

# TLS への Bleichenbacher's CAT 攻撃の考察および実装

嶂南 秀敏<sup>1</sup> 王 イントウ<sup>1</sup> 藤崎 英一郎<sup>1</sup>

**概要：**インターネット上での安全な通信実現のために、SSL/TLS が利用されている。SSL/TLS ではハンドシェイク部とレコード部に分かれており、まずハンドシェイク部で公開鍵暗号技術を使って、通信相手の認証や、次のレコード部で使用する秘密鍵の共有を行っている。その後レコード部では、共有した鍵を使って、共通鍵暗号技術を利用することで大量のデータの暗号化を行い、暗号化通信を行っている。ハンドシェイク部で利用する公開鍵暗号の一つに RSA 暗号がある。TLS での RSA 暗号利用は、はじめに送りたいデータに対して PKCS#1 に基づきパディングを行い、その後暗号文を作成することで、高い安全性を実現している。ここで PKCS とは、Public-Key Cryptography Standards の略称で、インターネット上でのさまざまな公開鍵暗号に関する技術仕様が定められている規格群である。#1 から #15 まで存在し、#1 には RSA 暗号について記載されている。TLS はこれまで何度も脆弱性とそれを利用する攻撃手法が発見され、その都度何度もバージョンアップがなされており、SSL2.0 から始まり名称変更を経て最新版は TLS1.3 となっている。代表的な SSL/TLS の脆弱性の一つに、PKCS#1 を狙った攻撃がある。そこで本研究では SSL/TLS の攻撃手法や、PKCS#1 の脆弱性を調査した後、Bleichenbacher 攻撃、Manger 攻撃のアルゴリズム解析や実装を行ったのち、2019 年に発表された格子の手法を用いた PKCS#1 に対する CAT 攻撃の実装とそれらの実装結果に関する考察を行う。

**キーワード：**SSL/TLS, RSA, PKCS#1, Bleichenbacher's Attack

## 1. はじめに

インターネット上での安全対策が要求される通信において通信内容の暗号化・通信内容の改ざん検出・通信相手の認証を行う TLS (Transport Layer Security) が広く用いられている [1]。TLS は、SSL2.0 (Secure Socket Layer) から始まり、何度も仕様上の脆弱性が発見され、それに伴ってバージョンアップがなされてきた。現在は SSL から TLS への名称変更を経て、TLS1.3 となっている。現在、TLS1.3 以前のすべてのバージョンのそれぞれに対して有効な攻撃手法が発見されている。それにより 2020 年に、Chrome, Firefox などの主要 web ベンダーにおいて TLS1.1 以前のすべてのバージョンのサポートが終了されることがわかっている。TLS1.3 の安全性について現在も議論が行われているが今のところ根本的な脆弱性は発見されていない。

TLS に対する攻撃は大きく 2 つに分類され、仕様そのものの問題と実装上の問題に分けられる。前者の問題では、Renegotiation を用いた中間者攻撃 [2] や暗号化モードのパディング処理での脆弱性をついた PaddingOracle 攻撃 [3] など

どその他にも様々な攻撃手法が発見されている。

また、過去にはバージョンアップにより脆弱性が改善されても過去に用いられた対策済みの攻撃手法のアイデアを再度違う方法で用いることで新しいバージョンでも有効な攻撃手法が発見されたケースが存在している。

そこで、本研究では TLS 各バージョンの仕様上の脆弱性と効果的な攻撃手法を調査しまとめるとともに、2019 年に発表された新たな PaddingOracle 攻撃である、Bleichenbacher's CAT Attack の考察と実装を行う。

## 2. SSL/TLS

### 2.1 SSL/TLS の概要

SSL (Security Socket Layer) / TLS (Transport Layer Security) とは、インターネット上の HTTP 通信においてサーバーとクライアント間での通信内容の暗号化に用いられる技術であり、インターネット上での安全な通信を実現するために開発された。我々がブラウザを通してインターネットを利用する際に一般的に使用されており、URL に “https” と記載されていれば SSL/TLS を利用していることを意味している。

SSL/TLS では手順として大きく 2 つに分けることができ、ハンドシェイクプロトコルとレコードプロトコルで構

<sup>1</sup> 北陸先端科学技術大学院大学  
Japan Advanced Institute of Science and Technology  
石川県能美市旭台 1-1, 923-1292

成される。

### 2.1.1 ハンドシェイクプロトコル

ハンドシェイクプロトコルでは、サーバーとクライアントでお互いの認証を行い、利用する TLS のバージョン、レコードプロトコルで使用するセッション鍵の共有を行う。実現のために公開鍵暗号技術を利用している。

