社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2003－ＣＳＥＣ－22　(20)
2003／7／18

署名生成機能の危殆化を検出できるディジタル署名方式

上山 真貴子† 四方 順司† 松本 勉†

†横浜国立大学大学院環境情報研究院　〒240-8501 神奈川県横浜市保土ヶ谷区常盤台 79-7
E-mail: †{ueyama,shikata,matsumoto}@mlab.jks.ynu.ac.jp,

あらまし　ディジタル署名技術の需要の高まりとともに，署名生成機能の危殆化対策が求められている．危殆化対策として，Forward-Secure Digital Signature やヒステリシス署名などがあるが，これらの方式では本来の署名生成者が検証者からの問い合わせを受けたり，偶然に危殆化を発見したりすること以外に自身の署名生成鍵が漏洩した事実に気づくことができない．そのため漏洩後の対策が遅れてしまい，結果として署名生成鍵の漏洩による被害の拡大につながる恐れがある．本稿では，署名生成鍵の漏洩後の被害を減少させるために，本来の署名生成者が，署名生成鍵が物理的に漏洩したことを早期に検出できる署名方式を提案する．

キーワード　ディジタル署名，署名生成機能の危殆化，forward-secure digital signature

A Digital Signature Scheme That Can Detect the Compromise of Signing Ability

Makiko UEYAMA† Junii SHIKATA† and Tsutomu MATSUMOTO†

† Graduate School of Environment and Information Sciences, Yokohama National University,
79-7 Tokiwadai, Hodogaya-ku, Yokohama-shi, Kanagawa, 240-8501 Japan
E-mail: †{ueyama,shikata,matsumoto}@mlab.jks.ynu.ac.jp

Abstract　The measures against compromise of signing ability are becoming important with the rise of the demand for the digital signature technology. As measures against the exposure of the signing key which is one of the cases of the compromise of signing ability, several signature schemes such as Forward-Secure Digital Signature and Hysteresis signature were proposed. However, even if the legitimate signer uses those measures, without a discovery of the forged signature or the inquiry from the verification process he cannot notice the fact that his signing key has been exposed. Therefore, a safety measure after the exposure of signing key would be delayed, which may lead to a more serious damage. In this paper, to reduce the damage after the exposure of signing key, we propose a digital signature scheme that can detect the compromise of signing ability at the early stage. By using this scheme, we can reduce the damage caused by exposure of signing key.

Keyword　digital signature, compromise of signing ability, forward-secure digital signature

1. はじめに

1.1. 背景

ディジタルデータを有効に利用する上で，いつ存在したデータなのか，誰が生成したデータなのかといった証拠を付加するディジタル証拠性技術の需要が高まっている．なかでもディジタル署名技術は主要なディジタル証拠性技術であるが，署名生成機能の危殆化問題が未解決である[2].

ディジタル署名は本来，正当なユーザ（Alice）だけが生成できるようにするべきである．しかし，何らかの原因で正当なユーザではない第3者（Charlie）に署名生機能が与えられてしまう場合が考えられ，本稿ではそれを署名生成機能の危殆化とよぶことにする．また，Charlieが生成した一見Aliceが生成したように見える署名文を偽造署名とよぶことにする．

署名生成機能が危殆化した場合，Aliceや署名検証者（Bob）が偽造署名を提示されることによって不利益を被る可能性がある．また同時に，偽造署名と正当な署名の区別ができないためにAliceのディジタル署名の信頼性が失われることも考えられる．そのため，Aliceが偽造署名による被害を抑え，信頼性を維持するためには，危殆化した署名生成機能を用いて生成された署名を失効させて対処することが考えられる．ところが，この場合正当な署名も偽造署名と同じ署名生成鍵で生成されているので，失効手続きの結果偽造署名とともに失効してしまう．つまり，署名生成機能が危殆化したことによって，偽造署名の生成，信頼性の喪失，失効してしまう正当な署名という3つの被害が発生する．従って，ディジタル署名を安心して利用するためには，これらの被害を抑えるような署名生成機能の危殆化対策を考える必要がある．

1.2. 従来の対策技術

Forward-Secure Digital Signature（FSDS）[3]は，署名生成鍵を一定期間ごとに更新し，危殆化した期間以前に生成された正当な署名を失効させずにすますことにより，危殆化による被害を抑えようとする署名方式である．しかし，高橋らが指摘したように，危殆化した期間を特定できないので実際には意図したような効果は得られない[6]．一方，危殆化した時期を知ることができればFSDSが効果を発揮し，早く知るほど偽造署名による被害を抑えることができるはずである．そこで，我々は危殆化を検出することが被害を抑える効果につながることに注目した．

危殆化を検出しようとする考え方には，安斎らによって提案された複製端末発見方式がある[7]．この方式は，携帯電話などの端末を対象とするサービスを提供する機関が，複製端末が不正にサービスを利用することにより生じる被害を抑えるためユーザにサービス供給時に用いる共通鍵を更新させ，その際に用いる乱数によって複製端末の存在を検出しようというものである．我々は同じようにして署名生成機能の危殆化が検出できると考えた．

