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

2008－DPS－134 (35)
2008－CSEC－ 40 (35)
2008／3／7

暗号技術危殆化に伴う再署名に関する考察

平 井 康 雅[†1] 松尾 真一郎[†1] 尾形 わかは[†2]

現在，契約書等の電子文書に対する捺印行為，改竄防止の実現に，ディジタル署名技術が広く利用されている．これらの署名が施された文書には，法律等により一定期間の保証を求められるものが多数存在し，ディジタル署名技術の安全性についても，同等期間の保証が求められている．しかしながら，新たな攻撃手法の発見によりアルゴリズムが危殆化し，保証期間の終了以前に，その安全性が低下するという問題がある．そのため，既に署名が施された文書の安全性を保証するためには，署名者が，安全なアルゴリズムや鍵を用いて，再署名を施す必要が生じるが，再署名を必要とする文書が大量に存在している等の場合には，署名者の負担が膨大になる．そこで，我々は，アルゴリズムが危殆化した際にも，署名者の再署名処理を無くし，安全性を保証可能な再署名方法について考察を行う．

Consideration of Re-Signing Scheme for Long-term Assurance

Yasumasa Hirai,[†1] Shin'ichiro Matsuo[†1] and Wakaha Ogata[†2]

Digital signature schemes are widely used to prevent from falsifying digital documents such as contract. Some of these documents must be preserved for a long time. Therefore, security of these signature schemes must be provided same-time assurance. However, digital signature algorithm may become insecure by attacks before assurance term end. To keep the security of signatures, signers must re-sign them by secure algorithms and signature keys. However, if a signer has distributed a lot of signed documents, it costs for collecting and re-signing them much. In this study, we consider a secure re-signing scheme which can reduce signer's re-signing cost.

1. はじめに

企業等で扱われる文書には，契約書のような捺印行為を必要とし，その正当性を検証可能とする必要がある文書が多数存在する．また，これらの文書は，法律等により保管期間が定められているなど，一定期間，その正当性が保証をされるべきものが存在する．近年，これらの文書は電子化され，捺印行為や，その文書の正当性を検証する仕組みとして，RSA 署名, DSA といったディジタル署名技術が広く利用されている．そのため，文書が保証されるべき期間に偽造等を防止するために，ディジタル署名技術についても同期間，安全性の保証が求められる．

一般的にディジタル署名アルゴリズムは，ハッシュ関数と公開鍵暗号方式を基とした署名処理との組み合わせにより実現されており，それらの安全性が担保されなくなることにより，ディジタル署名技術の安全性が低下してしまう．これらの安全性が低下する要因の一つとして，新たな攻撃手法の発見による暗号技術の危殆化が挙げらる．特に，2004 年に Wang らによる SHA-1 に対する攻撃[3] 手法が発見されたことにより，ハッシュ関数の危殆化への対策が大きな課題となっている．

これら暗号技術の危殆化は，通常，新たな攻撃手法が発見されてから，署名の偽造等の具体的な不正行為が実行可能となるまでにはある程度の時間を要する．そのため，既に署名が施された文書の正当性を定められた期間保証するためには，署名者が，安全なアルゴリズムや鍵を用いて再署名を行えば良い．しかしながら，再署名が必要となる文書が大量に存在し，分散している場合等には，署名者が文書を回収し，それら全てに対して再署名を施すことは大きな負担となる．そのため，署名者の負担を軽減し，既存の署名の正当性を，文書の有効期間内検証可能な方策が求められる．

そこで，本稿で我々は，ディジタル署名技術で利用されている暗号技術が危殆化した際に，既存の PKI

†1 NTT データ 〒 135-8671 東京都江東区豊洲 3-3-9
NTT DATA Corporation., 3-9, Toyosu 3-chome, Koto-ku, Tokyo 135-8671, JAPAN
†2 東京工業大学 〒 152-8550 東京都目黒区大岡山 2-12-1
Tokyo Institute of Technology., 2-12-1 Ookayama, Meguro-ku, Tokyo, 152-8550, JAPAN

