTT Information Sharing Platform Laboratories, 1-1 Hikarinooka, Yokosuka-shi, Kanagawa-ken, 239-0847 JAPAN.

2. **selective forgery**
   攻撃者が選んだ，ある特定のメッセージに対する署名を偽造することができる．

3. **existential forgery**
   少なくとも 1 つのメッセージに署名することができる．(このメッセージは，意味のあるものでなくてもよい．)

2.2.2 署名に対する攻撃

どこまでの情報を攻撃者に与えるかに応じて，以下のような攻撃方法に分類できる．

1. **key-only attack (passive attack)**
   署名者の公開鍵だけを用いて攻撃する．

2. **message attack**
   攻撃者はすでに作成されたあるメッセージに対する署名，もしくは攻撃者が選択したメッセージに対する署名を用いて攻撃する．これは，以下のタイプに分類される．

   2-1 known-message attack 攻撃者は (攻撃者自身が選んだものではない) いくつかのメッセージに対する署名の集合をもとに攻撃する．

   2-2 chosen-message attack 攻撃者自身が選択した複数のメッセージに対する署名を手に入れることができるが，署名を見てからメッセージを選択することはできない．

   2-3 adaptive chosen-message attack 攻撃者自身が選択したメッセージに対する署名を手に入れることができ，手に入れた署名を検討してから新たに署名をつけたいメッセージを選ぶことができる．これを複数回繰り返すことができる．

2.2.3 最も望まれるべき安全性

署名に望まれる最も高い安全性は，adaptive chosen-message attack に対して，existential forgery ができないことである．以下では，この安全性を満たすものを”安全な”電子署名方式であると呼ぶ．

2.3 安全性の定義

署名の安全性に対しての定量的な定義を与える．まず，多項式時間チューリングマシンである攻撃者 Adv を仮定する．そして，この攻撃者 Adv を用いて偽造を行うシナリオを構成することで安全性の定義を行う．

鍵生成アルゴリズムにより，1^k の入力に対して，内部で乱数 $R(Gen)$ を用いて公開鍵と秘密鍵の組 (pk, sk) が出力される．(以下では，$R(X)$ で X 内部で生成した乱数を表す．) $(pk, sk) \stackrel{R(Gen)}{\longleftarrow} Gen(1^k)$．この pk を攻撃者 Adv に与える．攻撃者 Adv は署名オラクル SO への問い合わせを q_{sig} 回行いながら，最終的に偽造署名 (M', σ') を出力する．$(M', \sigma') \stackrel{R(Adv)}{\longleftarrow} Adv^{SO}(pk)$．ここで，$Verify_{pk}(M', \sigma') = 1$ となったとき偽造に成功したことになる．

この署名偽造確率 ϵ を以下のように定義する．

$$
\begin{aligned}
\epsilon(k) &= Advantage(k) \\
&= \mathbf{Pr}_{R(Gen),R(Adv),R(SO)}[\\
&\quad (pk, sk) \stackrel{R(Gen)}{\longleftarrow} Gen(1^k); \\
&\quad (M', \sigma') \stackrel{R(Adv)}{\longleftarrow} Adv^{SO}(pk); \\
&\quad Verify_{pk}(M', \sigma') = 1]
\end{aligned}
$$

時間 $t(k)$ において少なくとも確率 $\epsilon(k)$ で偽造できる偽造者が存在するとき，この署名方式は (t, q_{sig}, ϵ)-break という．

定義 1

ある署名方式 $\Pi = (Gen, Sign, Verify)$ に対して，(t, q_{sig}, ϵ)-break である偽造者が存在しないとき，(t, q_{sig}, ϵ)-secure であるという．

3 TSS(Tree-Signature Scheme)

ある署名方式Sにツリー構造を導入した方式TSS(Tree-Signature Scheme) を検討する．これは，署名方式の 1 つの公開鍵と秘密鍵のペアを 1 つのノードとしてとらえて，それをツリー状に構成することでこのツリー全体を 1 つのスキームとしたものである．この TSS は，ある仮定のもとで安全性の証明が可能である．

