RSA 暗号／署名に関する評価報告書

2001 年 12 月 20 日

電気通信大学

太田 和夫

RSA暗号/署名に関する評価報告書

December 20, 2001

Abstract

RSA 法の安全性については文献 [7, 13] を調査した．自己評価書に報告された以上の攻撃法は見つからなかった．よって，RSA Primitive の安全性は，現時点では問題ないと考える．

RSA-PSS の安全性の証明については，文献 [2, 4, 8, 11] を調査した．文献 [11] の証明を詳細に検査し，正しい証明であると判断した[†1]．CRYPTREC に提案された方式 (PSS2000) ではビット列 E に $E = bc$ に加えて，$H_{ID} \parallel cc$ の設定が可能となった．文献 [11] では，この場合を含んだ安全性の検討はされていなかったので，この点に関する安全性の検討は継続する必要がある．

RSA-OAEP の安全性の証明について，文献 [3, 6, 10, 14, 12] を調査した．最新の仕様 (RSA-OAEP01) について，OAEP の安全性の証明を行なった[†2]．

[†1] 証明をフォローするために苦労した点も多々あるので，難解だった箇所の解説も含めて，本文で報告する

[†2] 文献 [10] の RSA-OAEP94 に関する安全性の証明についてフォローできない箇所についても本文にて報告する

Chapter 1

RSA-PSS評価報告書

1.1 はじめに

確率的暗号 (Probabilistic Signature Schemes, PSS) は Bellare と Rogaway によって提案されたデジタル署名の Encoding 技法である．署名対象の文書に乱数成分を付加することで，RSA 署名のように確定的な署名であっても，毎回異なった署名が生成される．確定的な署名法を確率的に変更するだけでなく，RSA-PSS はランダムオラクルモデルのもとで安全性が証明できる [4].

安全性の証明された署名法として，全域ハッシュ法 (Full Domain Hash Schems: FDH) [2] や，本人確認法からデジタル署名を構成する変換法 [9] が提案されているが，PSS は これらの方式に比べてより緊密 (tight) な帰着関係を証明できる特徴を有している．

Corn は FDH について帰着関係を緊密にする技法 (Corn の技法) [8] を提案している．Jonsson は，PSS の安全性を Corn の技法を適用して再評価している [11]．以下では，論文 [11] を検討した結果を報告する．

なお，RSA 法の安全性については文献 [7, 13] を調査した．自己評価書に報告された以上の攻撃法は見つからなかった．よって，RSA Primitive の安全性は，現時点では問題ないと考える．

1.2 記法

PSS のエンコーディングには PSS を提案した論文 [4] でのバージョン (以降では PSS96 とよぶ) と，IEEE P1363a で採用されているバージョン (以降では PSS2000 とよぶ) がある．CRYPTREC 応募の方式は PSS2000 である．

1.2.1 RSA Primitive

公開鍵を (N, e)，秘密鍵を (N, d) とする．ここで，e は 3 以上の奇数で $\mathrm{GCD}\{e, (p-1)(q-1)\} = 1$ をみたし，d は $de \equiv 1(\mathrm{LCM}\{p-1, q-1)\}$ をみたす．

RSA 検証プリミティブ f を

$$f(x) = x^e \bmod N \tag{1.1}$$

で，署名生成プリミティブ f^{-1} を

$$f^{-1}(y) = y^d \bmod N \tag{1.2}$$

で定義する．x と y は $\{0, 1, \ldots, N-1\} = Z_N$ の整数である．$|N| = k$ とする．

1.2.2 PSS Primitive

PSS96　と PSS2000 にどちらも，署名対象の文書 M をエンコードされた文書 y に変換する手順であり，二つの関数 h と g を用いる．g はマスク生成関数とよばれる．

$$h : \{0,1\}^* \longrightarrow \{0,1\}^{k_h}$$
$$g : \{0,1\}^{k_h} \longrightarrow \{0,1\}^{k_g}$$

PSS96

[PSS96-Encode (k_r, k_g, M)**]**

$r \xleftarrow{R} \{0,1\}^{k_r}$;
$w \longleftarrow h(M \parallel r)$;
$r^* \longleftarrow g(w) \oplus (0^{k_g - k_r} \parallel r)$;
Return $y = 0 \parallel r^* \parallel w$.

[PSS96-Verify (k_r, k_g, M, y)**]**

Write $y = b \parallel r^* \parallel w \quad (|b| = 1, |r^*| = k_g \text{ and } |w| = k_h)$;
If $b = 1$, then return 0 and exit.
Write $g(w) \oplus r^* = \gamma \parallel r \quad (|\gamma| = k_g - k_r \text{ and } |r| = k_r)$;
If $h(M \parallel r) = w$ and $\gamma = 0^{k_g - k_r}$, then return 1, else return 0.

RSA-PSS96 の署名作成は文書 M に対して以下の処理をする．RSA-PSS96 の署名検証は署名文 x に対して以下の処理をする．

[RSA-PSS96 (M)]

[RSA-PSS96-Sign (k_r, k_g, M)**]**

$y = \text{PSS96-Encode}(M)$
$x = f^{-1}(y)$

[RSA-PSS96-Verify (k_r, k_g, M, x)**]**

$y = f(x)$
$\text{PSS96-Verify}(M, y)$

注 1.2.1 文献 [4] で使用されているパラメータ k_0 はここでの k_r に対応し，k_1 はここでの k_h に対応している．$k_g = k - k_1 - 1$ となっている．

PSS2000

PSS2000 では $M \parallel r$ にハッシュを施す代わりに，M にハッシュを施した値 H を新たな文書と見なして処理をする．

[PSS2000-Encode (k_r, k_g, H)**]**

$r \xleftarrow{R} \{0,1\}^{k_r}$;
$w \longleftarrow h(0^u \parallel H \parallel r)$;
$r^* \longleftarrow g(w) \oplus (0^{k_g - k_r - 1} \parallel 1 \parallel r)$;
Return $y = 0 \parallel r^* \parallel w \parallel E$.

[PSS2000-Verify (k_r, k_g, H, y)**]**

Write $y = b \parallel r^* \parallel w \parallel E'$ $(|b| = 1, |r^*| = k_g,\ |w| = k_h$ and $|E'| = k_E)$;
If $b = 1$ or $E \neq E'$, then return 0 and exit.
Write $g(w) \oplus r^* = \gamma \parallel r$ $(|\gamma| = k_g - k_r$ and $|r| = k_r)$;
If $h(0^u \parallel H \parallel r) = w$ and $\gamma = 0^{k_g - k_r - 1} \parallel 1$, then return 1, else return 0.

ここで，$E = bc$ あるいは $E = H_{ID} \parallel cc$, $k_E = 8$ あるいは $k_E = 16$ である．

RSA-PSS20000 の署名作成は文書 M に対して以下の処理をする．RSA-PSS2000 の署名検証は署名文 x に対して以下の処理をする．

[RSA-PSS2000 (k_r, k_g, M)**]**

$H = h(M)$

[RSA-PSS2000-Sign (k_r, k_g, H)**]**

$y \longleftarrow$ PSS2000-Encode(H)
$x \longleftarrow f^{-1}(y)$
Return x

[RSA-PSS2000-Verigy (k_r, k_g, H, x)**]**

$y \longleftarrow f(x)$
$b \longleftarrow$ PSS2000-Verify(H, y)
Return b.

1.2.3 全域ハッシュ法 (Full Domain Hash)

この署名法は，ランダムオラクルモデルを用いた証明の有効性を示すものとして文献 [2] で提案され，安全性の証明が与えられた．

[RSA-FHD (M)]

[SignFDH$_{N,d}(M)$**]**

$y \longleftarrow H_{FDH}(M)$;
Return $x = y^d \bmod N$

[VerfyFDH$_{N,e}(M, x)$**]**

$H = h(M)$
$y \longleftarrow x^e \bmod N$; $y' \longleftarrow H_{FDH}(M)$
If $y = y'$, then return 1, else return 0.