1.3. 目的と要件

危殆化対策技術に求められているのは，危殆化による被害の軽減である．従来の危殆化対策技術には危殆化を検出する機能が欠けており，そのために被害が拡大する可能性があった．そこで，より一層危殆化による被害を抑えることを可能とするために，本稿ではディジタル署名方式に危殆化を検出できる機能を新たに加えることを目的とする．

そこで，我々は署名生成鍵の更新時に危殆化を検出する方向で考える．FSDSのように署名生成鍵を更新することにより危殆化の範囲を限定することができる上に，危殆化した事実を検出して失効手続きなどの対応をとり，被害の拡大を抑えることが可能となる．

Aliceが鍵更新時に危殆化を検出することを可能にするには，危殆化の痕跡を鍵更新に反映させればよい．そのために，鍵更新に必要な信頼できる第3者機関であるTTP(Trusted Third Party)を設け，TTPに危殆化の痕跡，つまり攻撃者Charlieが鍵更新をした記録を残すことを考える．このようなディジタル署名方式を考えるに当たって以下の環境を想定する．

エンティティとしては署名者Alice，署名検証者Bob，攻撃者Charlie，信頼できる第3者機関TTP（Trusted Third Party）を考える．そして，TTPがCharlieによってデータを奪われることはないと仮定する．Charlieは Aliceの署名生成機能を入手し，偽造署名を生成することを目的とした攻撃を行う．ただし，署名生成機能の入手は容易ではないとする．さらに，Charlieは AliceとTTP間の通信を傍受してそのデータを入手することができるとし，また，何らかの原因で通信がうまくいかない可能性があることも考慮する．

登録時と失効手続き時のAliceとTTP間の認証については，それらが確実に行われることを前提とする．つまり，登録時に確実に署名生成鍵と署名検証鍵が正当なユーザであるAliceの手元に渡り，失効作業を請求できるのも正当なユーザであるAliceだけであるとする．

本稿では，このような想定の下で次の要件を満たすように， TTPを利用して鍵更新を行い，Aliceが危殆化を検出できる機能を持つ署名方式を提案する．

要件1．署名方式としては従来と同程度の安全性を持つこと

要件2．署名生成機能が危殆化した事実をAliceが偶然に頼らずに検出することができる機能を持つこと

2. 提案方式

2.1. 概要

TTP は複数のユーザの鍵更新を扱うが，ここでは簡単のため，Alice が期間 *i-1* から期間 *i* への鍵更新を行う場合を以下に説明する．

Alice は短い期間ごとに署名生成鍵を更新する．危殆化の検出には，どの期間の署名生成機能が危殆化したかを知る必要がある．また，危殆化を発見した後の対処時には検証対象のディジタル署名がどの期間の署名生成機能によって生成されたかを知る必要がある．そこで，ひとつの署名生成鍵 SK_{i-1} に対し，一対一に対応するラベル $L_{i-1}^{(n)}$ を用意し，SK_{i-1} から生成したディジタル署名を正しく検証するには $L_{i-1}^{(n)}$ が必要であるような署名方式を考える．また，ラベルを発行する機関が TTP であるとする．

Alice は鍵更新時に TTP との通信を必要とするが，何らかの原因で通信に失敗したとき，TTP に再び更新を要求する必要がある．従って，同一期間に複数回鍵更新手続きが実行される可能性がある．ここで，同一期間に Alice によって生成された更新要求のうち何番目であるかを示す番号を *j*，同一期間に TTP が生成した更新記録のうち何番目であるかを示す番号を *n* と置く．Alice は期間 i-1 から *i* への *j* 番目の鍵更新時に更新依頼記録 $CR_i^{(j)}$ を記録し，更新要求 $C_i^{(j)}$ と暗号化鍵 $epk_i^{(j)}$ を SK_{i-1} で署名して TTP に送る．TTP は Alice の期間 *i*-1 から *i* への *n* 番目の更新時に生成した更新履歴 $D_i^{(n)}$ を記録し，Alice へ更新変数 $a_i^{(n)}$ と期間 *i*-1 から *i* への *n* 番目の更新記録 $\delta_i^{(n)}$ を送る．ラベル $L_i^{(n)}$ は $\delta_i^{(n)}$ に含まれる．このとき，$C_i^{(j)}$ の値として同一の値は生成できないものとする．ただし，Alice の同一期間の $C_i^{(j)}$ は全て同じ値を用いることにする．また $D_i^{(n)}$ は Alice の全ての（$a_i^{(n)}$，$\delta_i^{(n)}$）の組を記録したものとする．署名検証鍵 *PK* は一定である．

TTP は更新要求ごとに異なる値を返すので，更新後の署名生成鍵で生成された偽造署名を区別することができ，危殆化した期間以外の正当な署名を保護することができる．従って，Alice は危殆化によって新たな署名検証鍵を得る必要がなく，継続して同じ署名検証鍵を利用できる．（図 2.1 参照）

危殆化検出にあたっては，次の３つの場合が挙げられる：（１）更新要求の署名から得るラベルと TTP の最新の更新記録との矛盾を検出した場合，（２）Alice が得た更新記録と更新依頼記録との間に矛盾を検出した場合，（３）従来どおり Alice が偶然偽造署名を発見した場合．