以前からツリー構造を用いた署名方式は提案されてきた [1] [2]．しかし，これらはツリーにおける 1 つのノードの鍵ペアとして使い捨て署名方式 (1 回の署名作成のみ有効) を用いたものがほとんどであった．

しかし，現在ではさまざまな署名方式が提案されている．このツリー構造を組み入れることに適した方式と合わせれば，従来の方式と比べても劣ることのない方式として構成できる．

3.1 TSS のモデル

始めに，記号の定義を行う．

- N_{num} : num 番のノード.
- pk_{num}, sk_{num} : num 番のノードにおける，公開鍵・秘密鍵のペア.
- d : ツリーの深さ．ルートノードの深さは 0 とし，その子ノードから下に向かって $1, 2, 3, \cdots, D$ としている．

- $num = (d, z_1, z_2, \ldots, z_d)$.
  ただし，$0 \leq z_i \leq Q-1 (i = 1, 2, \ldots, d)$．$Q$ については後述．また，$d = 0$ のとき $num = (0)$ であるが，これを単に 0 と書く．

1 つのノードには秘密鍵・公開鍵ペアと子ノードの公開鍵の署名値を保持している，1 つの親ノードは Q 個の子ノードを持っている．ルートノードの秘密鍵 sk_0 を用いて，その Q 個の子ノードの公開鍵に署名をつける．次に，その署名がつけられた公開鍵のノードの秘密鍵を用いて，それぞれの子ノードの署名を行う．つまり，$num = (d, z_1, \ldots, z_d)$ のとき，ノード N_{num} は Q 個の子ノードを持ち，それらは $num = (d+1, z_1, \ldots, z_d, z_{d+1}), 0 \leq z_{d+1} \leq Q-1$ として表現される．以下，これを繰り返すことで，深さ d におけるノードの数は $(Q)^d$ 個となる．このツリーの葉ノードの鍵ペアを用いて，メッセージの署名に用いる．

以下で，鍵生成・署名・検証の手順を示す．

3.2 鍵生成 GenTSS

$$(pk_0, sk_0, l) \overset{R(GenTSS)}{\longleftarrow} GenTSS(1^k, Max)$$

(ここで Max は最大署名回数である．)

鍵生成の手順を以下に示す．署名方式 S を用いる．

1. $D = \lceil \log_Q(Max) \rceil - 1$.
2. $(pk_0, sk_0) \overset{R(GenTSS)}{\longleftarrow} GenS(1^k)$.
3. $l = 0$.

まず，最大署名使用回数 Max からツリーの深さ D の値を決め，ツリーの大きさが決まる．次に，ルートノードに対応する鍵ペアを，署名方式 S に従い生成する．署名作成においては，署名使用回数 l を管理しながら，ツリーの葉ノードを左から順番に使用してメッセージに署名する．

3.3 署名 SignTSS

$$\theta_l \leftarrow SignTSS_{sk_0}(M, l)$$

署名作成の手順を以下に示す．

1. $h_{D+1} = l$, $z_{D+1} = h_{D+1} \bmod Q$.
2. $d = D$. While $d \geq 1$:
   (a) $h_d = (h_{d+1} - z_{d+1})/Q$.
   (b) $z_d = h_d \bmod Q$.
   (c) $d = d - 1$.
3. $n_0^l = 0$ とする．
4. $d = 1$. While $d \leq D$:
   (a) $n_d^l = (d, z_1, \ldots, z_d)$.
   (b) $r = f(d||z_1||\cdots||z_d)$ とする．(ただし，ここで f は擬似ランダム関数とし，|| で結合を表す．)
   (c) $(pk_{n_d^l}, sk_{n_d^l}) \overset{r}{\longleftarrow} GenS(1^k)$.
   (d) $\sigma_{n_{d-1}^l} \leftarrow SignS_{sk_{n_{d-1}^l}}(pk_{n_d^l})$.
5. $\sigma_{n_D^l} \leftarrow SignS_{sk_{n_D^l}}(M)$.
6. $l = l + 1$.
7. Return :
   $\theta_l = (l, \sigma_{n_0^l}, \sigma_{n_1^l}, \ldots, \sigma_{n_D^l}, pk_{n_1^l}, \ldots, pk_{n_D^l})$