定理 1.2.2 (RSA-FDH の安全性) [2]
RSA の逆関数計算が (t', ϵ')-安全とする．RSA-FDH 署名は $(t, q_{sig}, q_{hash}, \epsilon)$-

安全である．ここで，

$$t(k) = t'(k) - [q_{hash}(k) + q_{sig}(k) + 1] \cdot O(k^3)) \tag{1.3}$$

かつ

$$\epsilon(k) = [q_{sig}(k) + q_{hash}(k)] \cdot \epsilon'(k). \tag{1.4}$$

1.2.4 文献 [11] 以前に知られていた安全性

定理 1.2.3 (RSA-PSS の安全性) [4]
RSA の逆関数計算が (t', ϵ')-安全とする．RSA-PSS96 (k_0, k_1) 署名[†1]は $(t, q_{sig}, q_{hash}, \epsilon)$-安全である．ここで，

$$t(k) = t'(k) - [q_{hash}(k) + q_{sig}(k) + 1] \cdot k_0 \cdot \Theta(k^3) \tag{1.5}$$

かつ

$$\epsilon(k) = \epsilon'(k) + [2(q_{sig}(k) + q_{hash})^2 + 1] \cdot (2^{-k_0} + 2^{-k_1}). \tag{1.6}$$

定理 1.2.4 (RSA-FDH の安全性 (改良版)) [8]
RSA の逆関数計算が (t', ϵ')-安全とする．RSA-FDH 署名は $(t, q_{sig}, q_{hash}, \epsilon)$-安全である．ここで，

$$t(k) = t'(k) - [q_{hash}(k) + q_{sig}(k) + 1] \cdot O(k^3) \tag{1.7}$$

かつ

$$\epsilon(k) = \frac{1}{(1 - \frac{1}{q_{sig}+1})^{q_{sig}+1}} \cdot q_{sig}(k) \cdot \epsilon'(k). \tag{1.8}$$

1.2.5 論文 [11] の主張点

論文 [11] では，Corn のアイデアを用いて PSS を評価している．帰着関係を緊密 (tight) に評価したことに加えて，以下を示した．

(1) salt 長を可変にした場合にも安全性の証明を与える．(K_{sig} を署名者が用いる salt 長の全体，K_{ver} を検証者が用いる salt 長の全体として，salt 長を可変にすることを許してモデル化している．)

[†1]パラメータ k_0 は k_r に対応し，k_1 は k_h に対応している．

(2) 関数 h, g が相関がある場合も含めて，帰納関係の効率を評価した．(二つのモデルを設定して，モデル間の帰着関係を与えている (補題 1.3.1)．)

主定理は以下のとおり．

定理 1.2.5 (RSA-GenPSS の安全性)
RSA の逆関数計算が (t', ϵ')-安全とする．RSA-GenPSS (K_{sig}, K_{ver}) 署名は $(t, q_{sig}, q_{hash}, \epsilon)$-安全である．

$$t(k) = t'(k) - \frac{2^{K_E+1}}{1 - q_{tot}2^{-k_h}} \cdot \left(\sqrt{k_h \ln 2} + \sqrt{q_{hash}(k) + 2q_{sig}(k)}\right)^2 \cdot O(T_f(k)) \quad (1.9)$$

かつ

$$\epsilon(k) = \frac{1}{\pi(p, \gamma, q_{sig})} \cdot \epsilon'(k) + c(q_{tot}, k_h) + 2^{-k_h}. \quad (1.10)$$

ここで，p は 0 から 1 の任意の数である．$c(q, l)$ は l ビットの長さの系列から独立に q 個の列を選んだときの衝突が生じる確率であり，以下となる．

$$c(q, l) \leq \binom{q}{2} 2^{-l}. \quad (1.11)$$

α は w が与えられたとき，$g(w)$ とは独立でない $h(y)$ の個数を表す．

$$q_{tot} = max\{\alpha, 1\} \cdot q_{hash} + (\alpha + 2) \cdot q_{sig}, \quad (1.12)$$

と定める．

$$\gamma(q_{hash} + q_{sig}, k_r) = min\{(q_{hash} + q_{sig})2^{-k_r}, 1\}, \quad (1.13)$$

$$\pi(p, \gamma, q_{sig}) = (1 - p)\left(\frac{p}{\gamma \cdot (1 - p) + p}\right)^{q_{sig}} \quad (1.14)$$

とおく．

p の最適値は

$$P_{max} = \frac{2q_{sig}}{q_{sig} + 1 + \sqrt{(q_{sig} - 1)^2 + 4q_{sig}/\gamma}}. \quad (1.15)$$

で与えられる．

注 1.2.6 salt のビット数が大きい場合には，π の値が 1 に近くなり，文献 [4] の結果を与える．salt のビット数が小さい場合には $\pi << 1$ となり，文献 [8] の結果を与える．

文献 [11] で与えられた証明技法は，従来，独立に扱われていた FDH と PSS を，安全性の証明の観点から一つの枠組みで扱うことを可能にしている．

1.3 文献[11]で与えられた定理1.2.5 の証明の検証

1.3.1 セキュリティモデル

二つのモデル,「Reduced モデル」と「通常モデル」を考える．どちらのモデルでも，forger は公開鍵を知っており，署名オラクルと，関数 h, g をシミュレートする二つのハッシュオラクルにアクセスできる．

Reduced モデルでは，二つのハッシュオラクルは独立に動作すると仮定する．g の定義域については両モデルで同じ．通常モデルで h オラクルは任意の長さの入力を受け付けるが，Reduced モデルでは入力が特定の条件をみたすときのみ動作することとする．

通常モデルでは，これら二つのオラクルの間に依存関係を許す．これは，RSA-PSS2000 の仕様で，$g(w)$ を関数 h を用いて構成していることによる．

署名法の一般的な記述

RSA-PSS2000 を一般化して扱うために次の関数を新たに定義する．

$$\varphi : \{0,1\}^{k_h} \times \{0,1\}^K \ni (H, r) \mapsto (0^u \parallel H \parallel r) \in \{0,1\}^*$$
$$\psi : \{0,1\}^K \ni r \mapsto (0^{k_g - k_r - 1} \parallel 1 \parallel r) \in \{0,1\}^{k_g}$$
ここで，$K = K_{sig} \cup K_{ver}$ とする．

以下では，φ と ψ の性質として，単射性，関数値 β が与えられたとき $\beta = \varphi(\alpha), \beta = \psi(\alpha)$ をみたす α が容易に求まることのみを仮定する．

Encoding 技法は次のように一般的に記述できる．RSA-GenPSS では文書 M を直接入力して，RSA-GenPSS-Reduced では文書 M に対して $H = h(M)$ を求めてから入力する．

[GenPSS-Encode (H, k_0)**]**

$r \xleftarrow{R} \{0,1\}^{k_0}$;
$w \longleftarrow h(\varphi(H, r))$;
$r^* \longleftarrow g(w) \oplus \psi(r)$;
Return $y = 0 \parallel r^* \parallel w \parallel E$.

[GenPSS-Verify (H, y, k_0)**]**

Write $y = b \parallel r^* \parallel w \parallel E'$ $(|b| = 1, |r^*| = k_g,\ |w| = k_h$ and $|E'| = k_E)$;
If $b = 1$ or $E \neq E'$, then return 0 and exit;
If possible, write $g(w) \oplus r^* = \psi(r)$ with $|r| = k_0$, Else return 0 and exit;
If $h(\varphi(H, r)) = w$, then return 1 else return 0.

[RSA-GenPSS-Reduced (H, k_0)**]**

[RSA-GenPSS-Reduced-Sign (H, k_0)**]**

$y \longleftarrow$ GenPSS-Encode(H, k_0)
$x \longleftarrow f^{-1}(y)$
Return x

[RSA-GenPSS-Reduced-Verify (H, x, k_0)**]**

$y \longleftarrow f(x)$
$b \longleftarrow$ GenPSS-Verify(H, y, k_0)
Return b.

[**RSA-GenPSS** (H, k_0)]

[**RSA-GenPSS-Sign** (M, k_0)]

$H \longleftarrow h(M)$
$y \longleftarrow$ RSA-GenPSS-Reduced-Sign(H, k_0)
Return x

[**RSA-GenPSS-Verify** (H, x, k_0)]

$H \longleftarrow h(M)$
$b \longleftarrow$ RSA-GenPSS-Reduced-Verify(H, x, k_0)
Return b.

証明の方針

$\mathcal{F}$ を，RSA-GenPSS 署名を確率 ϵ で偽造する Forger とする．$\mathcal{F}_{Fed}$ を，RSA-GenPSS-Reduced 署名を確率 ϵ'' で偽造する Forger とする．ここで，RSA-GenPSS-Reduced では，文書のハッシュ値が署名法の枠外で生成されて，Encoding 処理に引き渡される．

まず，$\mathcal{F}$ を仮定して $\mathcal{F}_{Fed}$ を構成し，次に $\mathcal{F}_{Fed}$ を用いて，RSA 関数の逆関数 (f^{-1}) を計算する Inverter $(\mathcal{I})$ を構成する．二つを組合わせることで，RSA-GenPSS の偽造の問題を RSA の逆関数計算の問題に関連付ける．RSA-GenPSS は RSA-PSS96 と RSA-PSS2000 をモデル化したものであったので，これら二つの方式の安全性を保証できる．

解析にあたっての仮定

(1) 関数 g を実現するために関数 h を利用することを想定して，w が与えられたとき，$g(w)$ とは独立でない $h(y)$ の個数を α とする．一般に，α は w ごとに異なるが，議論を簡単にするためにすべての α に対して一定と仮定する．

(2) $h(y)$ が $g(w)$ に依存するときには，$h(y)$ が $g(w)$ の部分列として現れるものと仮定する．通常，GenPSS では $\alpha = \lceil (k - 1 - k_r - k_E)/k_h \rceil$ が成立つ．

(3) 偽造文で使用される $h(y)$ と $g(w)$ は，y と w が必ず h-oracle query, g-oracle query として現れている．

1.3.2 準備

モデル間の帰着

次の補題が成立つ.