### 2.1.2 レコードプロトコル

レコードプロトコルでは、ハンドシェイクプロトコルで生成したセッション鍵を使用して通信内容の暗号化し、相手に送信している。実現のために、共通鍵暗号技術を利用している。

## 2.2 Public-Key Cryptography Standards (PKCS)

Public-key Cryptography Standards (PKCS) とは公開鍵暗号の使用法やフォーマットやパディングの方法などが規定されている標準である。PKCS は#1 ~ #15 まで存在しており、例えば#1 は RSA 暗号について、#3 は Diffie-Hellman 鍵共有法について記載されている。

TLS では、RSA 暗号利用時この PKCS に沿ってパディングされ暗号化される。そのため、本研究では PKCS#1 という、TLS における RSA 暗号利用について扱う。

### 2.2.1 PKCS#1 Ver 1.5

1993 年に制定され、TLS 1.2 以前のバージョンで使用可能であり、RFC2313 [4] に記されている。

PKCS#1 Ver 1.5 では、以下の状態になるようにデータのパディング処理をした後、暗号化を行う。

**定義.** RSA 公開鍵が  $N, e$  のとき  $k = \log_{256} N$  とする。EB を  $k$  bytes の暗号化ブロックとするとき、以下の条件を満たす場合 EB は "PKCS Comforming" であるという。

$$EB = EB_1 || \dots || EB_k$$

- $EB_1 = 00$
- $EB_2 = 02$
- $EB_3 \sim EB_{10}$  は非零の乱数
- $EB_{11} \sim EB_k$  のどれか 1 つが 00

復号したものの、それを仮に  $m$  とする、 $m$  が PKCS Comforming であるとき、その暗号文である  $c$  もまた PKCS Comforming であると呼ぶこととされている。

EB を公開鍵で暗号化した後、送信する。受け取った側は、復号した後 PKCS Comforming であるかどうか確認し、Data を取り出す。

TLS での RSA 暗号鍵交換利用時、この EB 内に次のレコードプロトコルで使用するセッション鍵 (PreMasterSecret) が含まれている。そのため、攻撃者は EB を解読することでレコードプロトコルでの暗号化されたデータを全て知ることができる。

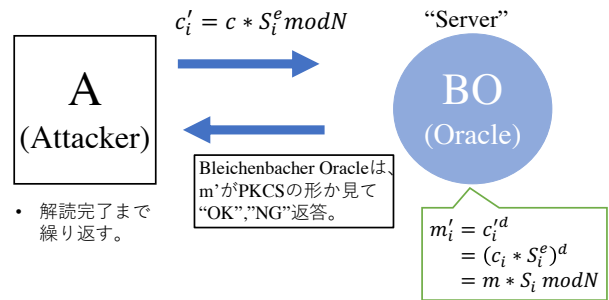


図1 Bleichenbacher Attack

## 3. RSA 暗号鍵交換での SSL/TLS への攻撃手法

### 3.1 Bleichenbacher Attack

#### 3.1.1 概要

Bleichenbacher Attack は 1998 年に Daniel Bleichenbacher によって発表された攻撃 [5] である。PKCS#1 Ver 1.5 の脆弱性を利用することで、ハンドシェイクプロトコルにおいて RSA 暗号利用のときに Server からの応答を利用し、「PreMasterSecret」を解読するものとなっている。

#### 3.1.2 Bleichenbacher Attack アルゴリズムの方針

Attacker は公開鍵  $e, N$  から  $m \equiv c^d \pmod{N}$  となるような暗号文  $c$  を復号したいとする。

Attacker は基本的に、整数  $s$  を選択した後以下のようにクエリ  $c'$  を計算する。

$$c' \equiv cs^e \pmod{N}$$

オラクルはクエリ  $c'$  受信後、

$$\begin{aligned} m' &\equiv (c')^d \bmod N \\ &\equiv (cs^e)^d \bmod N \\ &\equiv ms \bmod N \end{aligned}$$