従って，この方式を利用することによって，Alice は鍵更新時に Charlie によって署名生成機能が危殆化したことを知り，すばやく対応することで被害を抑えることが可能となる．

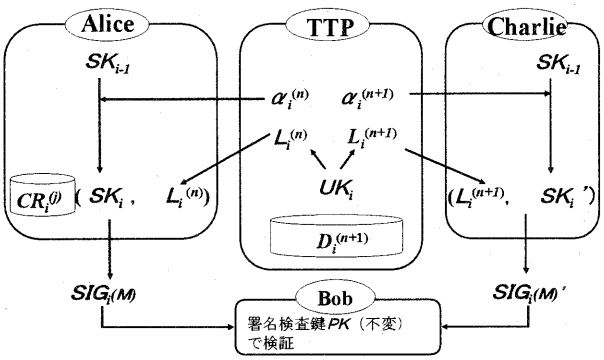

図 2.1　提案方式の概要

2.2. モデル

(0) 記号

i：期間を示す番号であり，*i*=0,1,2,3,…とする．

j：同じ期間内で更新要求を何番目に生成したかを示す番号

n：同じ期間内において何番目の更新記録かを示す番号

SK_i：期間 *i* における Alice の署名生成鍵

UK_i：Alice の署名生成鍵を SK_{i-1} から SK_i に更新するときに TTP が用いる更新鍵

PK：Alice の署名生成鍵 SK_i に対応する署名検証鍵

$epk_i^{(j)}$：Alice が期間 *i*-1 から *i* への更新作業に用いる暗号化方式の暗号化鍵で，*j* 番目に用いるもの

$esk_i^{(j)}$：Alice が期間 *i*-1 から *i* への更新作業に用いる暗号化方式の復号鍵で，*j* 番目に用いるもの

tpk：TTP の暗号化用公開鍵

tsk：TTP の復号用秘密鍵

$C_i^{(j)}$：Alice の期間 *i* のための *j* 番目に生成した更新要求で，Alice 以外が生成できないパラメータ

$C_0^{(1)}$：Alice の登録要求

$CR_i^{(j)}$：Alice の期間 *i* における *j* 番目の更新依頼記録

$\delta_i^{(n)}$：TTP が記録する Alice のための期間 *i* への *n* 番目の更新記録

$\delta_0^{(1)}$：TTP が記録する Alice のための発行記録

δ_{last}：TTP が記録している最新の Alice の更新記録

$D_i^{(n)}$：TTP が記録する Alice のための期間 *i* への *n* 番目の更新記録までの全ての更新変数と更新記録の組を記録した更新履歴

$D_0^{(1)}$：TTP が記録する Alice のための発行履歴

D_{last}：TTP が記録している最新の Alice の更新履歴

$a_i^{(n)}$：Alice の期間 *i* のための更新作業に用いる更新変数で，*n* 番目に発行されたもの

$T_i^{(n)}$：TTP が Alice の期間 *i* のための更新要求を受信した日時データで，*n* 番目に発行されたもの

$P_0^{(1)}$：Alice のラベルの初期パラメータ

$P_i^{(n)}$：Alice の期間 i における TTP が n 番目に発行したラベルのパラメータ

M：署名対象メッセージ

σ_i：SK_iによって生成された署名用データ

$SIG_i(M)$：SK_iによるMの署名つきデータ

$L_i^{(n)}$：$(P_i^{(n)}, i, T_i^{(n)})$ を SK_i に対応するラベル $L_i^{(n)}$ と呼ぶ

$\omega_i^{(j)}$：Alice の期間 i のための j 番目に生成した更新要求に対して生成された警告するデータで，対応するラベルが入る

$W_i^{(j)}$：Alice の期間 i のための j 番目に生成した更新要求に対して得られるそのときまでの Alice の警告履歴

W_{last}：TTP が記録している最新の Alice の警告履歴

$A_i^{(j)}$：Alice が期間 i への j 番目の更新要求に対する返信から生成した失効要求

k：セキュリティパラメータ

λ：乱数

(1)準備

TTP がセキュリティパラメータ k を入力して Alice のための署名生成鍵 SK_0，更新鍵 UK_0，署名検証鍵 PK，ラベルの初期パラメータ $P_0^{(1)}$を出力するアルゴリズムを鍵生成アルゴリズム GEN とする：

$$GEN(1^k) = (SK_0, UK_0, PK, P_0^{(1)}).$$

GEN を実行した後，Alice からの登録要求 $C_0^{(1)}$とそれを受領した日時データ $T_0^{(1)}$，ラベルの初期パラメータ $P_0^{(n)}$を入力し，発行記録 $\delta_0^{(1)}$，発行履歴 $D_0^{(1)}$を出力するアルゴリズムを登録アルゴリズム REG とする：

$$REG(C_0^{(1)}, T_0^{(1)}, P_0^{(1)}) = (D_0^{(1)}, \delta_0^{(1)}).$$

このときの発行記録，発行履歴，ラベルを次に示す：

$$D_0^{(1)} = \delta_0^{(1)} = (C_0^{(1)}, L_0^{(1)})$$

$$L_0^{(1)} = (P_0^{(1)}, 0, T_0^{(1)})$$

TTP は発行履歴 $D_0^{(1)}$を記録し，Alice に署名生成鍵 SK_0と署名検証鍵 PK，発行記録 $\delta_0^{(1)}$を渡す．

ここで，Alice は公開鍵証明書を Alice が認証局(CA)に発行してもらう場合と，TTP が CA に発行してもらう場合，そして TTP が CA を兼任する場合の 3 通りの方法で入手することができる．