署名使用回数 l から，ルートノード N_0 からメッセージの署名に用いる葉ノードに至る経路が一意に決定する．h_d により，その深さ d における左から h_d 番目のノードであることを表している．また z_d により，1 つの親ノードが持っている子ノードの中で，左から z_d 番目であることを表している．この表現を用いて，ノード番号 num を $num = n_d^l = (d, z_1, z_2, \ldots, z_d)$ としている．

経路が決まればそれをルートノードから辿り，経路上のノードの鍵ペアを作成する．そして，親ノードの秘密鍵で子ノードの公開鍵に順に署名をつけていく．これにより，署名のチェーンが構成される．最後に，葉ノードの秘密鍵を用いて，メッセージ M に署名を行う．

注意 1 つのノードの鍵ペアは Q 回の使用を考えているので，途中の経路上のノードの鍵ペアを必要に応じてテンポラリメモリに保存しておく．こうすることで，鍵生成処理時間の短縮ができる．

3.4 検証 VerifyTSS

$$\{1, 0\} \leftarrow VerifyTSS_{pk_0}(M, \theta_l).$$

検証の手順を以下に示す．

1. $V = VerifyS_{pk_{n_D^l}}(M, \sigma_{n_D^l})$.
2. If V=0 then return 0.
   Else set $d = D - 1$, While $d \geq 1$:
   (a) $V = VerifyS_{pk_{n_d^l}}(pk_{n_{d+1}^l}, \sigma_{n_d^l})$.
   (b) If $V = 1$ then $d = d - 1$.
   Else return 0.

3. $V = VerifyS_{pk_0}(pk_{n_1^l}, \sigma_{n_0^l})$
   If $V = 1$ then return 1.
   Else return 0.

署名チェーンの検証を繰り返し，ルートノードまで辿る．ここで署名者の公開鍵 pk_0 により検証を行い，合格と判断されれば 1 を出力する．

4 安全性証明

上で示した方式 TSS の安全性について検討する．

4.1 安全性の証明

TSS の安全性について，以下の定理 1 が導ける．

定理 1

署名方式 S は，(t', Q, ϵ')-secure であると仮定する．この時，任意の q_{sig} に対して，TSS は (t, q_{sig}, ϵ)-secure である．

- $t = t' - \{NUM \cdot T_{GenS} + (NUM - 1 + q_{sig}) \cdot T_{SignS}\}$
- $\epsilon = \epsilon' \cdot (NUM - 1)$

$$\text{ただし, } NUM = \sum_{k=1}^{D+1} X_k$$

$$X_0 = q_{sig} \quad , \quad X_d = \lceil X_{d-1}/Q \rceil \quad (d = 1, 2, \ldots, D)$$

$$D = \lceil \log_Q(q_{sig}) \rceil - 1$$

T_{GenS}, T_{SignS} は，それぞれ TSS の鍵生成時間，署名作成時間を表している．

NUM はツリー全体のノード数である．

以下で，この定理 1 を証明する．

[証明] まず，仮定を置く．

仮定

(t, q_{sig}, ϵ)-break であるような攻撃者 Adv を仮定する．

この仮定から，署名方式 S を署名オラクルへの Q 回の問い合わせによって偽造可能な偽造者 F を導き出す．偽造者 F の振る舞いは，ある偽造署名に使いたい pk^* の入力に対して，なんらかのメッセージと偽造署名の組 (m^*, σ^*) を出力するようなシミュレータであると考えることができる．

先ほど仮定した攻撃者 Adv をうまく誘導してやることで，このシミュレータの内部を構成する．次で，このシミュレータの内部を詳細に説明する．

4.1.1 シミュレータの構成

まず，シミュレータの入力と出力は以下のように考えている．

- **入力** 偽造者 F が署名偽造に使用したい公開鍵 pk^*.
- **出力** (m^*, σ^*)：署名方式 S において，
  $VerifyS_{pk_0}(m^*, \sigma^*) = 1$ となる (m^*, σ^*) の組.