を計算し、 $m'$  が PKCS Comforming であるかどうかを確認する。

もしオラクルの返答の意味が「 $c'$  は PKCS Comforming」ならば、Attacker はクエリ  $c'$  を復号したものである、 $ms \bmod N$  の最初の 2 バイトが 0x0002 となっていることがわかる。ここで  $k$  を公開鍵  $N$  のバイト長とした時便宜上、

$$B = 2^{8(k-2)}$$

ように  $B$  を設定する。「 $c'$  が PKCS Comforming である」という情報から Attacker は (1) 式を得ることができる。

$$2B \leq ms \bmod N < 3B \quad (1)$$

上記のようにクエリを作成し繰り返し送ることで、(1) 式のような、 $ms \bmod N$  の存在する範囲を狭めていくことができる。 $i$  回目のクエリ送信時に得られる平文候補集合を  $M_i$  とすると、最終的に得られる  $M_i$  の要素が一つに定まったとき、 $m$  を求めることができる。

### 3.1.3 Bleichenbacher Attack アルゴリズムの流れと処理

Bleichenbacher Attack は大きく 4 つのフェーズからできている。「Blinding」フェーズ、「Searching」フェーズ、「Narrowing」フェーズ、「Computing」フェーズである。

#### (1) 「Blinding」フェーズ

このフェーズでは Attack 開始時に、攻撃対象の暗号文  $c$  を PKCS Comforming の形に変換する。

暗号文  $c$  に対し  $c(s_0)^e \bmod N$  をオラクルに送ったときにそれが PKCS Comforming になるような、整数  $s_0$  をランダムに選択する。そうして、最初に成功した値  $s_0$  に対して以下の処理を行う。

$$\begin{aligned} c_0 &\leftarrow c(s_0)^e \bmod N \\ M_0 &\leftarrow \{[2B, 3B - 1]\} \\ i &\leftarrow 1. \end{aligned}$$

攻撃対象の暗号文  $c$  が PKCS Comforming であれば、 $s_0 = 1$  として処理する。

#### (2) 「Searching」フェーズ

このフェーズで行う処理は 3 パターンある。

(a)  $i = 1$  のときだけ行う。

$s_1 \geq \frac{N}{3B}$  を満たし、 $c_0(s_1)^e \bmod N$  が PKCS Comforming となる最小の  $s_1$  を見つける。

(b)  $i > 1$  かつ  $|M_{i-1}| \geq 2$  のとき行う。

$s_i > s_{i-1}$  を満たし、 $c_0(s_i)^e \bmod N$  が PKCS Comforming となる最小の  $s_i$  を見つける。

(c)  $|M_{i-1}| = 1$  のとき行う。

$c_0(s_i)^e \bmod N$  が PKCS Comforming を満たし、以下の式を満たす最小の  $r_i, s_i$  を見つける。

$M_{i-1} = \{[a, b]\}$  とすると、

$$\begin{aligned} r_i &\geq 2 \frac{(bs_{i-1} - 2B)}{N} \\ \frac{2B + r_i N}{b} &\leq s_i \leq \frac{3B + r_i N}{a} \end{aligned}$$

#### (3) 「Narrowing」フェーズ

「Searching」フェーズで、見つけた  $s_i$  を用いて、新しく平文候補集合  $M_i$  を求める。

$$M_i \leftarrow U_{(a,b,r)} \left\{ \left[ \max \left( a, \left\lceil \frac{2B + rN}{s_i} \right\rceil \right), \min \left( b, \left\lfloor \frac{3B - 1 + rN}{s_i} \right\rfloor \right) \right] \right\}$$

$[a, b] \ni M_{i-1}$  かつ  $\frac{as_i - 3B + 1}{N} \leq r \leq \frac{bs_i - 2B}{N}$

#### (4) 「Computing」フェーズ

このフェーズは、平文候補集合  $M_i$  のサイズが 1 つになったときに行う。例えば  $M_i = \{[a, a]\}$  となった場合、以下の処理を行うことで平文  $m$  を導出する。

$$m \leftarrow a \cdot (s_0)^{-1} \bmod N$$

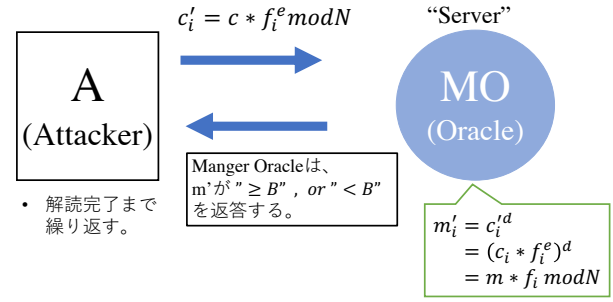


図2 Manger Attack

それ以外のときは  $i \leftarrow i + 1$  として「Searching」フェーズに移る。

処理手順は、最初に一度「Blinding」を行い、その後「Searching」→「Narrowing」→「Computing」を繰り返して実行する。

## 3.2 Manger Attack

### 3.2.1 概要

Manger Attack は、PKCS#1 Ver 2.0 での脆弱性をついた攻撃であり、James Manger によって 2001 年に発表された [6]。PKCS#1 Ver 2.0 での脆弱性では、先頭の 1byte がエラー処理によって「0x00」であることがわかってしまうことがあった。そのため、Bleichenbacher Attack と同様に、ハンドシェイクプロトコルでの Server の脆弱性を利用して、RSA 暗号で暗号化された「PreMasterSecret」を解読する攻撃となっている。