の仕組みを利用し，署名の安全性を定められた期間保証可能とする，署名者の処理を軽減した再署名処理方法について考察を行う．

2. 既存の署名と暗号技術の危殆化

本章では，初めに，既存のディジタル署名の概要について示し，次に，その安全性を低下させる要因となる暗号技術の危殆化について示す．

2.1 既存のディジタル署名の概要

署名者は，まず，電子文書に対してディジタル署名を施すために，署名鍵 (秘密鍵) と，その対となる検証鍵 (公開鍵) を生成し，公開鍵に対して信頼のおける認証局から発行される公開鍵証明書を入手する．これらの鍵対は，利用するディジタル署名アルゴリズムに応じて，十分な鍵長のものが生成される．また，検証鍵を保証するための公開鍵証明書には，有効期間が定められており，鍵の漏洩等により公開鍵証明書が失効されるなどしない限り，公開鍵証明書の有効期間内は対応する署名鍵で署名された文書の正当性を検証することが可能である．なお，以下では，署名を生成するためのディジタル署名技術として，一般的に広く利用されている RSA 署名，DSA 等の Hash-then-Sign 型のディジタル署名技術を想定している．

ディジタル署名の生成，検証の流れを図 1 で示す．

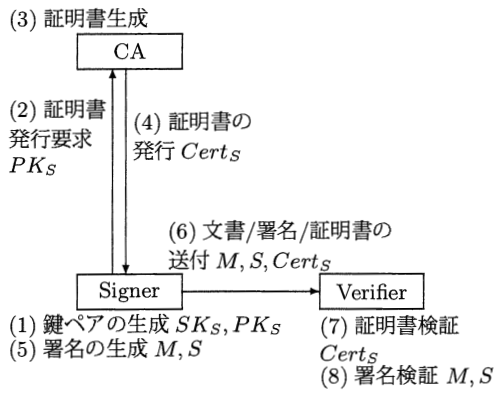

図 1 一般的なディジタル署名の生成と検証

公開鍵証明書の発行 署名者 Signer は，電子文書にディジタル署名を施す際に利用する秘密鍵 (署名鍵)SK_S と検証に用いる公開鍵 (検証鍵)PK_S を生成し，検証鍵に対する公開鍵証明書 $Cert_S$ の発行を，信頼のおける認証局 (CA) に依頼する．一般的に公開鍵証明書では，X.509 等のフォーマットを用い，署名者の公開鍵が含まれた証明書フォーマットに対して，CA が自らの秘密鍵 SK_{CA} とハッシュ関数を用いてディジタル署名を施す方式が用いられる．これら公開鍵証明書の生成の際に CA が用いるディジタル署名においても，署名者の署名処理と同様に，ハッシュ関数等の暗号技術が利用されている．本稿では，公開鍵証明書の発行に用いる暗号技術は十分安全なものを利用することを想定しており，公開鍵証明書の暗号技術の危殆化については対象外とする．

電子文書への署名 署名者 Signer は，ある電子文書 M に対する署名を生成する．まず，ハッシュ関数 $H()$ を用いて M のダイジェスト $H(M)$ を生成する．その後，SK_S を用いて $H(M)$ に対して署名処理を行い，署名 S を生成する．署名者は，電子文書 M および署名 S を，$Cert_S$ と共に検証者に送付する．

証明書検証 受け取った電子文書のディジタル署名の正当性を検証するために，検証者 Verifier は，まず，公開鍵証明書 $Cert_S$ の正当性検証を実行することにより，検証鍵 PK_S の正当性を検証する．公開鍵証明書の正当性検証では，主に以下の 3 点を検証する．

(1) 信頼できる認証局が発行する証明書であるか
(2) 有効期限が切れていないか
(3) 失効していないか

(1),(2) については $Cert_S$ に記載された情報から有効期限の確認および発行元認証局の特定を行う．次に，$Cert_{CA}$ に記載されている PK_{CA} を用いて $Cert_S$ に施された署名を検証することで確認する．また，(3) については $Cert_S$ に関する失効リストが発行されていないか，また，それら失効リストの正当性を検証することにより確認する．なお，$Cert_{CA}$ については予め保持し，信頼しているものとする．