つまり，このシミュレータがうまく働き出力を得られたときは，偽造者 F が署名方式 S の偽造に成功しているということになる．

次に，シミュレータの内部は，以下の要素で構成している．

- **Adv** TSS に対して，(t, q_{sig}, ϵ)-break である攻撃者 Adv.
- **F-署名オラクル (F-SO)** 偽造者 F が攻撃者 Adv に対して返答を行う署名オラクル.

攻撃者 Adv は F-署名オラクルに q_{sig} 回の問い合わせを行うことで，TSS の偽造を行う．

また，シミュレータ外部の，

- **署名オラクル (SO)** 署名方式 S において，公開鍵とメッセージの問い合わせに対して，その公開鍵での検証に合格する正当な署名値を返答する署名オラクル.

の存在を用いている．これは，F-署名オラクルが返答できない部分，すなわち pk^* に対しての署名作成時に F-署名オラクルが問い合わせる．

4.1.2 シミュレータの動作

それでは次に，シミュレータの動作をまとめる．

1. $D = \lceil \log_Q(q_{sig}) \rceil - 1$, $NUM = \sum_{k=1}^{D+1} X_d$
   (ただし, $X_0 = q_{sig}, X_d = \lceil X_{d-1}/Q \rceil (d = 1, \ldots, D)$)
2. $num' = (d, z_1, z_2, \ldots, z_d)$ をランダムに決める．
   ただし，$1 \leq d \leq D$ で，$0 \leq z_i \leq Q - 1 (i = 1, 2, \ldots, d)$ の範囲でランダムに z_i を選ぶ.
3. $(m', \theta') \overset{R(Adv)}{\longleftarrow} Adv^{F\text{-}SO}(pk_0)$.
   ただし, $\theta' = (l', \sigma_{n_0^{l'}}, \sigma_{n_1^{l'}}, \ldots, \sigma_{n_D^{l'}}, pk_{n_1^{l'}}, \ldots, pk_{n_D^{l'}})$
4. Set $d = 1$. While $d \leq D$:
   If $\sigma_{n_{d-1}^{l'}} \neq \sigma_{n_{d-1}^{l}}$ then
   If $pk_{n_{d-1}^{l'}} = pk^*$
   then return $(m^*, \sigma^*) = (pk_{n_d^{l'}}, \sigma_{n_{d-1}^{l'}})$
   Else d=d+1.

5. If $\sigma_{n^{l'}_D} \neq \sigma_{n^{l}_D}$ then
   If $pk_{n^{l'}_D} = pk^*$
   return $(m^*, \sigma^*) = (m', \sigma_{n^{l'}_D})$

(m^*, σ^*) が出力されたとき，偽造者 F の署名方式 S への偽造は成功したことになる．何も出力されない場合は，偽造者 F の偽造は失敗している．

ここで，攻撃者 Adv からの問い合わせに対して，F-署名オラクルは以下のサブルーチンを実行している．
INPUT：M_j.
(ただし，$0 \leq j < Q^{D+1}$ として攻撃者 Adv がランダムに指定する．)