(2)署名生成

期間 i において，署名対象メッセージ M，署名生成鍵 SK_i，ラベル $L_i^{(n)}$，署名生成の都度新たに生成する乱数 λを入力して署名$(\sigma_i, L_i^{(n)})$を生成するアルゴリズムを署名生成アルゴリズム SIG とする：

$$SIG(M, SK_i, L_i^{(n)}, \lambda) = (\sigma_i, L_i^{(n)}).$$

また，署名付きデータ $SIG_i(M)$を次のように表す：

$$SIG_i(M) = (M, \sigma_i, L_i^{(n)}).$$

(3)署名検証

Bob が署名付きデータ $SIG_i(M)$と署名検証鍵 PK を入力し，その署名がラベルと M の両方と整合する場合を *valid*，そうでない場合を *invalid* として判定するアルゴリズムを署名検証アルゴリズム VER とする：

$$VER(PK, SIG_i(M)) = \begin{cases} 1 & if\ valid \\ 0 & if\ invalid \end{cases}.$$

Bob は，出力で 1 を得た場合で，かつ失効リストに検証対象のラベルが載っていない場合に，その署名を有効と判断する．

(4)鍵更新手続き

期間 i-1 から期間 i に更新する過程を説明する．更新手続き全体のフローチャートを図 2.2 に示す．

Alice は更新要求を TTP に送信する都度，新しい暗号化鍵 $epk_i^{(j)}$と復号鍵 $esk_i^{(j)}$を生成する．また，TTP に送る更新要求 $C_i^{(j)}$を生成し，更新依頼記録 $CR_i^{(j)}$を以下のように生成して記録する：

$$CR_i^{(j)} = (C_i^{(j)}, j, esk_i^{(j)}).$$

このとき，前回の依頼記録が残っていれば消去する．次に，Alice が更新要求 $C_i^{(j)}$および暗号化鍵 $epk_i^{(j)}$のペア $(C_i^{(j)}, epk_i^{(j)})$ を署名生成鍵 SK_{i-1}とラベル $L_{i-1}^{(n)}$で署名し，更に TTP の暗号化鍵 tpk で暗号化して TTP に送る．

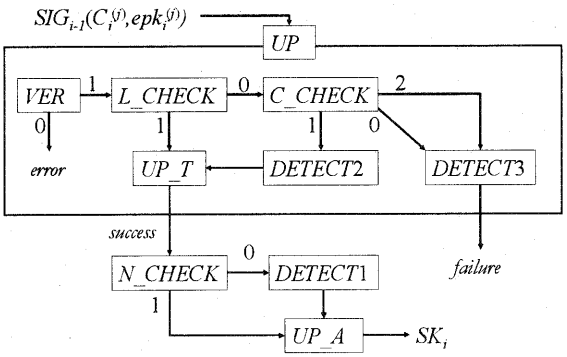

図 2.2　更新手続きのフローチャート

一方，TTP が Alice によって送られてきたデータを復号鍵 tsk で復号して得た署名付きデータ $SIG_{i-1}(C_i, epk_i^{(j)})$を入力し，更新が成功した場合を *success* として更新変数 $\alpha_i^{(n)}$と更新記録 $\delta_i^{(n)}$，警告履歴 $W_i^{(j)}$の組を出力し，更新が失敗した場合を *failure* として警告履歴

$W_i^{(j)}$のみを出力するアルゴリズムを更新アルゴリズム *UP* とする：

$$UP(SIG_{i-1}(C_i^{(j)}, epk_i^{(j)}) = \begin{cases} (\alpha_i^{(n)}, \delta_i^{(n)}, W_i^{(j)}) & if\ success \\ W_i^{(j)} & if\ failure \end{cases}.$$

success, *failure* の定義については（5）で説明する．

次に，TTP は *UP* の出力結果を $epk_i^{(j)}$で暗号化して Alice に送る．

Alice は TTP から送られてきたデータを復号鍵 $esk_i^{(j)}$で復号し，$W_i^{(j)}$のみを得た場合は，失効要求 $A_i^{(j)}$を生成する：

$$A_i^{(j)} = W_i^{(j)}.$$

一方，（$a_i^{(j)}$，$\delta_i^{(j)}$，$W_i^{(j)}$)の組を得た場合に Alice が $\delta_i^{(n)}$と更新依頼記録 $CR_i^{(j)}$を入力して $n \le j$ であるか $n > j$ であるかを判定するアルゴリズムを更新回数確認アルゴリズム *N_CHECK* とする：

$$N\_CHECK(\delta_i^{(n)}, CR_i^{(j)}) = \begin{cases} 1 & if\ n \le j \\ 0 & if\ n > j \end{cases}.$$