署名検証 $Cert_S$ で保証されている署名者の公開鍵 PK_S を用いて電子文書 M に施されたディジタル署名 S の正当性検証を行う．

2.2 ディジタル署名技術に求められる安全性要件

ディジタル署名技術は，以下の 2 つの要件を満たしている必要がある．

Correctness 正当な署名者が生成した署名は検証で受理される

Unforgeability 正当な署名者以外は，検証で受理される署名を作成できない

2.3 ディジタル署名技術における危殆化

Hash-then-Sign 型のディジタル署名技術は，電子文書のダイジェストを計算するハッシュ関数に関わる部分と，署名処理を実行するための秘密鍵/公開鍵に関わる部分から構成される．そのため，ディジタル署

名技術の安全性に影響を及ぼす暗号技術の危殆化として，ハッシュ関数のアルゴリズム，鍵の危殆化の2点を考慮する必要がある．これら暗号技術の危殆化が進んだ場合，以下のような不正行為が可能となる．

(1) 既存の電子文書のダイジェストと同じダイジェストとなり，署名の値が一致するために正しい署名と判断される電子文書を作成

(2) 正当な署名者以外が，署名者の署名鍵を計算でき，任意の電子文書に対する署名を生成

これらの危殆化が発生する要因としては，計算機能力の向上，アルゴリズムの脆弱性の発見が挙げられる．しかしながら，これらの要因によりディジタル署名技術の安全性が低下する場合，直ちに不正行為が実行可能となるわけではなく，不正行為を実行するために必要となる計算時間が徐々に短くなり，ある時間が経過した後，現実的な時間内で不正行為が実現可能となる．そのため，ある適切な期限までに対策を講じることにより，不正行為の実現を防止することが可能である．

2.4 考えられる危殆化対策と課題

まず，鍵の危殆化等が生じた場合にも，電子文書を長期保証するために，署名時に電子文書に対するタイムスタンプを取得し，偽造を防止するための方策[1),2)]が示されている．この方策では，攻撃者が後に鍵を取得した場合であっても，偽造した電子文書に対する正当なタイムスタンプを取得できないようにすることにより，署名の長期保証を実現している．一方，既存の電子文書においては，タイムスタンプが付与されていないものや，タイムスタンプが電子文書のダイジェストに対してのみ付与されているものがある．このような場合，ハッシュ関数が危殆化すると，署名に含まれるダイジェストと衝突する他のダイジェストを発見可能となるため，攻撃者は検証で受理される偽造署名を生成可能となる．そこで，署名者が新たに安全なアルゴリズムあるいは鍵を用い，既存の電子文書に対して再度署名処理を実施する方策が考えられており，タイムスタンプの長期保証実現方法等に用いられている[1)]．しかしながら，電子文書においては，署名者が，既に配布している署名の回収，全ての文書に対する正当性検証および再署名処理することは大きな負担となる．

そこで，既存の署名に対する暗号技術危殆化後の不正行為を防止し，且つ，署名者の処理を軽減可能な再署名処理方法が求められている．

3. 危殆化による再署名処理方法に関する考察

本章で，2.3節で示したハッシュ関数のアルゴリズム，鍵の危殆化のいずれか一方，あるいは両方が起こった場合の「暗号技術危殆化後の不正行為の防止」を実現し，且つ,「署名者の再署名処理を軽減」を実現可能な再署名処理方法に関する考察を行う．

3.1 再署名処理方法に対する要件

以下に，再署名処理方法が満たすべき要件を示す．

3.1.1 安全性要件

再署名処理が施された文書は，以下の2つの安全性要件を満たしている必要がある．

再署名における Correctness 再署名処理が実施された時点において検証で受理されたならば，再署名処理された署名は，その後も検証で受理される

再署名における Unforgeability 検証で受理される署名は，正当な署名者が生成した署名に対して，再署名処理が定められた期間内に実施されたものに限る

3.1.2 効率に関する要件

我々は，効率的な再署名処理方法を実現する上で，再署名処理方法が以下の要件を満たすべきであると考える．

- 既存の仕組みに存在しない新たな信頼の置ける第三者機関 (TTP) を仮定しない
- 署名者の署名鍵を使用せず，電子文書/署名の組を有するものであれば誰でも実行可能

3.2 再署名処理方法

本節では，再署名処理方法として，予め設定された危殆化対策期間に時刻認証局が発行するタイムスタンプを取得し，既存の署名とそれらを同時に検証することにより，3.1節で示した要件を満たす再署名処理方法について示す．