補題 1.3.1 通常モデルで高々 q_{sig} 回の signing query と, h_{hash} 回の h-or g-oracle query で動作する forger $\mathcal{F}$ が与えられると，Reduced モデルで高々 q_{sig} 回の signing query と, $(h_{hash} + q_{sig})$ 回の h- or g-oracle query で動作する forger $\mathcal{F}_{Red}$ を構成できる.

$\mathcal{F}_{Red}$ が偽造に成功するための必要十分条件は，$\mathcal{F}$ が偽造に成功し，かつ $\mathcal{F}$ の h-oracle に対する質問 ($\mathcal{F}$ による signing oracle から発生する h-oracle への質問も含む) を介して得られる値の中で，h-衝突が発生しない場合である.

ここで，h-oracle への質問の回数は，高々

$$q_{tot} = max\{\alpha, 1\} \cdot q_{hash} + (\alpha + 2) \cdot q_{sig}, \tag{1.16}$$

となる.

サンプリング補題

次の補題が成立つ.

補題 1.3.2

$$T = \left\lfloor \frac{N}{|Y|} \cdot \left(\sqrt{k_0 \cdot \ln 2} + \sqrt{q} \right)^2 \right\rfloor \tag{1.17}$$

とおく．T 個の整数をランダムかつ独立に生成する $(z_1, \ldots, z_T \in Z_N)$．このとき，$f(z_1), \ldots, f(z_T)$ の中の q 個未満のサンプルが集合 Y に属する確率は 2^{-k_0} 未満である.

1.3.3 定理 1.2.5 の証明

l ビットの長さの系列から独立に q 個の列を選んだときの衝突発生の確率を, $c(q, l)$ で表す．$\mathcal{F}_{Red}$ が生じる h-衝突の確率は，高々 $c(q_{tot}, k_h)$ である.

Reduced モデルで $\mathcal{F}_{Red}$ が 発行する実際の h-oracle query の個数を q_{Red} と書く． $q_{Red} < q_{hash} + 2q_{sig}$ をみたす.

制約条件「Reduced モデルのもとで h-衝突が生じない」のもとで，$\mathcal{F}_{Red}$ が偽造に成功する確率は，少なくとも

$$\epsilon'' = \frac{\epsilon - c(q_{tot}, k_h)}{1 - c(q_{red}, k_h)} \tag{1.18}$$

となる．

以下，$\mathcal{F}_{Red}$ を用いて RSA-primitive f の逆関数 (f^{-1}) を計算する Inverter $\mathcal{I}$ を構成する．

$\mathcal{I}$ はカウンタ i と，$|H| = k_h$ をみたす H と $k_0 \in K_{sig}$ に対して定めた $R(H, k_0)$ を更新する．i の初期値は 0, $R(H, k_0)$ の初期値は空集合とする．$R(H, k_0)$ は使用されたもののみを管理する．

$\mathcal{I}$ の最終目標は，Z_N^* から一様ランダムに選ばれた整数 η について $\eta^{1/e} \bmod N$ を計算することである．

シミュレーション

[h-oracle のシミュレーション]

入力: ビット列 Q

H1 i を更新する．Q が正しい型，すなわち $\varphi(H_i, r_i)$ で $|H_i| = k_h$ かつ $|r_i| = k_0 \in K_{sig} \cup K_{ver}$ の形をしているなら，以降の処理に進む．他の場合には “error” を戻す

H2 $(H_i, r_i) = (H_j, r_j)$ をみたす j $(j < i)$ が存在するなら，$w_i = w_j$ とおいて step 8 に進む．

H3 $R(H_i, k_0) \longleftarrow R(H_i, k_0) \cup \{r_i\}$ を行なう．

H4 確率 p_{k_0} で $b_i = 0$ を設定し，確率 $(1 - p_{k_0})$ で $b_i = 1$ を設定する．(p_{k_0} の値は後に決める．)

H5 $x_i \xleftarrow{R} Z_N$ を選び，$y_i \longleftarrow \eta^{b_i} \cdot f(x_i)$ を，y_i が $0 \parallel r_i^* \parallel w_i \parallel E \quad (|r_i^*| = k_g$ かつ $|w_i| = k_h)$ をみたし，$w_j = w_i$ となる $j < i$ が存在しないようになるまで繰り返す．

H6 $h(\varphi(H_i, r_i)) = w_i$ と定義する．

H7 $g(w_i) = r_i^* \oplus \psi(r_i) \quad$ と定義する．

H8 w_i を戻す．

[g-oracle のシミュレーション]

入力: ビット列 Q

G1 i を更新して $w_i = Q$ とおく．$|w_i| \neq k_h$ ならば “error.” を戻す．

G2 $j < i$ となるある j に対して $w_i = w_j$ ならば，$g(w_j)$ を戻し，他の場合は $g(w_i) \xleftarrow{R} \{0,1\}^{k_g}$.

[**signer-oracle のシミュレーション**]

入力: ビット列 Q と整数 k_0

S1 i を更新して $H_i = Q$ とおく．$|H_i| \neq k_h$ ならば "error" を戻す. $k_0 \notin K_{sig}$ ならば "error" を戻す.

S2 確率 κ で $c_i = 1$ を設定し，確率 $(1-\kappa)$ で $c_i = 0$ を設定する．(κ の値は後に決める．κ は k_0 と $R(H_i, k_0)$ に依存する．)

S3 $c_i = 0$ ならば $r_i \xleftarrow{R} R(H_i, k_0)$ を実行して step 4 に進む．$c_i = 1$ ならば $r_i \xleftarrow{R} \{0,1\}^{k_0} \setminus R(H_i, k_0)$ を実行して step 5 に進む.

S4 $(H_j, r_j) = (H_i, r_i)$ をみたして，h-oracle への j 番目の質問となっている j $(j < i)$ がある (このことは，S3 での r_i の作り方より，常に成立つ). $b_i = 0$ ならば， $x_i = x_j$ とおいて step 9 へ進む．その他の場合は停止する.

S5 もし $(H_j, r_j) = (H_i, r_i)$ をみたして，j 番目の signing query となっている j $(j < i)$ があるならば，$x_i = x_j$ とおいて step 9 へ進む．その他の場合は次に進む.

S6 $x_i \xleftarrow{R} Z_N$ を選び，$y_i \longleftarrow f(x_i)$ を，y_i が $0 \parallel r_i^* \parallel w_i \parallel E \quad (|r_i^*| = k_g$ かつ $|w_i| = k_h)$ をみたし，$w_j = w_i$ となる $j < i$ が存在しないようになるまで繰り返す.

S7 $h(\varphi(H_i, r_i)) = w_i$ と定義する.

S8 $g(w_i) = r_i^* \oplus \psi(r_i)$ と定義する.

S9 x_i を戻す.

解析

計算時間と成功確率の帰着関係を評価する.

H4 で $b_i = 1$ とすることで複数個の i 成分から $\eta^{1/e} \bmod N$ が求まるようにできる．しかし，この場合には $b_i = 1$ なので signing query の S4 で停止す

ることもある．一方，$b_i = 0$ と設定すると S4 で停止しないが，$\eta^{1/e} \bmod N$ の計算には貢献しない．よって p_{k_0} にはトレードオフがある．

真の署名者が signature query に答えるとき，salt 長 k_0 中の使用される乱数の値は独立であり一様に分布している筈である．もし S2 で k_0 の値に拠らずに salt を一様に生成すると，$r_i \in R(H_i, k_0)$ となる r_i が S4 で停止しない確率は p_{k_0} であり，$r_i \notin R(H_i, k_0)$ となる r_i が S4 で停止しない確率は 1 となる (S3 より，後のケースでは S4 は実行されずに，S5 が実行されるため)．よって，S2 で用いる κ は，$\sigma = \left| R(H_i, k_0) 2^{-k_0} \right|$ とするとその関数として表せて，$r_i \in R(H_i, k_0)$ の発生確率を，$r_i \notin R(H_i, k_0)$ の発生確率の $1/p_{k_0}$ 倍とするように選べば，signature query に対する真の署名者の答えと区別できなくなる．すなわち，

$$\frac{\kappa(\sigma)}{\sigma} = \frac{1-\kappa(\sigma)}{p_{k_0}(1-\sigma)} \Longleftrightarrow \kappa(\sigma) = \frac{\sigma}{p_{k_0}(1-\sigma)+\sigma} \tag{1.19}$$

となる．このとき，Inverter が S4 で停止しない確率は

$$\kappa(\sigma)p_{k_0} + (1-\kappa(\sigma)) \times 1 = \frac{p_{k_0}}{(1-p_{k_0})\sigma + p_{k_0}} \tag{1.20}$$

となる．

$k_0 \notin K_{ver}$ のときは，$p_{k_0} = 1$ とする．なぜなら，Forger は長さ k_0 の salt を用いた偽造署名を出力できないので，$b_i = 1$ としても $\eta^{1/e} \bmod N$ には貢献しないためである．

$p_{k_0}^{max}$ を式 (1.23) を最大にする p とする．(実際には，式 (1.16) で与えられる.)

$$\sigma \leq min\{(q_{hash} + q_{sig})2^{-k_r}, 1\} = \gamma(q_{hash} + q_{sig}, k_r) \tag{1.21}$$

が成立つことに注意すると，

$$\frac{p_{k_0}^{max}}{(1-p_{k_0}^{max})\sigma + p_{k_0}^{max}} \geq \frac{p}{(1-p)\gamma + p} \tag{1.22}$$

がすべての $k_0 \in K_{sig} \cup K_{ver}$ に対して成立つことが示せる．

最後に，$\mathcal{F}_{Fed}$ が偽造文 (H, x) を出力する確率を評価する．このとき，$\mathcal{F}_{Fed}$ の決め方より，$\mathcal{F}$ が (M, x) を偽造しており，$H = h(M)$ をみたす M が存在するのであった．仮定より[†2] $h(\varphi(M, r))$ は知られており，$(H, r) = (H_i, r_i)$

[†2] $\varphi(M, r)$ が h-oracle query に現れないときに，ϵ から $1/2^{k_h}$ を減算することで対応してもよい.