N_CHECK が 1 を出力した場合は *UP_A* を，0 を出力した場合は *DETECT1* を実行後，*UP_A* を実行する．

次に，Alice が警告履歴 $W_i^{(j)}$と $L_{i-1}^{(n)}$を入力し，失効要求 $A_i^{(j)}$を出力するアルゴリズムを失効要求 1 とし，*DETECT*1 と呼ぶ：

$$DETECT1(W_i^{(j)}, L_{i-1}^{(n)}) = A_i^{(j)}.$$

このときの $A_i^{(j)}$を次のように定める：

$$A_i^{(j)} = (W_i^{(j)} \| L_{i-1}^{(n)}).$$

また，Alice が署名生成鍵 SK_{i-1}，$a_i^{(n)}$を入力し，新しい署名生成鍵 SK_i を出力するアルゴリズムを Alice の鍵更新アルゴリズム *UP_A* とする：

$$UP\_A(SK_{i-1}, \alpha_i^{(n)}) = SK_i.$$

この時点で更新依頼 $CR_i^{(j)}$，更新記録 $\delta_i^{(n)}$，更新変数 $a_i^{(n)}$，前の期間における署名生成鍵 SK_{i-1} を消去する．ただし，ラベルは全て別に記録しておく．

(5) *UP*(・)の構成

TTP が Alice によって送られてきた署名付きデータ $SIG_{i-1}(C_i, epk_i^{(j)})$と，Alice の署名検証鍵 *PK* を *VER* に入力して 0 を出力した場合は，*error* として作業を終了する．1 を出力した場合に，TTP が記録していた最新の更新記録 δ_{last}内のラベル L_{last} と Alice の署名のラベル $L_{i-1}^{(n)}$を入力し，$L_{i-1}^{(n)}$と L_{last} が一致するかどうかを判定するアルゴリズムをラベル確認アルゴリズム *L_CHECK* とする：

$$L\_CHECK(L_{i-1}^{(n)}, L_{last}) = \begin{cases} 1 & if\ L_{i-1}^{(n)} == L_{last} \\ 0 & f\ L_{i-1}^{(n)} \ne L_{last} \end{cases}.$$

L_CHECK が 1 を出力した場合は *UP-T* を，2 を出力した場合は *C_CHECK* を実行する．ただし，δ_{last}は次のようになっているとする：

$$\delta_{last} = (C_{last}, epk_{last}, n_{last}, L_{last}).$$

次に，TTP が更新要求と暗号化鍵の組($C_i^{(j)}$，$epk_i^{(j)}$)と更新履歴 D_{last}を入力し，($C_i^{(j)}$，$epk_i^{(j)}$)を含む更新記録が TTP の更新履歴 D_{last}内に存在する場合を *presence* とし，そうでない場合で $epk_i^{(j)}$のみが異なる場合を *absence*1，$C_i^{(j)}$が異なる場合を *absence*2 として判定するアルゴリズムを依頼内容確認アルゴリズム *C_CHECK* とする：

$$C\_CHECK((C_i^{(j)}, epk_i^{(j)}, D_{last}) = \begin{cases} 0 & if\ presence \\ 1 & if\ absence1 \\ 2 & if\ absence2 \end{cases}.$$

C_CHECK が 0 または 2 を出力した場合は *DETECT*3 を，1 を出力した場合は *DETECT*2 を実行してから *UP_T* を実行する．

次に，TTP が更新履歴 D_{last}を入力し，警告 $\omega_i^{(j)}$を出力するアルゴリズムを危殆化検出アルゴリズム 2 の *DETECT*2 とする：

$$DETECT2(D_{last}) = \omega_i^{(j)}.$$

このとき，$\omega_i^{(n)}$を次のように定める：

$$\omega_i^{(j)} = (L_i^{(1)} \| \cdots \| L_i^{(n-1)}).$$

一方，TTP が期間 *i*-1 から *i* への更新に用いる更新鍵 UK_i と更新要求 $C_I^{(j)}$，$C_i^{(j)}$を受信した日時データ $T_I^{(n)}$，暗号化鍵 $epk_i^{(j)}$，更新変数 $a_i^{(n)}$を入力し，更新記録 $\delta_i^{(n)}$を出力するアルゴリズムを TTP の鍵更新アルゴリズム *UP_T* とする：

$$UP\_T(UK_i, C_i^{(j)}, T_i^{(n)} epk_i^{(j)}, \alpha_i^{(n)}) = \delta_i^{(n)}.$$