1. $h_{D+1} = j$，$z_{D+1} = h_{D+1} \bmod Q$.
2. $d = D$. While $d \geq 1$:
   (a) $h_d = (h_{d+1} - z_{d+1})/Q$.
   (b) $z_d = h_d \bmod Q$.
   (c) $d = d - 1$.
3. $n^j_0 = 0$ とする.
4. Set $d = 1$. While $d \leq D$:
   (a) $n^j_d = (d, z_1, \ldots, z_d)$.
   (b) If $num' = n^j_d$ then
   $pk_{n^j_d} = pk^*$.
   Else $r = f(d||z_1||\cdots||z_d)$,
   $(pk_{n^j_d}, sk_{n^j_d}) \xleftarrow{r} GenS(1^k)$.
   (c) $d \geq 2$ において，
   If $num' = n^j_{d-1}$ then
   $\sigma_{n^j_{d-1}} \leftarrow SO(pk^*, pk_{n^j_d})$
   Else $\sigma_{n^j_{d-1}} \leftarrow SignS_{sk_{n^j_{d-1}}}(pk_{n^j_d})$.
5. $\sigma_{n^j_0} \leftarrow SignS_{sk_0}(pk_{n^j_1})$.
6. If $num' \neq n^j_D$ then
   $\sigma_{n^j_D} \leftarrow SignS_{sk_{n^j_D}}(M_j)$
   else $\sigma_{n^j_D} \leftarrow SO(pk^*, M_j)$
7. Return $\theta'_j = (j, \sigma_{n^j_0}, \sigma_{n^j_1}, \ldots, \sigma_{n^j_D}, pk_{n^j_1}, \ldots, pk_{n^j_D})$.

4.1.3 攻撃者 Adv の攻撃

攻撃者 Adv の視点　攻撃者 Adv の視点からは，署名オラクルに問い合わせをしているという見え方しかしない．シミュレータの場合はその署名オラクルが公開鍵 pk^* に対する真の秘密鍵 sk^* を持たない F-署名オラクルであり，リアルの場合は真の秘密鍵 sk^* を持つ署名オラクルである．

以下では，シミュレーションとリアルのそれぞれにおける攻撃者 Adv の情報の見え方の差異について検討する．

攻撃者 Adv の見える情報として，

1. a：事前に与えれらる外部情報.
2. pk_0：TSS における公開鍵.
3. r：内部の乱数 $R(Adv)$.
4. $\theta'_0, \theta'_1, \ldots, \theta'_{q_{sig}-1}$：F-署名オラクルからの返答.
5. $\theta_0, \theta_1, \ldots, \theta_{q_{sig}-1}$：署名オラクルからの返答.

があげられる．

攻撃者 Adv からの視点は，それぞれ，

シミュレータの場合

$$View_{Adv}(sim) = (a, pk_0, r, \theta'_0, \ldots, \theta'_{q_{sig}-1})$$

リアルの場合

$$View_{Adv}(real) = (a, pk_0, r, \theta_0, \ldots, \theta_{q_{sig}-1})$$

と表すことができる．

ここで，a, pk_0 はあらかじめ与えられる情報であるので同じものである．また，r も攻撃者 Adv 内部のものであるので同じである．異なるものは θ'_0 や θ_0 などの返答である．これは sk^* の有無によるものであるが，F-署名オラクルも外部の署名オラクルに問い合わせることで真の署名オラクルと同様に振る舞うことができる．

よって，$\forall n \in \mathbf{N}$，$\forall X \in \{0,1\}^n$ として，

$$\mathrm{Prob}[View_{Adv}(real) \rightarrow X] = \mathrm{Prob}[View_{Adv}(sim) \rightarrow X]$$

と考えられる．

攻撃者 Adv の攻撃　攻撃者 Adv の攻撃とはどのような場合があるか検討してみる．

偽造者 Adv の出力は，(m', θ') である．これは，以下の条件を満たしている必要がある．

(1) 署名検証に合格する.

(2) $(m', \theta') \notin \{(m_0, \theta_0), (m_0, \theta_0), \ldots, (m_{q_{sig}-1}, \theta_{q_{sig}-1})\}$

(1) から，ルートノードの鍵ペアの公開鍵 pk_0 で検証されなければならないということであるので，ルートノードとチェーンで繋がったノードの鍵ペアを偽造したということに他ならない．また (2) から，それが問い合わせの返答に用いられた鍵ペア以外の新たな鍵ペアのノードを付け足して署名をつけたか，もしくは既存の鍵ペアを偽造したという場合に限られる．このことは，ツリーのどこかのノードの鍵ペアを偽造したということである．よって，攻撃者 Adv が行う偽造とは，F-署名オラクルが問い合わせの返答に作成したツリーのどこかのノードに新たな枝を付けたし偽造を行う場合のみであると結論付けることができる．