### 3.2.2 Manger Attack アルゴリズムの方針

Attacker は公開鍵  $e, N$  から  $m \equiv c^d \pmod{N}$  となるような暗号文  $c$  を復号したいとする。図 2 のようなオラクルを形成し、クエリ  $c'$  を送ることで攻撃を行う。

Attacker は Bleichenbacher Attack と同じく基本的に、以下のようにクエリ  $c'$  を計算することで、攻撃を進めていく。

$$c' \equiv c f^e \bmod N$$

Bleichenbacher Attack と少し異なっていることとしては、オラクルの返答によって知ることができる情報は、 $c'$  をクエリ送信した時、Bleichenbacher Attack と同様に  $k$  を公開鍵  $N$  のバイト長とすると、

$$0 \leq m f \bmod N < B$$

$$\text{ここで } B = 2^{8(k-1)}$$

である。これを繰り返すことで、 $m$  を 1 つに絞っていく。

### 3.2.3 Manger Attack アルゴリズムの流れと処理

MangerAttack のアルゴリズムは 3 ステップで構成される。

#### • Step 1:

Step1 は、オラクルが「 $\geq B$ 」と返答するまで  $f_1 = 2, 4, 8, \dots, 2^i$  として繰り返し、クエリを  $c' =$

$f_1^e c \pmod N$  と計算し攻撃を進める。Step1 が終了後、得られる  $m$  についての不等式は (2) 式のようにになっている。

$$\frac{B}{f_1} \leq m < \frac{2B}{f_1} \quad (2)$$

Step1 の終了後、Step2 を行う。

- Step 2:

Step2 では、オラクルが「 $< B$ 」と返答するまで、 $f_2 = (\lfloor \frac{n+B}{B} \rfloor + k) \cdot \frac{f_1}{2}$  where  $k = 0, 1, 2, \dots$  として繰り返し、 $c' = f_2^e c \pmod N$  と計算し攻撃を進める。Step2 が終了後、得られる不等式は以下の (3) 式ようになっている。

$$\frac{N}{f_2} \leq m < \frac{N+B}{f_2} \quad (3)$$

Step2 の終了後、Step3 を行う。

- Step 3:

Step3 は、最終的に  $m$  が一つに確定するまで行う。Step3 が最も多く繰り返されるステップであり、繰り返す毎に、 $m$  の存在する可能性のある範囲が二分の一のようになっていく。

公開鍵  $N$  の bit 長程度程度の試行が必要である。Step1 は、最大で 10 回程度である。Step2 の試行回数は最大で 256 回である。Step3 の試行回数が最も多い。

### 3.3 CAT Attack

#### 3.3.1 概要

CAT Attack [7] は 2019 年に発表された Bleichenbacher Attack や Manger Attack の発展形の攻撃である。以前の 2 つの攻撃は、一つのサーバーに対し連続的にクエリを解読完了まで送り続ける必要があり、実社会ではブラウザに設定されたタイムアウト (平均 30 秒) に間に合わないことがわかっていていた。

CAT Attack では、同じ証明書を用了ミラーサーバーにパラレルに攻撃した後に、受け取ったデータを Hidden Number Problem [8] として扱い、格子の技術を用いることで解読が可能となっている。

#### 3.3.2 Hidden Number Problem (HNP)

ここでは、CAT Attack でのパラレルアタック実現のために用いられている Hidden Number Problem について簡単に説明する。

Hidden Number Problem は、1996 年に Dan Boneh らによって発表された論文 [8] 内で定義された問題である。Hidden Number Problem は以下のように定義されている。

**定義 3.1.** 素数である  $p$  と、 $k$  は固定され、 $\mathcal{O}_\alpha(t)$  を入力  $t$  に対して、 $\alpha \cdot t \pmod p$  の  $k$  番目までの最上位 bit の値を計算するオラクルとする。

$$\mathcal{O}_\alpha(t) = MSB_k(\alpha \cdot t \pmod p)$$

オラクル  $\mathcal{O}_\alpha(t)$  が与えられたときに、多項式時間で Hidden Number である  $\alpha \pmod p$  を計算する問題である。