をみたす i がとれる．$\varphi(H_i, r_i)$ は h-oracle query である．なぜなら，もし h-oracle query でなく，signature query になっていて，(H_i, r_i) に対して M' が存在して $H_i = h(M')$ ならば (H, x) は $\mathcal{F}_{Fed}$ の偽造文とはならないからである．

$\varphi(H_i, r_i)$ が h-oracle query なら，確率 $(1-p)$ で $b_i = 1$ となり Inverter は $f^{-1}(\eta)$ を計算できる．「Reduced モデルで h-衝突が存在しない」との制約条件のもとで，Inverter の成功確率は，少なくとも

$$(1-p)\left(\frac{p}{(1-p)\gamma + p}\right)^{q_{sig}} \cdot \epsilon'' = \pi(p, \gamma, q_{sig}) \cdot \epsilon'' \tag{1.23}$$

となる．

S4 での停止の影響は考察済みだが，ループを永久に回し続けるのは不可能なので，H5, S6 の扱いが問題となる．集合 Y を，$0 \parallel r^* \parallel w \parallel E$ の型をしており，w がそれ以前の $j < i$ によって $w = w_j$ とは書けない要素の集合と定義する．すると，事象 Y の発生確率が与えられれば，繰り返し回数と q 個そろえるときの失敗確率が補題 1.3.2 で与えられる．

Z_N からランダムに選んだ値が Y に属する確率は，$2^{-K_E-1} \cdot (1 - q_{tot}2^{-k_h})$ であるので，H5, S6 を

$$\frac{2^{K_E+1}}{1 - q_{tot}2^{-k_h}} \cdot \left(\sqrt{k_h \ln 2} + \sqrt{q_{hash}(k) + 2q_{sig}(k)}\right)^2 \tag{1.24}$$

回繰り返すと，補題によって，q_{sig} 個の signature query と $(q_{sig} + q_{hash})$ 個の hash query が揃わない確率は，2^{-k_h} で抑えられることが分かる．

以上の議論をまとめると，制約条件なしに Inverter が偽造に成功する確率 ϵ は，少なくとも

$$\begin{aligned} \epsilon'(k) &> (1 - c(q_{tot}, k_h)) \cdot \pi(p, \gamma, q_{sig}) \cdot (\epsilon''(k) - 2^{-k_h}) \\ &\geq \pi(p, \gamma, q_{sig}) \cdot (\epsilon(k) - c(q_{tot}, k_h) - 2^{-k_h}) \end{aligned} \tag{1.25}$$

となる．ここで，$(1 - c(q_{tot}, k_h))$ は制約条件を外すためであり，2^{-k_h} は，H5, S6 での失敗確率を減算したものである．

$K_{sig} \cap K_{ver} = \emptyset$ のときには，$k_r \in K_{sig}$ なら $p_{k_r} = 1$ として，$k_r \in K_{ver}$ なら $p_{k_r} = 0$ とおけばよい．このとき，$\pi(p, \gamma, q_{sig}) = 1$ が成立つ．

1.4 残された課題

PSS2000 ではビット列 E に $E = bc$ に加えて，$H_{ID} \parallel cc$ の設定が可能となった．今回内容を検討した文献 [11] では，この場合を含んだ安全性の検討はさ

れていなかった[†3] この点についての検討は継続する必要がある.

このトピックするについて入手可能だった資料 Burt Kaliski "Hash Function Firewalls in Singature Schemes" (IEEE P1363 Working Group Meeting, June 2, 2000) の内容をチェックした.

PSS2000 の「$E = H_{ID} \parallel cc$」の安全性の証明はなかったものの, 他の方式 (ISO/IEC 9796-2, ISO/IEC 14888-2) に比べて, PSS2000 が安全であるとの状況証拠が与えられていた. 報告者にとっては, 納得できるものであった.

RSA 社のセキュリティニュース [†4] にて,

「3－1. 基本 RSA 署名

μ (M) としてハッシュ関数 Hash(M) だけを利用するコアの RSA アルゴリズムによる署名の仕組みについてはご存知の方も多いと思いますが、これは教科書的説明の目的以外に利用されるべきではありません。典型的な Hash 関数のサイズでは先の小素数法等に対して脆弱性がありセキュリティ上大変問題があります。」

との記述が掲載されていた. RSA 社が想定している「教科書的 RSA」の範囲を問い合わせる等の情報収集をしておいたほうがよいと考えます.

[†3] 文献 [11] page 9 下から 12 行目: We mention that this paper does not consider the role of the hash identifiifier, neither does it deal with the isssue of hash substitution attacks in case where there is no hash identifier included in the signature.

[†4] http://www.rsasecurity.com/japan/securitynews/20011011.html

Chapter 2

RSA-OAEP評価報告書

2.1 はじめに

最適非対称暗号パディング技法 (Optimal Asymmetric Encryption Padding, 以降では OAEP) は Bellare と Rogaway によって提案された公開鍵暗号の Encoding 技法である [3].

落し戸つき一方向性関数 f (One-way Trapdoor Permutation: OWTP) が与えられたとき, 最強の安全性 (IND-CCA2) をみたす公開鍵暗号を, f-OAEP によって構成できるとされていた [3]. 一般に, 公開鍵暗号が IND-CC2 をみたすと Non-malleable となることが知られている [1]. Shoup は f が OWTP であっても f-OPEP について malleable な反例を示すことで, 文献 [3] の主張に誤りがあることを指摘した [14].

安全性の証明にあたっては, 攻撃者が decryption oracle に ciphertext query c を質問できるなら, c の平文を知っているはずであるという「平文既知性 (Plaintext Awareness: 以降では PA と略記する)」 が重要である.

文献 [3, 1, 5] では, PA が, 無限の計算能力を持った KE (Knowledge Extractor) を用いてモデル化されていた.

藤崎らは, KE とは異なる多項式時間計算能力をもった Plaintext Extractor を用いることで, f が部分一方向性関数 (Partial-One-way Trapdoor Permutation: POWTP) をみたすならば, f-OPEP が IND-CCA2 であることの安全性の証明を与えた. また, RSA 関数では One-way 性と部分 One-way 性が等価であることが示せるので, RSA-OAEP の安全性が保証できた [10].

ところで, Manger によって指摘された OAEP 実装上の問題点 [12] を克服するために仕様が変更された (**01espdif**[†1]) . この変更が安全性の証明に与

[†1]CRYPTREC に提出された Specification update (**01espdif**) の page 3 line 7 に記載さ

える影響を押さえておくことは，CRYPTREC に応募された RSA-OAPE (以降では，RSA-OAEP01 と略記する) にとって重要と考えて，本報告では，論文 [10] の証明をベースにして，現在の仕様 (**RSA-OAEP01**) に対する証明を行なう．

なお，RSA 法の安全性については文献 [7, 13] を調査した．自己評価書に報告された以上の攻撃法は見つからなかった．よって，RSA Primitive の安全性は，現時点では問題ないと考える．

2.2 記法

OAEP のエンコーディングには OAEP を提案した論文 [3] でのバージョン (以降では OAEP94 とよぶ) と，PKCS #2 v2 で採用されているバージョン (以降では OAEP01 とよぶ) がある．CRYPTREC 応募の方式は RSA-OAEP01 である．

2.2.1 RSA Primitive

公開鍵を (N, e)，秘密鍵を (N, d) とする．ここで，e は 3 以上の奇数で $\mathrm{GCD}\{e, (p-1)(q-1)\} = 1$ をみたし，d は $de \equiv 1(\mathrm{LCM}\{p-1, q-1)\}$ をみたす．

RSA 暗号化プリミティブ f を

$$f(x) = x^e \bmod N \tag{2.1}$$

で，復号プリミティブ f^{-1} を

$$f^{-1}(y) = y^d \bmod N \tag{2.2}$$

で定義する．x と y は $\{0, 1, \ldots, N-1\} = Z_N$ の整数である．$|N| = k$ とする．

2.2.2 OAEP 変換

OAEP94 と OAEP01 のどちらも，暗号化対象の通信文 M をエンコードされた通信文 (EM) y に変換する手順であり，二つの関数 G と H を用いる．これらはマスク生成関数とよばれる．

れている「dbMask=MGF(seed,emLen-hLen)」は,「dbMask=MGF(seed,emLen-hLen-1)」が正しいと思われる．page 4 line 1 について同様．(確認要)

OAEP01

RSA-OAEP01 は以下の関数を用いて構成する.

$$G : \{0,1\}^{k_0} \longrightarrow \{0,1\}^{k-k_0-8}$$
$$H : \{0,1\}^{k-k_0-8} \longrightarrow \{0,1\}^{k_0}$$

注 2.2.1 文献[3](OAEP94) ではビット単位の表記が使われており，OAEP01 ではバイト単位の表記が使われている．OAEP の安全性の証明では，ビット単位の表記を用いることが通常なので，以下では，ビット単位の表記を用いる.