更新記録 $\delta_i^{(n)}$を以下に示す：

$$\delta_i^{(n)} = (C_i^{(j)}, epk_i^{(j)}, n, L_i^{(n)}).$$

また，TTP は更新履歴 $D_i^{(n)}$を次のように更新する：

$$D_i^{(n)} = (D_{last} \| (\alpha_i^{(n)}, \delta_i^{(n)})).$$

UP_T が実行された状態を，*success* と定義する．

また，TTP が $L_{i-1}^{(n)}$を入力し，警告 $\omega_i^{(j)}$を出力するアルゴリズムを危殆化検出アルゴリズム 3 とし，*DETECT3* と呼ぶ：

$$DETECT3(L_{i-1}^{(n)}) = \omega_i^{(j)}.$$

このとき，$\omega_i^{(j)}$の値を次のように定める．

$$\omega_i^{(j)} = L_{i-1}^{(n)}.$$

TTP は *DETECT2*, *DETECT3* の実行後，警告履歴 $W_i^{(j)}$ を次のように生成する：

$$W_i^{(j)} = (W_{last} \| \omega_i^{(j)}).$$

このとき，*DETECT3* の実行後に警告履歴を生成した状態を *failure* と定義する．

(6) 失効手続き

・Alice が偶然偽造署名を発見したとき

Alice が偶然，偽造署名つきデータ $SIG_i(M)$' を見つけ，そのラベル L_i' より期間 i の署名生成機能が危殆化した事実を知ったとき，偽造署名のラベル L_i' を TTP に渡し，TTP はラベル L_i' を失効リストに載せる．

・Alice が失効要求を生成したとき

失効要求 $A_i^{(j)}$が空でない場合は，*Alice* は $A_i^{(j)}$を TTP に渡し，TTP は $A_i^{(j)}$に含まれる全てのラベルを失効リストに載せ，警告履歴 $W_i^{(j)}$を空にする．

Alice が更新変数を入手できず，失効要求 $A_i^{(j)}$のみを生成した場合は，上記の失効手続きに加えて新たな更新依頼 $C_{i\text{-}new}$を生成して TTP に渡し，新しい更新変数 $a_{i\text{-}new}$とラベル $L_{i\text{-}new}$を得て鍵更新を行う．TTP は，新しい更新記録 $\delta_{i\text{-}new}$ を生成し，更新履歴に更新変数とともに記録する．この作業によって，Alice は新しい *PK* を利用する必要なく鍵更新を続けることができる．

2.3. 実現プロトコルの例

提案方式を実現するために利用できる署名方式の条件は，

・鍵更新に独立の二つの秘密鍵を使用し，TTP を必須とする鍵更新が可能であること

・署名から対応する署名生成鍵を特定できること

である．具体例として，藤崎によって発表された無制限に鍵更新可能な離散対数問題に基づいた Forward-Secure 署名方式[5]を用いて実現する方式を示す．なお，Alice と TTP の間の通信には現在安全であると考えられる暗号化方式を用いるものとし，$(epk_i^{(j)}, esk_i^{(j)})$，$(tpk, tsk)$ はそれぞれその方式における Alice および TTP のそれぞれの鍵とする．

(0) 記号

新たに使用する記号を説明する．

$g_i^{(n)}$：期間 i-1 から i への更新のために生成した n 番目の底

$UK_i^{(n)}$：TTP が Alice の期間 i-1 から i への更新のために n 番目に生成した更新鍵

q：素数

Z_q：0 以上かつ q 未満の整数集合

Z_q^*：$=Z_q-\{0\}$

γ：乱数

$H(\cdot)$：ハッシュ関数

(1) 準備

・鍵生成アルゴリズム：*GEN*

素数 q を選択し，以下のように署名生成鍵 SK_0，更新鍵 $UK_0^{(1)}$，署名検証鍵 *PK*，底 $g_0^{(1)}$（ラベルの初期パラメータ $P_0^{(1)}$に相当）の組を生成する．

$$SK_0 \in Z_q$$

$$UK_0 \in Z_q$$

$$g_0^{(1)} \in Z_q^*$$

$$y = (g_0^{(1)})^{UK_0} \bmod q$$

$$h = (g_0^{(1)})^{SK_0} \bmod q$$

$$PK = (y, h)$$

(2) 署名生成

・署名生成アルゴリズム：*SIG*

期間 i における署名(σ_i, $L_i^{(n)}$)を出力する．ただし，λは署名を生成する都度新たに選択するものとする．

$$\lambda \in Z_q$$

$$X_i = H(PK, L_i^{(n)}, (g_i^{(n)})^{\lambda}, M)$$

$$Y_i = \lambda - X_i SK_i \bmod q$$

$$\sigma_i = (X_i, Y_i)$$

(3) 署名検証

・署名検証アルゴリズム：*VER*

署名検証鍵 *PK*，署名 σ_i= (X_i, Y_i) および L_i=($g_i^{(n)}$, $e_i^{(n)}$, $z_i^{(n)}$, i, $T_I^{(n)}$)を入力し，以下の比較を行う．ただし，$e_i^{(n)}$, $z_i^{(n)}$については後に定義する．ここで，

$$\begin{cases} u = (g_i^{(n)})^{Z_i^{(n)}} y^{e_i^{(n)}} \\ v = (g_i^{(n)})^{Y_i} h^{X_i} \end{cases}$$