ここで、 $MSB_k(x)$  は  $|x - z \cdot 2^{(\log_2 p - k - 1)}| < p/2^{k+1}$  を満たす整数  $z$  を出力する関数となっている。例えば、 $k = 3, x = 45 = 101101_{(2)}$  とすると、 $MSB_k(x) = 5$  となっている。

定義 3.1 から以下の定理 3.2 が述べられており、CAT Attack における格子利用のアイデアは定理の証明の部分から用いられている。

**定理 3.2.**  $[1, p-1]$  の範囲に存在する整数を  $\alpha$  とする。オラクル  $\mathcal{O}_\alpha(t) = MSB_k(\alpha \cdot t \pmod p)$  が与えられており、 $k = \lceil \sqrt{\log_2 p} \rceil + \lceil \log_2 \log_2 p \rceil$  である。

$d = 2\sqrt{\log_2 p}$  として、 $t_1, \dots, t_d$  を  $\mathbb{Z}_p^*$  から、ランダムに一樣独立で選んだとき以下の (4) 式を満たす決定的多項式時間アルゴリズム  $A$  が存在する。

$$\Pr[A(t_1, \dots, t_d, \mathcal{O}(t_1), \dots, \mathcal{O}(t_d)) = \alpha] \geq \frac{1}{2} \quad (4)$$

$i = 1, \dots, d$  として、 $t_1, \dots, t_d$  をオラクル  $\mathcal{O}_\alpha(t)$  に入力したときに対応する出力を  $a_1, \dots, a_d$  とする。その時、オラクルの出力によって (5) 式が得られる。

$$|(\alpha t_i \pmod p) - a_i \cdot 2^{(\log_2 p - k - 1)}| < \frac{p}{2^k} \quad (5)$$

Hidden Number である  $\alpha$  を見つけるために以下の (7) 式の  $d+1$  次元の格子を作成し、目標ベクトル  $\omega$  を (6) 式と設定して、

$$\omega = (a_1, \dots, a_d, 0) \quad (6)$$

$$HNBase = \begin{pmatrix} p & 0 & 0 & \cdots & 0 & 0 \\ 0 & p & 0 & \cdots & 0 & 0 \\ & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & p & 0 \\ t_1 & t_2 & t_3 & \cdots & t_d & \frac{1}{p} \end{pmatrix} \quad (7)$$

LLL 基底簡約アルゴリズムをした後、CVP を解くアルゴリズムを使用することで (8) 式の形の最近ベクトル  $\nu$  を求めることができる。

$$\nu_\alpha = \left( r_1, \dots, r_d, \frac{\alpha}{p} \right) \quad (8)$$

最終的に、 $\alpha = r_1 \cdot t_1^{-1} \pmod N$  を計算することで、Hidden Number である  $\alpha$  を計算している。

#### 3.3.3 格子暗号技術

Hidden Number Problem と CAT Attack に関連する代表的な格子問題である、最短ベクトル問題 (SVP) と最近ベクトル (CVP) 問題を簡単に説明する。

- 最短ベクトル問題 (SVP)

基底が与えられたときに、格子上に存在する非零の最短ベクトルを見つける問題である。代表的なアルゴリ

ズムに、LLL 基底簡約アルゴリズム [9] がある。

- 最近ベクトル問題 (CVP)

基底と目標ベクトルが与えられたときに、目標ベクトルに最も近い、格子上に存在するベクトルを見つける問題である。代表的なアルゴリズムに Babai の最近平面アルゴリズム [10] が存在する。

CAT Attack では、復号のために SVP に帰着され、Hidden Number Problem では CVP に帰着している。

これら 2 つの問題を解決するためのアルゴリズムを用いている。

### 3.3.4 CAT Attack アルゴリズムの方針

Attacker は公開鍵  $e, N$  から  $m \equiv c^d \pmod{N}$  となるような暗号文  $c$  を復号したいとする。CAT Attack では、同じ公開鍵を持つ複数のサーバー（オラクル）に平行にクエリを送ることで攻撃を進める。CAT Attack は平行に Manger Attack を行うものと、平行に Bleichenbacher Attack を行う 2 種類あるが、本節は平行 Manger Attack を扱う。そのため以下の図 3 のように、複数の Manger Oracle を用いた平行アタックの説明を行う。