[OAEP01-Encode $(M, pHash, k, k_0)$**]**

$r \xleftarrow{R} \{0,1\}^{k_0}$;
$s \longleftarrow G(r) \oplus pHash \parallel 0^{k-n-2k_0-8\times 2} \parallel 01 \parallel M$
where $n = |M|$, that is, n is the bit length of M[a];
$t \longleftarrow r \oplus H(s)$;
Return $y = 0^8 \parallel t \parallel s$.

[a]文献 [10] と応募書類のデータ表記法が異なるため，安全性の証明に影響のないので，見やすい表記を採用した.

[OAEP01-Verify $(y, pHash, k, k_0)$**]**

Write $y = X \parallel t \parallel s \quad (|X| = 8, |t| = k_0$ and $|s| = k - 8 - k_0)$;
$r \longleftarrow t \oplus H(s)$;
Write $G(r) \oplus s = pHash' \parallel \alpha \parallel T \parallel M$
$(|pHash'| = k_0, \alpha = 0^{(8\text{の倍数})}, |T| = 8$ かつ $T \neq 0^8)$;
If $pHash' \neq pHash$, if $X \neq 0^8$, or if $T \neq 0^7 1$, then output "decoding error." Else output M.

OAEP94

OAEP94 は以下の関数を用いて構成する.

$$G : \{0,1\}^{k_0} \longrightarrow \{0,1\}^{k-k_0}$$
$$H : \{0,1\}^{k-k_0} \longrightarrow \{0,1\}^{k_0}$$

[OAEP94-Encode (M, k, k_0)**]**

$r \xleftarrow{R} \{0,1\}^{k_0}$;
$s \longleftarrow G(r) \oplus M \parallel 0^{k_1}$;
$t \longleftarrow r \oplus H(s)$;
Return $y = t \parallel s$.

[OAEP94-Verify (y, k, k_0)**]**

Write $y = t \parallel s \quad (|t| = k_0$ and $|s| = k - k_0)$;
$r \longleftarrow t \oplus H(s)$;
Write $G(r) \oplus s = \alpha \parallel M \quad (|\alpha| = k_1)$;
If $\alpha \neq 0^{k_1}$, then output "decoding error." Else output M.

2.2.3 RSA-OAEP

RSA-OAEP01

RSA-OAEP01 の暗号化は通信文 M に対して以下の処理をする. RSA-OAEP01 の復号は暗号文 x に対して以下の処理をする.

[RSA-OAEP01 (M, P, k, k_0)**]**

$pHash = Hash(P) \in \{0,1\}^{k_0}$

[RSA-OAEP01 $(M, pHash, k, k_0)$**]**

$y =$ OAEP01-Encode$(M, pHash, k, k_0)$
$x = f(y)$

[RSA-OAEP01-Verify $(x, pHash, k, k_0)$**]**

$y = f^{-1}(x)$
OAEP01-Verify$(y, pHash, k, k_0)$

RSA-OAEP94

RSA-OAEP94 の暗号化は通信文 M に対して以下の処理をする．RSA-OAEP94 の復号は暗号文 x に対して以下の処理をする．

[RSA-OAEP94 (M, k, k_0)**]**

[RSA-OAEP94 (M, k, k_0)**]**

$y = \text{OAEP94-Encode}(M, k, k_0)$
$x = f(y)$

[RSA-OAEP94-Verify (x, k, k_0)**]**

$y = f^{-1}(x)$
$\text{OAEP94-Verify}(y, k, k_0)$

2.3 RSA-OAEP94 の安全性

藤崎らは，文献 [1] でモデル化した KE とは異なるモデル化 (KE の代わりに，多項式時間計算能力をもった Plaintext Extractor) を用いることで，次の帰着関係を証明した．

定理 2.3.1 (RSA-OAEP94 の安全性) [10]
実行時間が t 以下で decryption oracle への質問回数が q_D 以下，ハッシュ関数 G への質問回数が q_G 以下，ハッシュ関数 H への質問回数が q_H 以下の OAEP 変換 $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ の"semantic security" を破る CCA-adversary $\mathcal{A}$ が与えられたとき，$\mathcal{A}$ の advantage, ϵ, は

$$\epsilon(k) > 2 \times \frac{\mathtt{Succ}^{\mathrm{OW}}(t') + q_H \times \mathtt{Succ}^{\mathrm{p\text{-}OW}}(t') + p_D \times (\frac{1}{2^{k_1-1}} + \frac{2q_G+1}{2^{k_0}}) + \frac{q_G}{2^{k-1}}}{1 - \frac{q_G}{2^{k_0}} - \frac{q_H}{2^{k-k_0}}} \quad (2.3)$$

ただし，

$$t'(k) = t(k) + q_G(k)q_H(k) \cdot (T_f + O(1)) \quad (2.4)$$

ここで，T_f は関数 f の計算時間を表す．

2.3.1 記号

証明で用いる記号を定義する．復号対象の暗号文を $c^\star$ と書いて，$c^\star = f(s^\star, t^\star)$ と表す．

G-List: G への質問 γ とその答え G_γ の組の履歴．
H-List: H への質問 δ とその答え H_δ の組の履歴．
AskG: $r^\star$ を G に質問する事象．
AskH: $s^\star$ を H に質問する事象．
FAskH: $s^\star$ を A_1 が H に質問する事象．
SBad: $s = s^\star$ が成立する事象．
RBad: $r = r^\star$ が成立する事象．このとき

$$H(s) \oplus t = H(s^\star) \oplus t^\star \tag{2.5}$$

が成立つ．
Bad: Bad = RBad $\vee$ SBad
AskR: r を G に質問する事象．
AskS: s を H に質問する事象．
AskRS: AskRS = AskR $\wedge$ AskS
FBad: A_1 が $r^\star$ を G に質問し，かつ

$$G_{r^\star} \neq s^\star \oplus (m_i \parallel 0^{k_1}) \qquad (i = 0, 1) \tag{2.6}$$

が成立する事象．
GBad: A_2 が $r^\star$ を G に質問し，かつ

$$G_{r^\star} \neq s^\star \oplus (m_i \parallel 0^{k_1}) \qquad (i = 0, 1) \tag{2.7}$$

が成立する事象．
Fail: Plaintext Extractor が誤った復号結果を出力する事象．

2.3.2 Simulator $\mathcal{B}$ の構成

CCA Adversary として $\mathcal{A}$ が与えられたとき，暗号文 $c^\star$ を復号する $\mathcal{B}$ を構成する．

$\mathcal{B}$ は，$\mathcal{A}$ が真の decryption oracle とやりとりを行なっているのか，シミュレータとやりとりをしているのかを区別出来ない環境を $\mathcal{A}$ に提供することで，$\mathcal{A}$ から semantic security を破る能力を引き出して，その出力を用いて最終的に $c^\star$ を復号する．$\mathcal{B}$ は，decryption oracle をシミュレートする decryption simulator $\mathcal{DS}$ と，$(s^\star, t^\star)$ を求める Inverter $\mathcal{I}$ を一部に取り込んだ構成になっている．

ここでは，まず $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ の semantic security を破る CCA2 Adversary $\mathcal{A} = (A_1, A_2)$ を考える．

B1 $\mathcal{K}(1^k)$ を走らせて関数 f を入手する．

B2 $(s^\star, t^\star) \xleftarrow{R} \{0,1\}^{k-k_0} \times \{0,1\}^{k_0}$ で生成した $(s^\star, t^\star)$ から，$c^\star = f(s^\star, t^\star)$ で生成した $c^\star$ を入手する．

B3 生成した公開鍵 (関数 f) で A_1 を動作させて平文の組 $\{m_0, m_1\}$ とビット列 st を入手する．乱数ビット b を生成して，$c^\star$ を m_b の暗号文として出力する．

B4 $A_2(c^\star, st)$ を動作させて，decryption oracle への ciphertext query c を入手すると，以下に述べる decryption simulator を動かして対応する平文を求めて，A_2 に通知する．

B5 $A_2(c^\star, st)$ を動作させて b' を入手する[†2]と，G と H への質問リスト (G-List, H-List) の中から $x = (s^\star, t^\star)$ を以下に示す Inverter 手順で見つけて，$c^\star$ の逆像として出力する．

Decryption Simulator の構成

記号を定義する．

$[\mu]_{k_1}$: μ の下位 k_1 ビットを抽出したもの．
$[\mu]^n$: μ の上位 n ビットを抽出したもの．

[Decryption Simulator ($\mathcal{DS}$)]

入力: $c = f(s, t)$

[†2] semantic security を破る Adversary $\mathcal{A}$ の能力は，$b = b'$ を成立たせることである．

DS1 $(\gamma, G_\gamma) \in$ G-List, $(\delta, H_\delta) \in$ H-List のすべての組に対して,

$$\sigma = \delta, \tau = \gamma \oplus H_\delta, \mu = G_\gamma \oplus \delta \tag{2.8}$$

とおき,

$$c = f(\sigma, \tau) \text{ かつ } [\mu]_{k_1} = 0^{k_1} \tag{2.9}$$

が成立つかを調べる.