3.3 再署名処理方法の構成要素

再署名処理方法の構成要素を以下に示す．

3.3.1 プレイヤ

我々は，再署名処理方法を実現する上で，署名処理に関わるプレイヤとして，認証局，署名者，検証者に加え，TTP として時刻認証局を想定する．時刻認証局は，ある電子データがある時刻に存在したことを証明するためのタイムスタンプを発行する機関であり，電子申請等で広く利用されている．再署名処理方法において，時刻認証局は，再署名処理方法の実現のための特別な処理を必要とせず，利用者から送付されたデータに対して，その存在時刻のみを保証するものとする．以下に再署名処理方法のプレイヤを示す．

CA: 公開鍵証明書の発行/失効を行う認証局 (TTP)

TSA: 利用者から送付されたデータに対して，受付時刻が付与されたタイムスタンプを発行する時刻認証局 (TTP)

Signer: 電子文書 M に対して署名鍵を用いて署名を

施す署名者

Verifier: Signer が署名した文書を検証し，保管する検証者

3.3.2 時　　刻

証明書，署名，再署名処理および検証に関わる時刻を以下のように定義する．また，本稿では，アルゴリズムの脆弱性の発見等により危殆化が確認されてから，実際に不正行為が可能となる前までを危殆化対策期間（再署名処理期間）とし，攻撃者がこの期間にはまだ署名の偽造を実施不可能であるとする．時間の定義を図 2 に示す．

$T_{Cert1} \sim T_{Cert2}$: Signer の公開鍵証明書の有効期間

$T_{s1} \sim T_{s2}$: 電子署名が保証されるべき期間（$T_{Cert1} \leq T_{s1} < T_{s2} \leq T_{Cert2}$）

T_1: 脆弱性が発見された（危殆化）時刻

T_2: 不正行為が実現可能となる時刻

$T_1 \sim T_2$: 危殆化対策（再署名処理）が実施される期間

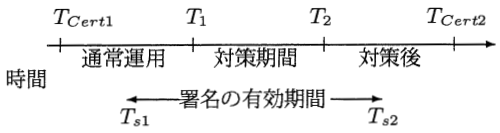

図 2　時間の定義

3.3.3 使用するアルゴリズム

再署名処理方法で使用するアルゴリズムを以下で定義する．我々は，ハッシュ関数の危殆化を想定しているため，時刻 T_2 時点で危殆化するハッシュ関数と，T_2 以降でも安全であるハッシュ関数の 2 種類のハッシュ関数を想定する．

$H_{old}()$: T_1 以前に電子文書に対する署名で用いられるハッシュ関数であり，T_1 で脆弱性が発見されるが，T_2 以前は，不正行為は実行できない (OW, Collision-resistance, second pre-image resistance が担保されている)．T_2 以降は不正行為が実行可能なレベルまで危殆化するため，T_2 以降は使用禁止となる．

$H_{new}()$: 公開鍵証明書，タイムスタンプの生成で用いられるハッシュ関数であり，少なくとも T_{Cert2} までは，安全性が担保されている．$H_{new}()$ は，少なくとも T_{Cert_2} までは安全であるという要件を満たしてさえいれば，プレイヤ毎に異なるハッシュアルゴリズムを用いても良いものとする．

$Sig\langle a\rangle(b)$: ある電子文書 M のダイジェスト b に対して，署名鍵 a を用いて署名処理を実施するためのアルゴリズムであり，署名，証明書，タイムスタンプの生成で用いられる．どの署名に用いられている署名アルゴリズムも，構造上の問題による危殆化は発生しないものとする．

$Verify\langle c\rangle(b, S)$: 検証を実施する電子文書 M のダイジェスト b と，署名 S，検証鍵 c を用いて検証を実施するためのアルゴリズムであり，$true/false$ を出力結果として返す．

3.3.4 鍵と証明書

再署名処理方法で用いる鍵，証明書等について，以下で示す．

SK_{CA}: CA が発行する証明書および失効リストに署名を施すための署名鍵であり，T_2 以降も安全性が担保されていることとする

PK_{CA}: CA が発行する証明書および失効リストの署名を検証するための検証鍵

$Cert_{CA}$: CA の公開鍵証明書であり，Signer および Verifier は予め信頼のおける認証局の証明書であることを認識し，保持しているものとする