4.1.4 偽造者 F の導出と評価

偽造者 F の導出 攻撃者 Adv が出力した偽造に使用した公開鍵 pk' が，偽造者 F が偽造したい公開鍵 pk^* と一致していたとき，攻撃者 Adv が出力したあるメッセージと偽造署名の組 (m', θ') に偽造者 F が偽造したかった pk^* に対しての署名値が含まれている．よってこのとき，ある署名方式を署名オラクルへの Q 回の問い合わせによって偽造可能な偽造者 F が存在するということになる．

攻撃者 Adv が偽造に成功する確率が ϵ' であり，攻撃者 Adv が出力した偽造に使用した公開鍵 pk' が，偽造者 F が偽造したい公開鍵 pk^* と一致する確率が $\frac{1}{(NUM-1)}$ であると考えることができるので，偽造者 Adv が偽造に成功する確率は

$$\epsilon' = \epsilon \cdot \frac{1}{(NUM-1)}$$

と評価することができる．

また，この計算時間は，攻撃者 Adv の攻撃に要する時間 t と偽造者 F が F-署名オラクルとしてツリーを作成する時間の和として見積もることができるので，

$$\begin{aligned} t' &= t + \{NUM \cdot T_{GenS} + \\ &\quad (NUM - 1 + q_{sig}) \cdot T_{SignS} \end{aligned}$$

と評価できる．

以上の議論により，ある署名方式 S は「(t', Q, ϵ')-break である.」ということが導けた．

しかし，これはそもそもの前提,「ある署名方式 S は (t', Q, ϵ')-secure である.」ということに反する．よって背理法より，仮定は否定される．よって，TSS は (t, q_{sig}, ϵ)-secure であると結論付けることができる．

(証明終わり)

5 具体例

5.1 TSS-NTRUSign

表 1 に，NTRUSign-251，ECDSA-163，RSA-1024 の 3 つのスキームにおける鍵生成・署名・検証の処理時間について，800MHz のペンティアムマシンで動作させた結果を [3] より引用する．

	NTRUSign-251	ECDSA-163	RSA-1024
鍵生成	180,000	1424	500,000
署名	500	1424	9090
検証	303	2183	781

表 1: NTRUSign-251，ECDSA-163，RSA-1024 の処理時間．単位は [μs] である．

TSS-NTRUSign において $Q = 10^3, d = 2$ としたとき，以下の時間で処理できる．(署名作成時間にツリー全体のノード作成時間を平均して加えている．)

	NTRUSign-251-TSS
鍵生成	180,000
署名	680
検証	909

表 2: TSS-NTRUSign-251 の処理時間．単位は [μs] である．

5.2 TSS-FailStop

使い捨て FailStop 署名にツリー構造を導入することにより，先ほどの証明の手順と同様に，安全性の証明を行うことができる．ただ，使い捨て署名方式，つまり $Q = 1$ の場合は，署名作成においてはすべての子ノードの公開鍵に一度に署名を付けることを行う．

6 最後に

本稿では，ある一定回数の使用については安全であると考えられる署名方式に対して，ツリー構造を導入することで任意の使用回数に対して安全性を保証できる方式を示し，その安全性証明を行った．高性能かつ小さい値 Q に対して安全性が保証されるような署名方式の検討が今後の課題である．

参考文献

[1] R. C. Merkle,"A Certified Digital Signature," Proceedings of Crypto89, Lecture Notes in Computer Science Vol. 435, pp 218-238, 1989.

[2] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," Proceedings of the 21st Annual Symposium on Theory of Computing, ACM, 1989.

[3] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. Silverman and W. Whyte, "NTRUSign: Digital Signatures Using the NTRU Lattice.," in CT-RSA 2003, To appear. `http://www.ntru.com/`

[4] Torben Pryds Pedersen and Birgit Pfitzman, "Fail-Stop Signatures," SIAM Journal on Computing, Volume 26, Number 2, pp. 291-330, 1997.