DS2 いずれかの組で二つの関係式が同時に成立するなら, $[\mu]^n$ を出力する. 成立しないなら"Reject" を出力する.

Inverter の構成

[B5] で用いる Inverter の実現方法は以下のとおり.

[**Inverter** $(\mathcal{I})$]

γ に対して:

1: すべての $(\delta, H_\delta) \in$ H-List に対して

1-1: $z = \gamma \oplus H_\delta$

1-2: $c^\star = f(\delta, z)$ を検査する.

2: いずれかの γ で合格なら $G_\gamma = \delta \oplus (m_b \parallel 0^{k_1})$ と定義し, (δ, z) を $x = (s^\star, t^\star)$ とする. すべてで不合格なら $G_\gamma \xleftarrow{R} \{0,1\}^{k-k_0}$ と定義する.

3: G-List $\xleftarrow{R}$ G-List $\parallel (\gamma, G_\gamma)$.

δ に対して:

1: $H_\delta \xleftarrow{R} \{0,1\}^{k_0}$ と定義する.

2: H-List $\xleftarrow{R}$ H-List $\parallel (\delta, H_\delta)$.

3: すべての $(\gamma, G_\gamma) \in$ G-List に対して

3-1: $z = \gamma \oplus H_\delta$

3-2: $c^\star = f(\delta, z)$ を検査する.

4: いずれかの γ で合格すれば, (δ, z) を $x = (s^\star, t^\star)$ とする.

2.3.3 補題 [10]

定理 2.3.1 を証明するために，文献 [10] で準備された補題は以下のとおり．

$\mathcal{DS}$ を用いると CCA Adversary $\mathcal{A}$ を CPA Adversary に変換できるので，ここでは $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ の semantic security を破る CPA Adversary $\mathcal{A} = (A_1, A_2)$ が与えられたとして議論する．

補題 2.3.2 (RSA-OAEP94 の安全性 (CPA-adversary))
実行時間が t 以内, advantage が ϵ で, ハッシュ関数 G への質問回数が q_G 以下, ハッシュ関数 H への質問回数が q_H 以下で f-OAEP94 $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ の"semantic security" を破る CPA-adversary $\mathcal{A}$ が与えられたとき，置換 f の一方向性を破る adversary $\mathcal{B}$ の成功確率 ϵ' と実行時間 t' は

$$\epsilon'(k) > \frac{\epsilon}{2} \times (1 - \frac{q_G}{2^{k_0}} - \frac{q_H}{2^{k-k_0}}) - \frac{q_G}{2^{k-1}} \tag{2.10}$$

$$t'(k) \leq t(k) + q_G(k)q_H(k) \cdot (T_f + O(1)) \tag{2.11}$$

となる．ここで，T_f は関数 f の計算時間を表す.

補題 2.3.3 (Decryption Simulation の成功確率)
Decryption simulator $\mathcal{DS}$ が，encryption oracle から暗号文 $c^\star$ を高々一つ受信しているとき，decrypt query c の正しい答えを求める実行時間を t'，成功確率を ϵ' とすると，

$$\epsilon'(k) > 1 - \left(\mathtt{Succ}^{\text{sp-ow}}(t') + \frac{1}{2^{k_1-1}} + \frac{2q_G + 1}{2^{k_0}}\right) \tag{2.12}$$

$$t'(k) \leq q_G(k)q_H(k) \cdot (T_f + O(1)) \tag{2.13}$$

となる.

注 2.3.4 文献 [3, 1] で用いられている無限の計算能力を持った KE (Knowledge Extractor) を仮定すると，$\mathtt{Succ}^{\text{sp-ow}}(t') = 1$ なので，$\epsilon'(k) \geq 0$ という自明な結果しか得られない.

補題 2.3.5 (Decryption Simulation の成功確率 ($\neg$AskH))
Decryption simulator $\mathcal{DS}$ が，encryption oracle から暗号文 $c^\star$ を高々1 つ受信しており，$s^\star$ が H に質問されていないなら，decrypt query c の正しい答えを求める実行時間を t'，成功確率を ϵ' とすると，

$$\epsilon'(k) > 1 - (\frac{1}{2^{k_1-1}} + \frac{2q_G + 1}{2^{k_0}}) \tag{2.14}$$

$$t'(k) \leq q_G(k)q_H(k) \cdot (T_f + O(1)) \tag{2.15}$$

となる.

注 2.3.6 この結果は，文献 [3, 1] で用いられている無限の計算能力を持った KE (Knowledge Extractor) ，文献 [10] で用いられている有限の計算能力を持った PE (Plaintext Extractor) のどちらにも成立つ.

2.3.4 定理 2.3.1 の証明

$\neg\mathtt{AskH}$ の場合に q_D 個の decryption oracle へのすべての質問に正しく回答できる確率は，少なくとも

$$\left(1-\left(\frac{1}{2^{k_1-1}}+\frac{2q_G+1}{2^{k_0}}\right)\right)^{q_D} \tag{2.16}$$

となる．この値は

$$1-q_D\times\left(\frac{1}{2^{k_1-1}}+\frac{2q_G+1}{2^{k_0}}\right) \tag{2.17}$$

で下からおさえられる．したがって，少なくとも一つの decryption simulation に失敗する確率は

$$\begin{aligned}
&\mathbf{pr}[\text{one Fail}]\\
&\quad = \mathbf{pr}[\text{one Fail}\,|\,\neg\mathtt{AskH}]\times\mathbf{pr}[\neg\mathtt{AskH}]+\mathbf{pr}[\text{one Fail}\,|\,\mathtt{AskH}]\times\mathbf{pr}[\mathtt{AskH}]\\
&\quad \le \mathbf{pr}[\text{one Fail}\,|\,\neg\mathtt{AskH}]+\mathbf{pr}[\mathtt{AskH}]\\
&\quad \le q_D\times\left(\frac{1}{2^{k_1-1}}+\frac{2q_G+1}{2^{k_0}}\right)+\mathbf{pr}[\mathtt{AskH}]
\end{aligned} \tag{2.18}$$

となる．$\mathbf{pr}[\mathtt{AskH}]$ は $\mathtt{Succ}^{\text{sp-ow}}(q_H,t')$ でおさえられる.

simulator $(\mathcal{B})$ は，q_D 個の query を $\mathcal{DS}$ で構成した plaintext extractor に投げて対応する平文を入手し，Adversary に答えることが可能となり，Adversary が b' を答えるのを契機として，Inverter の作成した x を出力する.

このとき，$(\mathcal{B})$ が $c^\star$ を復号できる確率 ϵ' は

$$\begin{aligned}
\epsilon'(t'(k)) \;\ge\; &\left\{\frac{\epsilon}{2}\times\left(1-\frac{q_G}{2^{k_0}}-\frac{q_H}{2^{k-k_0}}\right)-\frac{q_G}{2^{k-1}}\right\}\\
&-\left\{q_D\times\left(\frac{1}{2^{k_1-1}}+\frac{2q_G+1}{2^{k_0}}\right)+\mathtt{Succ}^{\text{sp-ow}}(q_H,t')\right\}.
\end{aligned} \tag{2.19}$$

となる．右辺の第 1 項は補題 2.3.2 で与えられ，右辺の第 2 項は式 (2.18) で与えられる.

また，$\mathcal{DS}$ の実行時間は,

$$t'(k)\le q_G(k)q_H(k)\cdot(T_f+O(1)) \tag{2.20}$$

におさまる.

ところで, $\mathtt{Succ}^{\mathrm{OW}}(\tau)$ の定義に注意すると, $\mathtt{Succ}^{\mathrm{OW}}(t'(k)) \geq \epsilon'(t'(k))$, かつ $\mathtt{Succ}^{\mathrm{sp\text{-}ow}}(q_H, t'(k)) = q_H \mathtt{Succ}^{\mathrm{p\text{-}ow}}(t'(k))$ が成立つので,

$$\epsilon(k) > 2 \times \frac{\mathtt{Succ}^{\mathrm{OW}}(t') + q_H \times \mathtt{Succ}^{\mathrm{p\text{-}ow}}(t') + p_D \times (\frac{1}{2^{k_1-1}} + \frac{2q_G+1}{2^{k_0}}) + \frac{q_G}{2^{k-1}}}{1 - \frac{q_G}{2^{k_0}} - \frac{q_H}{2^{k-k_0}}}$$

ただし,

$$t'(k) = t(k) + q_G(k)q_H(k) \cdot (T_f + O(1))$$

が示せた.