SK_{TSA}: TSA が発行するタイムスタンプに用いる署名鍵であり，T_2 以降も安全性が担保されていることとする

PK_{TSA}: TSA が発行するタイムスタンプを検証するための検証鍵

$Cert_{TSA}$: TSA の公開鍵証明書

SK_S: $T_{Cert1} \sim T_2$ まで使用する Signer の署名鍵

PK_S: SK_S に対応する署名検証鍵であり，T_1 で鍵が危殆化する

$Cert_S$: Signer の公開鍵証明書

CRL_S: CA が発行する証明書失効リストであり，$Cert_S$ の失効情報が含まれる．また，$Cert_S$ の失効理由として，ハッシュ関数の危殆化，公開鍵の危殆化（鍵長）が含まれる

3.4 前 提 条 件

ある電子文書 M に対して，$T_{Cert1} \sim T_1$ において，通常の PKI における署名手順に従い，以下の処理が行われていることとする．データフローを図 3 に示す．

署名者 Signer は，署名鍵 SK_S および検証鍵 PK_S を生成し，PK_S に対する証明書発行申請を認証局 CA に対して行う．CA は，申請内容の確認を行い，Signer から受け取った PK_{CA} および申請内容に応じた公開鍵証明書 $Cert_S$ を生成し，署名者に送付する．ここで，CA が公開鍵証明書の生成に利用するハッシュ関数および鍵は，T_1 後も危殆化しないものを利用していることとする．次に，Signer は，電子文書 M に対して，SK_S を用いて署名処理を実行し，署名 $S_1 (= Sig\langle SK_S\rangle(H_{old}(M)))$ を生成する．その後，$(M, S_1, Cert_S)$ を検証者に送付する．

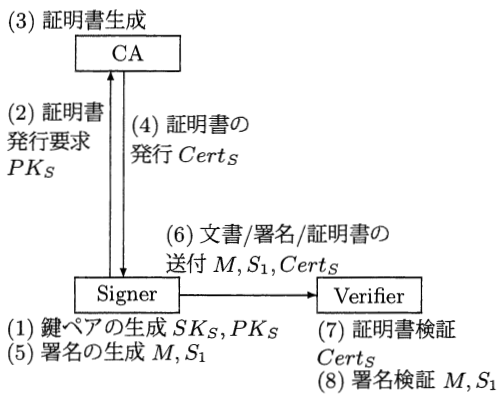

図 3 $T_{Cert1} \sim T_1$ におけるデータフロー

Verifier が時刻 T_V に S_1 を検証する場合，Verifier は，まず署名の有効期間内であるか（$T_{Cert1} < T_{s1} < T_V < T_{s2} < T_{Cert2}$ を満たしているか）を確認する．次に，CA の公開鍵証明書である $Cert_{CA}$ を用いて，$Cert_S$ の正当性を検証する (証明書の正当性検証については 2.1 節参照)．そして，有効期間内であり，証明書の正当性が検証できた場合，$Verify\langle PK_S\rangle(H_{old}(M), S_1)$ を実行し，$true$ であれば電子文書を受理する．

3.5 再署名処理のフロー

各期間での，再署名処理フローについて示す．T_{s2} 以降は，SK_S で施された署名で，正当性検証が必要なものは存在しないこととする．また，T_1 以降に正当な署名者によって新たに生成される署名は，安全な署名鍵およびハッシュ関数を用いて生成されるものとする．

3.5.1 $T_1 \sim T_2$ における再署名処理

暗号技術の危殆化が発生しているが，不正行為は実行できず，実害が生じない期間（対策期間）である $T_1 \sim T_2$ における処理を以下に示す (図 4 参照)．再署名処理を実施するためには，電子文書 M および対応する対策前署名 S_1 を必要とする．M および S_1 を所有するものであれば，再署名処理を実行可能である．ここでは，検証者が再署名処理を実行することとする．