とする．

$$H(PK, g_i^{(n)}, u) \stackrel{?}{\equiv} e_i^{(n)}$$

$$H(PK, L_i^{(n)}, v, M) \stackrel{?}{\equiv} X_i$$

双方とも一致すれば 1 を，そうでなければ 0 を出力する．

(4) 鍵更新手続き

TTP は *UP_T* 実行時に，必要に応じて更新履歴内の更新変数を用いて更新要求の署名に含まれるラベル $L_{i-1}^{(n)}$ を生成したときに使用した更新鍵 $UK_{i-1}^{(n)}$ を逆算し，適当な乱数を更新変数 $a_i^{(n)}$ に選んで新たに *Alice* の期間 i-1 から i への n 番目の更新に用いる更新鍵 $UK_i^{(n)}$ を生成し，*UP_T* に入力する．また，*UP_T* の過程において，ラベルを生成する都度新たに選択する乱数 γ を使用して $L_i^{(n)}$ を生成する．

$$UK_i^{(n)} = (\alpha_i^{(n)})^{-1} UK_{i-1}^{(n)}$$

$$g_i^{(n)} = (g_{i-1}^{(n)})^{\alpha_i^{(n)}}$$

$$\gamma \in Z_q^*$$

$$e_i^{(n)} = H(PK, g_i^{(n)}, (g_i^{(n)})^{\gamma})$$

$$z_i^{(n)} = \gamma - e_i^{(n)} UK_i^{(n)} \bmod q$$

$$L_i^{(n)} = (g_i^{(n)}, e_i^{(n)}, z_i^{(n)}, i, T_i^{(n)})$$

一方，*UP_A* については SK_{i-1} および更新変数 $a_i^{(n)}$ を入力として以下のようにして SK_i を出力する：

$$SK_i = (\alpha_i^{(n)})^{-1} SK_{i-1}.$$

3. 提案方式の解析

3.1. 解析

提案方式が 1.3 で述べた要件を満たしているか解析する．

3.1.1. 要件 1 について

Alice の署名を検証した結果有効となるのは改ざんされていない署名つきデータに対して Alice の署名検証鍵で検証した場合だけであり，Bob は TTP の発行するディジタル証明書により Alice の署名検証鍵 *PK* を知っているので，署名機能を持つといえる．

エンティティとして新たに加わった TTP による安全性の低下があっては従来の対策に比べ安全性に劣ってしまうが，今回は TTP からデータが盗まれないこと，TTP が不正を行わないこと，そして Alice と TTP の間の通信から得られるデータは暗号化によって守られていることから，安全性を低下させる要因は特段ないと考える．

3.1.2. 要件 2 について

危殆化のレベルを 3 つに分けて考える．

レベル 1 ：Charlie が署名生成鍵 SK_i を入手し，偽造署名を生成する．

レベル 2 ：Charlie が SK_i を入手し，さらに署名生成鍵を更新して複数期間に渡って偽造署名を生成する．

レベル 3 ：Charlie が Alice の更新手続き中に攻撃して SK_i と Alice が生成した $C_i^{(j)}$ を入手し，さらに署名生成鍵を更新して偽造署名を生成する．

・レベル 1 の検出

Charlie が鍵更新を試みないので，従来通り偽造署名を発見することが危殆化の検出となる．しかし，同時に偽造署名は Charlie が入手した署名生成鍵においてしか生成されないので，偽造署名による被害は限定される．

・レベル 2 の検出

Charlie が鍵更新を試みるとその更新記録が Alice に伝わり，署名生成機能が危殆化した事実を知らせることになる．従って，Alice は更新作業を介して危殆化の事実を知ることが可能であると考える．

・レベル 3 の検出

Charlie が Alice の更新作業中という短時間に更新要求と署名生成鍵を入手することはほとんど困難である．

仮にこの攻撃に成功した場合，通信異常がなければ *DETECT*1 によって危殆化が検出される．しかし，Charlie が鍵更新を行った回数が Alice から TTP への通信が失敗した回数に比べて同数以下であり，Charlie による更新が Alice の同じ更新作業中に行われた場合は検出できない．ただし，一つの期間につき有効なラベルは一つであるので，結局 Charlie が生成できる偽造署名ははじめに入手した署名生成機能によって生成されるものに限られ，新たに更新した偽造署名は全て無効となる．

以上から，提案方式は確かに署名生成機能の危殆化を検出する機能を持ち，かつ偽造署名による被害を限定することが可能であると考えられる．

3.2. 従来の方式との比較

従来の危殆化対策技術をいくつか挙げ，その概要と問題点を述べた後，危殆化対策技術としての効果を提案方式の場合と比較する．

・FSDS[3]

署名生成鍵 *SK* を一方向性関数 *h* を用いて短い期間で更新することで，ある日時 *T* における Alice の署名生成鍵を Charlie に盗まれたとしてもそれを用いて日時 *T* 以前の Alice の電子署名つきデータを生成できないようにする署名方式である．（図 4.1 参照）

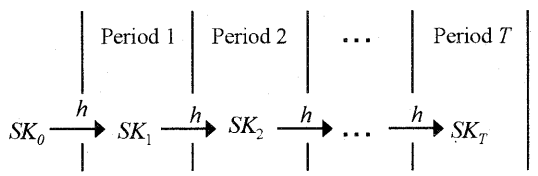

図 4.1　FSSDS の概要

ただし，Charlie が Alice のある期間における署名生

成鍵を知っている場合，それ以降の全ての期間の署名生成機能が危殆化するので，Alice には危殆化した期間を特定することができない．従って，FSDS の目的であった危殆化した期間以前に生成された正当な署名を守ることができない．また，危殆化検出機能を持たない．