2.4 RSA-OAEP01 の安全性

文献 [10] に従って, RSA-OAEP01 の安全性を評価する.

証明は, Decryption Simulator の構成と, CPA Adversary に対する安全性の評価の 2 点の検討が必要になる.

Decryption Simulator は, s 成分の具体的な型には依存せずに構成されているので, 仕様の変更による影響はないと考えられる.

よって, 以降では補題 2.3.2 の見直しを行なう.

2.4.1 仕様の違い

[OAEP01-Encode $(M, pHash, k, k_0)$**]**

$r \xleftarrow{R} \{0,1\}^{k_0}$;
$s \longleftarrow G(r) \oplus pHash \parallel 0^{k-n-2k_0-8\times 2} \parallel 01 \parallel M$

[OAEP94-Encode (M, k, k_0)**]**

$r \xleftarrow{R} \{0,1\}^{k_0}$;
$s \longleftarrow G(r) \oplus M \parallel 0^{k_1}$;

2.4.2 補題 2.3.2の見直し

記号の変更など

RSA-OAEP01 の安全性の証明で用いる以下の記号の定義を変更する.

FBad: A_1 が $r^\star$ を G に質問し，かつ

$$G_{r^\star} \neq s^\star \oplus (pHash \parallel 0\cdots001 \parallel m_i) \qquad (i=0,1) \tag{2.21}$$

が成立する事象.

GBad: A_2 が $r^\star$ を G に質問し，かつ

$$G_{r^\star} \neq s^\star \oplus (pHash \parallel 0\cdots001 \parallel m_i) \qquad (i=0,1) \tag{2.22}$$

が成立する事象.

OAEP94 では $k_1 = (k - k_0 - n)$ となっており，OAEP01 では $k_1 = (k - k_0 - n - 8)$ となっている.

$\mathcal{DS}$ の定義を以下のように変更する.

[Decryption Simulator ($\mathcal{DS}$)]

入力: $c = f(s,t)$

DS1 $(\gamma, G_\gamma) \in$ G-List, $(\delta, H_\delta) \in$ H-List のすべての組に対して,

$$\sigma = \delta, \tau = \gamma \oplus H_\delta, \mu = G_\gamma \oplus \delta \tag{2.23}$$

とおき,

$$c = f(\sigma,\tau) \text{ かつ } [\mu]^{k_1} = pHash \parallel 0\cdots001 \tag{2.24}$$

が成立つかを調べる.

DS2 いずれかの組で二つの関係式が同時に成立するなら，$[\mu]_n$ を出力する. 成立しないなら"Reject" を出力する.

$\mathcal{I}$ の構成法の変更点は, [B5] の実現方法のみであり，以下のとおり.

2: いずれかの γ で合格なら $G_\gamma = \delta \oplus (pHash \parallel 0\cdots001 \parallel m_b)$ と定義し，(δ, z) を $x = (s^\star, t^\star)$ として出力する. すべてで不合格なら $G_\gamma \xleftarrow{R} \{0,1\}^{k-k_0-8}$ と定義する.

2.4.3 補題 2.4.1 の証明

補題 2.4.1 (RSA-OAEP01 の安全性 (CPA-adversary))
実行時間が t 以下, advantage が ϵ で, ハッシュ関数 G への質問回数が q_G 以下, ハッシュ関数 H への質問回数が q_H 以下の f-OAEP01 $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ の"semantic security" を破る CPA-adversary $\mathcal{A}$ が与えられたとき, 置換 f の一方向性を破る adversary $\mathcal{B}$ の成功確率 ϵ' と実行時間 t' は

$$\epsilon'(k) > \frac{\widetilde{\epsilon}}{2} \times (1 - \frac{q_G}{2^{k_0}} - \frac{q_H}{2^{k-k_0-8}}) - \frac{q_G}{2^{k-1-k_0}}$$
$$t'(k) \leq t(k) + q_G(k) q_H(k) \cdot (T_f + O(1))$$

ここで, T_f は関数 f の計算時間を表し,
$\widetilde{\epsilon(k)} = \epsilon(k) - 4\Pr[\neg\mathtt{Bad}] \leq \epsilon - \frac{q_G}{2^{k_0}-2} - \frac{q_H}{2^{k-k_0-10}}$ とする.

証明:
$\Pr[\cdot]$ で decryption oracle を用いた場合の成功確率, $\mathtt{pr}[\cdot]$ で decryption simulator を用いた場合の成功確率を表す.

$$\epsilon \leq \mathrm{Adv}^{\mathrm{ind}}(\mathcal{A}) \leq 2 \times \Pr[\mathtt{AskG} \wedge \mathtt{AskH}] + 2 \times \Pr[\mathtt{AskG} \wedge \neg\mathtt{AskH}]$$

が成立つ.

ここで, $\neg\mathtt{Bad}$ のとき, $\mathcal{A}$ には decryption oracle と $\mathcal{DS}$ の区別がつかないので, $\Pr[A \,|\, \neg\mathtt{Bad}] = \mathtt{pr}[A \,|\, \neg\mathtt{Bad}]$ となる.

一般に,

$$\begin{aligned}
\Pr[\mathtt{A}] &= \Pr[\mathtt{A} \wedge \mathtt{B}] + \Pr[\mathtt{A} \wedge \neg\mathtt{B}] \\
&= \Pr[\mathtt{A} \,|\, \mathtt{B}] \times \Pr[\mathtt{B}] + \Pr[\mathtt{A} \,|\, \neg\mathtt{B}] \times \Pr[\neg\mathtt{B}] \\
&\leq \Pr[\mathtt{B}] + \Pr[\mathtt{A} \,|\, \neg\mathtt{B}] \\
&= \Pr[\mathtt{B}] + \mathtt{pr}[\mathtt{A} \,|\, \neg\mathtt{B}]
\end{aligned}$$

が成立つ.

これより,

$$\epsilon \leq 2 \times (\mathtt{pr}[\mathtt{AskG} \wedge \mathtt{AskH} \,|\, \neg\mathtt{Bad}] + \mathtt{pr}[\mathtt{AskG} \wedge \neg\mathtt{AskH} \,|\, \neg\mathtt{Bad}]) + 4\Pr[\mathtt{Bad}]$$

をえる. これ以降, $\tilde{\epsilon} = \epsilon - 4 \times \Pr[\neg\mathtt{Bad}]$ とおく[†3].

[†3] 文献 [10] では同じ議論が行なわれている.

$$\epsilon \leq 2 \times (\mathtt{pr}[\mathtt{AskG} \wedge \mathtt{AskH} \,|\, \neg\mathtt{Bad}] + \mathtt{pr}[\mathtt{AskG} \wedge \neg\mathtt{AskH} \,|\, \neg\mathtt{Bad}])$$

として評価されているが, 式変形をフォローできなかったので, ここでは $4\Pr[\mathtt{Bad}]$ の項を含めることにした.

$$(\mathtt{AskG} \wedge \neg \mathtt{AskH}) \wedge \neg \mathtt{Bad} = (\mathtt{AskG} \wedge \neg \mathtt{AskH}) \wedge \neg(\mathtt{FBad} \vee \mathtt{GBad}) \tag{2.25}$$

なので，$s^\star$ が H に発行されることなく，$r^\star$ が G に発行されて，かつ $\neg(\mathtt{FBad} \vee \mathtt{GBad})$ ならば，$G(r^\star)$ が $s^\star \oplus (pHash \parallel 0 \cdots 001 \parallel m_0)$ あるいは $s^\star \oplus (pHash \parallel 0 \cdots 001 \parallel m_1)$ に一致する確率は，

$$q_G \cdot 2^{-k_0} \times 2 \cdot 2^{-k+k_0} = 2q_G \cdot 2^{-k} = q_G \cdot 2^{-k+1} \tag{2.26}$$

となる．さらに，もし攻撃者が P 成分を制御できて，かつ関数 $Hash$ の一方向性が破れるとしたなら，

$$2^{k_0} \times q_G \cdot 2^{-k_0} \times 2 \cdot 2^{-k+k_0} = 2q_G \cdot 2^{-k+k_0} = q_G \cdot 2^{-k+1+k_0} \tag{2.27}$$

となる．従って，

$$\mathtt{pr}[(\mathtt{AskG}) \wedge \mathtt{AskH} \,|\, \neg \mathtt{Bad}] \geq \frac{\widetilde{\epsilon}}{2} - q_G \cdot 2^{-k+1+k_0} / \mathtt{pr}[\neg \mathtt{Bad}] \tag{2.28}$$