検証者 Verifier は，初めに，Signer の公開鍵証明書 $Cert_S$ の正当性検証を実行する．次に，再署名処理を実施したい電子文書 M, $Cert_S$ に対して，危殆化していないハッシュ関数 $H_{new}()$ を用いてダイジェスト $H_{new}(M)$，$H_{new}(Cert_S)$ を生成する．次に Verifier は，時刻認証局 TSA に対して発行要求データ $data = (H_{new}(M)||S_1||H_{new}(Cert_S))$ に対するタイムスタンプ発行申請を行う．

時刻認証局 TSA は，受け取った申請内容を確認

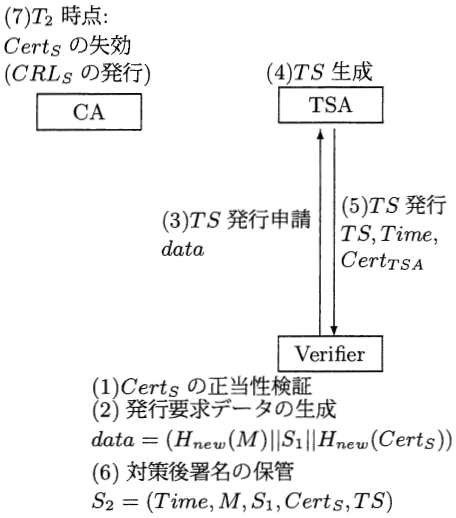

図 4 $T_1 \sim T_2$ におけるデータフロー

し，送付された $data$ に対して申請時の時刻 $Time$ を付与し，自らの署名鍵 SK_{TSA} を用いてタイムスタンプ $TS = Sig\langle SK_{TSA}\rangle(H_{new}(Time||data))$ を生成する．TSA は Verifier に $(TS, Time, Cert_{TSA})$ を送付する．Verifier は，対策後署名として $S_2(= (Time, M, S_1, Cert_S, TS))$ を保管する．

CA は，T_2 時点で，失効理由を鍵の危殆化，あるいは，アルゴリズムの危殆化として $Cert_S$ を失効し，失効リスト CRL_S に追加する．

3.5.2 $T_2 \sim T_{s2}$ における検証処理

T_{s1} 時に使用した鍵あるいはハッシュ関数が危殆化し，不正行為が実施可能となった期間であり，$T_1 \sim T_2$ で再署名処理が実行された対策後署名のみ，検証可能な期間である．検証者 Verifier は，対策前署名 S_1 の署名アルゴリズムを確認し，危殆化しているアルゴリズムが用いられている場合，以下の検証を実施する．$T_2 \sim T_{s2}$ における再署名 $S_2 = (Time, M, S_1, Cert_S, TS))$ の処理を以下に示す (図 5 参照)．

Verifier は，初めに現時刻 T_V が，$T_V < T_{s2} < T_{Cert2}$ を満たすかを確認する．また，再署名処理が，$T_1 \sim T_2$ で実施されているか（$T_1 < Time < T_2$ を満たすか）を確認する．次に，$Cert_{CA}$ を用いて $Cert_S$ および $Cert_{TSA}$ の正当性検証を実施する．また，$Cert_S$ に対する失効リスト CRL_S が T_2 時点において，鍵の危殆化あるいはハッシュ関数の危殆化以外の失効理由で発行されていないことを，CRL_S の内容および正当性検証を実施することにより確認する．その後，対策前署名 S_1 が正しいかどうかを，$Verify\langle PK_S\rangle(H_{old}(M), S_1)$ を用いて検証する．また，$M, S_1, Cert_S, Time$ のデー

Verifier

(1) 現在時刻 T_V と署名有効期間の確認
(2) 再署名処理実施時刻 $Time$の確認
(3) 証明書の正当性検証 $Cert_S, Cert_{TSA}$
ただし $Cert_S$ は失効していることが前提
(4) 失効リスト，失効理由の確認 CRL_S
(5) 対策後署名の検証
$S_2 = TS, M, S_1, Cert_S, Time$
(5-1) 対策前署名の署名検証 S_1
(5-2) タイムスタンプの検証 $TS, Time$
(6) 検証処理 1〜5 を全て満たしている場合，
電子文書/署名を受理