・Key-Insulated Signature[8]

署名生成鍵を更新する際に用いるマスターキーを耐タンパーハードウェアに格納することで，ある期間の署名生成鍵が盗まれたとしてもその前後の期間の署名生成機能を守る署名方式である．

問題点としては，署名生成機能の危殆化を検出する機能がないことが挙げられる．

・ヒステリシス署名[1]

署名つきメッセージを生成する際に，それ以前の署名生成履歴を作用させるとともに，署名生成履歴を更新して保管することで，提示された偽造署名を生成していないことを示す署名方式である．従来の署名方式を利用できるという利点がある一方，問題点としてやはり危殆化の検出機能を持たないことが挙げられる．

3.2.1. 危殆化対策技術の効果の比較

提案方式は，危殆化した期間を特定することができる．従って，FSDS が目的としていた危殆化した期間以前に生成された正当な署名を守ることが可能である．また，危殆化した期間よりも後の期間についても，正当な署名が偽造署名と異なる署名生成鍵によって生成されるため，正当な署名を失効させることなく守ることが可能である．

さらに提案方式では，Charlie が鍵更新を試みた場合 Alice が危殆化を検出することが可能であり，そのため失効作業をすばやく行うことで危殆化による被害を抑えることが可能となる．従って，従来の危殆化対策技術に比べ，実際に危殆化が起こってから危殆化を検出するまでの間に発生する被害を抑えることが可能であると考える．

3.2.2. 考察

本方式は，ヒステリシス署名やタイムスタンプなど他のディジタル証拠性技術と併用することで更に効果を高めることが可能である．

Alice がヒステリシス署名を用いた場合，提案方式では不可能だった危殆化した期間に生成された正当な署名と偽造署名の区別が可能となり，また，Alice が自ら危殆化を装うことを防ぐことも可能となる．TTP がヒステリシス署名を用いた場合も，TTP の不正を防ぐ効果がある．

Alice がタイムスタンプを併用した場合においては，ラベル内の日時からタイムスタンプの日時の間に生成された署名であることを示すことができ，本来の署名が生成されたはずの時期から外れた日時に生成された署名を偽造署名として排除することが可能となる．この性質は，近年増加してきた契約書などの日時データを併用するディジタルデータの偽造防止に効果を発揮すると考えられる．

4. おわりに

署名生成機能の危殆化対策技術として，署名生成機能が危殆化した事実を Alice が偶然に頼らずに検知することができる機能を新たに加えたディジタル署名方式を提案し，実現プロトコルの一例を示した．

本方式は時刻情報を利用しているため，Charlie は Alice と全く同じ署名生成鍵を作り出すことはできない．また，危殆化した時期の特定を可能にしているので，危殆化した期間の前後に生成された署名を守ることができる．

さらに，Charlie が TTP を利用して鍵更新をする限り，極めてまれなケースを除き Alice は署名生成機能の危殆化に確実に気づくことができ，偽造署名による被害を抑えることが可能となる．また，本方式は従来の危殆化対策技術であるヒステリシス署名やタイムスタンプを利用することで機能を増すことを示した．本方式の本質は鍵更新に TTP を利用することにあるので，TTP を利用した鍵更新ができる範囲ならば、本方式は他の技術との併用が可能であり，さまざまなニーズに柔軟に対応できる技術であると考えられる．

文　献

[1] 洲崎誠一，松本勉，“電子署名アリバイ実現機構—ヒステリシス署名と履歴交差，”情報処理学会論文誌，Vol.43，No.8，pp.2381-2393，Aug.2002.

[2] 洲崎誠一，松本勉，“電子署名の偽造に関する一考察，”情報処理学会コンピュータセキュリティシンポジウム（CSS2001）論文集，pp.211-216，Aug.2001.

[3] Mihir Bellare，Sara k. Miner，“A Forward-Secure Digital Signature Scheme，” Proc.Crypto 99，pp.431-448，Iul.1999.

[4] 藤崎英一郎，“無制限に鍵更新可能な離散対数問題に基づいた Forward-Secure 署名方式，”進学技報，ISEC2002-75，pp.29-32，Nov.2002.

[5] 岡本龍明，山本博資，“現代暗号，”pp.118-119，産業図書，1997.

[6] 高橋知史，洲崎誠一，松本勉，“「Forward-Secure Digital Signature」は役に立つか，”2002 年暗号とセキュリティシンポジウム予稿集，pp.837-842，電子情報通信学会，Ian.2002.

[7] 安斎潤，松崎なつめ，松本勉，“乱数を用いた複製端末発見方式(１)，”2001 年暗号と情報セキュリティシンポジウム予稿集，SCIS2002-5A-1，2001

[8] Yevgeniy Dodis，Jonathan Katz，Shouhuai Xu，Moti Yung，“Strong Key-Insulated Signature Schemes，” DIMACS Technical Report 2002-25，November 2002