をえる．これより，

$$\begin{aligned} \mathtt{pr}[\mathtt{AskG} \wedge \mathtt{AskH}] &\geq \mathtt{pr}[\mathtt{AskG} \wedge \mathtt{AskH} \wedge \neg \mathtt{Bad}] \\ &= \mathtt{pr}[\mathtt{AskG} \wedge \mathtt{AskH} \,|\, \neg \mathtt{Bad}] \times \mathtt{pr}[\neg \mathtt{Bad}] \\ &\geq \frac{\widetilde{\epsilon}}{2} \times \mathtt{pr}[\neg \mathtt{Bad}] - q_G \cdot 2^{-k+1+k_0} \end{aligned} \tag{2.29}$$

が示せる．

ところで，文献 [10] の議論によって，

$$\begin{aligned} \mathtt{pr}[\mathtt{Bad}] &= \mathtt{pr}[\mathtt{Bad} \,|\, \neg \mathtt{FAskH}] \times \mathtt{pr}[\neg \mathtt{FAskH}] + \mathtt{pr}[\mathtt{Bad} \,|\, \mathtt{FAskH}] \times \mathtt{pr}[\mathtt{FAskH}] \\ &\leq \mathtt{pr}[\mathtt{Bad} \,|\, \neg \mathtt{FAskH}] + \mathtt{pr}[\mathtt{FAskH}] \end{aligned} \tag{2.30}$$

において，$\mathtt{pr}[\mathtt{FAskH}] \leq q_H \cdot 2^{-k+k_0+8}$, $\mathtt{pr}[\mathtt{FBad} \wedge \mathtt{GBad} \,|\, \neg \mathtt{FAskH}] \leq q_G \cdot 2^{-k_0}$ が成立つので，

$$\begin{aligned} \mathtt{pr}[\mathtt{Bad}] &\leq \mathtt{pr}[\mathtt{FBad} \wedge \mathtt{GBad} \,|\, \neg \mathtt{FAskH}] + \mathtt{pr}[\mathtt{FAskH}] \\ &\leq q_G \cdot 2^{-k_0} + q_H \cdot 2^{-k+k_0+8} \end{aligned} \tag{2.31}$$

となる．また，$\mathtt{pr}[\mathtt{Bad}] = \Pr[\mathtt{Bad}]$ も成立つ．

以上より，

$$\frac{\epsilon}{2} \times (1 - q_G \cdot 2^{-k_0} - q_H \cdot 2^{-k+k_0+8}) - q_G \cdot 2^{-k+1+k_0} \tag{2.32}$$

が示せた． □

このとき，$(\mathcal{B})$ が $c^\star$ を復号できる確率 ϵ' は

$$\begin{aligned}\epsilon'(t'(k)) \quad \geq \quad & \left\{\frac{\widetilde{\epsilon}}{2} \times (1 - \frac{q_G}{2^{k_0}} - \frac{q_H}{2^{k-k_0-8}}) - \frac{q_G}{2^{k-1-k_0}}\right\} \\ & - \left\{q_D \times \left(\frac{1}{2^{k_1-1}} + \frac{2q_G+1}{2^{k_0}}\right) + \mathtt{Succ}^{\text{sp-ow}}(q_H, t')\right\}. \quad (2.33)\end{aligned}$$

となる．右辺の第 1 項は補題 2.4.1 で与えられ，右辺の第 2 項は式 (2.18) で与えられる．

$\mathtt{Succ}^{\text{ow}}(\tau)$ の定義に注意すると，

$$\epsilon(k) > 2 \times \frac{\mathtt{Succ}^{\text{ow}}(t') + q_H \times \mathtt{Succ}^{\text{p-ow}}(t') + p_D \times (\frac{1}{2^{k_1-1}} + \frac{2q_G+1}{2^{k_0}}) + \frac{q_G}{2^{k-1}}}{1 - \frac{q_G}{2^{k_0}} - \frac{q_H}{2^{k-k_0}}}$$

定理 2.4.2 (RSA-OAEP01 の安全性)
実行時間が t 以下で decryption cracle への質問回数が q_D 以下，ハッシュ関数 G への質問回数が q_G 以下，ハッシュ関数 H への質問回数が q_H 以下の OAEP01 変換 $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ の"semantic security" を破る CCA-adversary $\mathcal{A}$ が与えられたとき，$\mathcal{A}$ の advantage ϵ は

$$\widetilde{\epsilon(k)} > 2 \times \frac{\mathtt{Succ}^{\text{ow}}(t') + q_H \times \mathtt{Succ}^{\text{p-ow}}(t') + p_D \times (\frac{1}{2^{k_1-1}} + \frac{2q_G+1}{2^{k_0}}) + \frac{q_G}{2^{k-1-k_0}}}{1 - \frac{q_G}{2^{k_0}} - \frac{q_H}{2^{k-k_0-8}}}$$

ただし，

$$t'(k) = t(k) + q_G(k)q_H(k) \cdot (T_f + O(1))$$

ここで，T_f は関数 f の計算時間を表し，
$\widetilde{\epsilon} = \epsilon - 4\Pr[\neg\mathtt{Bad}] \leq \epsilon - \frac{q_G}{2^{k_0-2}} - \frac{q_H}{2^{k-k_0-10}}$ とする．

2.5 Manger の攻撃に対する対策の評価

文献 [12] で指摘された攻撃は，暗号文を復号した結果，形式検査でエラーが通知されたとき，エラーと種別が分かると，その情報を用いて暗号文が解読できるというものである．(文献 [6] の攻撃と類似している．選択暗号文攻撃の一種)

仕様を，誤りの種別が外に漏れないようにすることで，対策を施してある．

[OAEP01-Verify $(y, pHash, k, k_0)$**]**

Write $y = X \parallel t \parallel s \quad (|X| = 8, |t| = k_0$ and $|s| = k - 8 - k_0)$;
$r \longleftarrow t \oplus H(s)$;
Write $G(r) \oplus s = pHash' \parallel \alpha \parallel T \parallel M$
$(|pHash'| = k_0, \alpha = 0^{(8\text{の倍数})}, |T| = 8$ かつ $T \neq 0^8)$;
If $pHash' \neq pHash$, if $X \neq 0^8$, or if $T \neq 0^71$, then output "decoding error." Else output M.

"Decoding error" の通知を，プログラムの一箇所で判定するようにしたので，タイミング攻撃等に対する耐性が考慮されていると主張する根拠と考えられる．妥当な解決策だと判断する．

Bibliography

[1] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO'98*, pages 26–45. Springer, 1998. Lecture Notes in Computer Science No. 1462.

[2] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols,. In *Proc. of the First ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[3] M. Bellare and P. Rogaway. Optimal asymetric encryption — how to encrypt with RSA. In A.D. Santis, editor, *Advances in Cryptology — EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Berlin, Heidelberg, New York, 1995. Springer-Verlag.

[4] M. Bellare and P. Rogaway. The exact security of digital signatures –how to sign with RSA and Rabin. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, Berlin, Heidelberg, New York, 1996. Springer-Verlag.

[5] M. Bellare and A. Sahai. Non-malleable encryption: Equivalence berween two notions, and an indistinguishability -based characterization. In M. Wiener, editor, *Advances in Cryptology — CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 519–536, Berlin, Heidelberg, New York, 1999. Springer-Verlag.

[6] D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS#1. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO'98*, volume 1462 of *Lecture Notes*

in Computer Science, pages 1–12, Berlin, Heidelberg, New York, 1998. Springer-Verlag.

[7] D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, 46(2):203–213, 1999.

[8] J. S. Coron. On the exact security of full domain hash. In M. Bellare, editor, *Advances in Cryptology — CRYPTO'2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235, Berlin, Heidelberg, New York, 2000. Springer-Verlag.

[9] A. Fiat and A. Shamir. How to prove yourself. In A.M. Odlyzko, editor, *Advances in Cryptology — CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–208, Berlin, Heidelberg, New York, 1986. Springer-Verlag.

[10] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is chosen-ciphertext secure under the RSA assumption. In J. Kilian, editor, *Advances in Cryptology — CRYPTO'2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 260–274, Berlin, Heidelberg, New York, 2001. Springer-Verlag.

[11] Jakob Jonsson. Security proofs for the RSA-PSS signature schemes and its variants –draft 1.1. Available at http://eprint.iacr.org/2001/053/, 2001.

[12] J. Manger. A chosen ciphertext attack on rsa optimal asymmetric encryption padding (OAEP) as standardized in PKCS# v2.0. In J. Kilian, editor, *Advances in Cryptology — CRYPTO'2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 230–238, Berlin, Heidelberg, New York, 2001. Springer-Verlag.

[13] P. Q. Nguyen and J. Stern. The two faces of lattices in cryptology. In J. H. Silverman, editor, *Cryptography and Lattices: International Conference, CALC 2001*, volume 2146 of *Lecture Notes in Computer Science*, pages 146–180, Berlin, Heidelberg, New York, 2001. Springer-Verlag.

[14] V. Shoup. OAEP reconsidered. In J. Kilian, editor, *Advances in Cryptology — CRYPTO'2001*, volume 2139 of *Lecture Notes in Computer*

Science, pages 239–259, Berlin, Heidelberg, New York, 2001. Springer-Verlag.