図 5 $T_2 \sim T_{s2}$ における検証処理

タに対して $H_{new}()$，PK_{TSA} を用いて，TS が $Time$ における $(H_{new}(M)||S_1||H_{new}(Cert_S))$ の正当性なタイムスタンプであることを検証する．これらの検証処理が全て満たされれば，署名を受理する．

3.6 再署名方法の安全性に関する考察

本節では，3.2 節で示した再署名方法の安全性に関する考察結果を示す．

攻撃者が，以下を実施可能な場合，攻撃成功とする．

- 攻撃者は，$T_2 \sim T_{s2}$ において検証で受理される，任意のメッセージ M' および，その対策後署名を生成する．

3.6.1 想定する攻撃者

想定する攻撃者の能力を攻撃を実行可能な時刻で分離して以下に示す．

- **T_2 以前 (通常期間，対策期間)** 攻撃者は，署名者が作成したメッセージ M，対応する対策前署名 S_1 および公開情報である PK_S, PK_{CA}, PK_{TSA} を取得可能である．また，任意のメッセージ M' に対する T_2 以前のタイムスタンプを取得可能である．ただし，$H_{old}(M) = H_{old}(M')$ を満たす任意のメッセージ M' を発見すること，署名者秘密鍵 SK_S を取得することはできない．
- **$T_2 \sim T_{S_2}$(対策後)** 攻撃者は，署名者が作成したメッセージ M，対応する対策前署名 S_1，再署名処理時に TSA が発行したタイムスタンプ TS，および公開情報である PK_S, PK_{CA}, PK_{TSA} を取得可能である．$H_{old}(M) = H_{old}(M')$ を満たす任意のメッセージ M' を発見することができ，署名者秘密鍵 SK_S を取得可能である．そのため，任意のメッセージ M' に対する偽造署名 S_1' を生成可能であるとする．$T_2 \sim T_{Cert2}$ におけるタイムスタンプを取得可能であるが，T_2 以前のタイムスタンプは取得できないこととする．

3.6.2 考察結果

以下で，再署名処理方法の安全性要件である，再署名における Unforgeability についての考察結果を示す．

T_2 以前 (通常期間，対策期間) T_2 以前では，攻撃者は，署名者の署名鍵 SK_S を入手できない，また，$H_{old}(M) = H_{old}(M')$ となる M' についても見つけることができない．そのため，攻撃者は，$Verify\langle PK_S\rangle(H_{old}(M'), S_1') = true$ となる (M', S_1') を生成できない．

よって，攻撃者が作成した偽造署名は，図 5 で示す検証処理 (5-1) において $false$ となる．

$T_2 \sim T_{S_2}$(対策後) TS を発行する TSA は TTP であり，申請時の時刻 $Time(T_2 < Time < T_{S_2})$ に対する TS のみを発行する．また，TSA が TS に用いる署名鍵 SK_{TSA} およびハッシュ関数 $H_{new}()$ は危殆化していないため，攻撃者は，署名鍵 SK_{TSA} を入手できない，また，$H_{new}(M) = H_{new}(M')$ となる M' についても見つけることができない．そのため，攻撃者は，T_2 以前の $Time$ に対する TS を生成できない．

よって，攻撃者が作成した偽造署名は，図 5 で示す検証処理 (5-2) において $false$ となる．

このように，各期間で想定する攻撃者が生成した偽造署名を検知することができる．そのため，再署名における Unforgeability は満たされていると言える．

4. ま と め

本稿では，暗号技術の危殆化等による署名技術の安全性が低下した際に，既に署名が施された電子文書の安全性を担保するための，再署名処理方法として，署名者の処理量の増加を考慮し，既存の時刻認証局を利用する再署名処理方式を示し，その安全性に関する考察を行った．今後は，時刻認証局等の TTP を仮定しない方式を検討する．

参 考 文 献

1) 電子商取引推進協議会 認証・公証 WG, “電子署名文書長期保存に関するガイドライン”, 平成 13 年度報告書, 2002 年 3 月.
2) 次世代電子商取引推進協議会 セキュリティWG, “電子文書長期保存ハンドブック”, 平成 18 年度成果報告書, 2007 年 3 月.
3) X. Wang, D. Feng, X. Lai, and H. Yu, “Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD”, IACR Eprint archive 2004/199, Aug. 2004.