CAT Attack では平行にアタックするために、開始直後「Blinding」と称して、複数のオラクル用に攻撃対象暗号文の初期値をそれぞれ変える。利用するオラクル数  $k$  を  $1 \leq k \leq j$  とした時、PKCS Comforming となる平文  $M_1, \dots, M_k, \dots, M_j$  に対応する暗号文  $C_1, \dots, C_k, \dots, C_j$  を作成する。

その後、一定時間終了（クエリ数や時間で制限）までそれぞれ  $C_k$  ( $1 \leq k \leq j$ ) を攻撃対象暗号文としてそれぞれのサーバー（オラクル）に対して、平行に Manger Attack を行わせる。例えば  $k$  番目の平行アタックでは、 $M_k$  の候補の範囲を狭める作業を行う。

一定時間終了時、最終的にそれぞれのアタックで  $M_k$  ( $1 \leq k \leq j$ ) に関する平行数だけ不等式が得られる。

それらの不等式から、格子の基底を作成し、SVP を解くアルゴリズムを用いることで、CAT Attack での攻撃対象暗号文  $c$  に対応する平文  $m$  を計算することができる。

CAT Attack の攻撃のイメージは図 3 のようになっている。

### 3.3.5 CAT Attack アルゴリズムの流れと処理

まずはじめ、「Blinding」フェーズで、平行アタックを実現するために、攻撃対象暗号文  $c$  に対して異なる  $s^{(k)}$  ( $1 \leq k \leq j$ ) で PKCS Comforming の形に変換することで、以下のように平行数分暗号文  $C^{(k)}$  を生成する。

$$\begin{aligned} C^{(k)} &\equiv c \cdot \left(s^{(k)}\right)^e \pmod{N} \\ &\equiv \left(m \cdot s^{(k)}\right)^e \pmod{N} \end{aligned}$$

$C^{(k)}$  に対応する平文を  $M^{(k)}$  とする時、PKCS Comforming

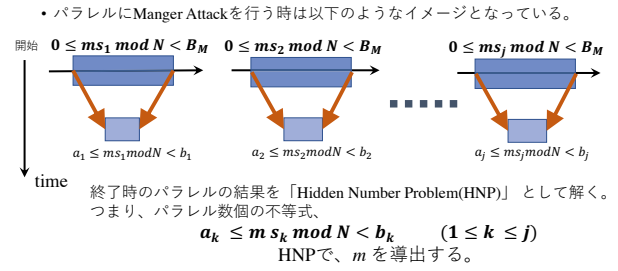


図 3 CAT Attack

であることから、以下の式を満たしている。

$$0 \leq M^{(k)} < B \quad (9)$$

そして、それぞれのオラクルに対して平行に Manger Attack を一定時間終了まで行う。 $k$  個目の平行アタックでは、 $M^{(k)} \in [0, B]$  の情報から、 $M^{(k)}$  の候補の範囲を狭めるために Manger Attack のアルゴリズムを開始する。

その後例えば、 $k$  個目の平行な Manger Attack の Step 1 の終了時に得られる不等式は以下の 10 のようになっている。

$$\frac{B}{f_1^{(k)}} \leq M^{(k)} < \frac{2B}{f_1^{(k)}} \quad (10)$$

一定時間終了時に平行数だけ  $M$  に関する不等式を得ることができる。終了時、 $k$  番目の平行アタックで  $i$  個のクエリを送信していた場合に得られた範囲が  $M^{(k)} \in [a_i^{(k)}, b_i^{(k)}]$  であったとき、攻撃対象暗号文  $c$  に対応する  $m$  についての式に変形できる。

$$\begin{aligned} a_i^{(k)} &\leq M^{(k)} < b_i^{(k)} \\ \Leftrightarrow 0 &\leq M^{(k)} - a_i^{(k)} < b_i^{(k)} - a_i^{(k)} \\ \Leftrightarrow 0 &\leq m \cdot s^{(k)} \pmod{N} - a_i^{(k)} < b_i^{(k)} - a_i^{(k)} \end{aligned} \quad (11)$$

(11) 式は、Hidden Number Problem (HNP) の (5) 式の形に似ている。そのため、HNP での Hidden Number を見つける手法を使って  $m$  を計算する。

ここで HNP と CAT Attack との格子を使って解決する方法の間には少し相違点がある。 $m$  を計算する際、HNP では CVP を解くアルゴリズム、CAT Attack では SVP を解くアルゴリズムを用いている。そのため、格子の基底の形も異なっている。4 節では、HNP 版と CAT Attack 版の両方格子での解法で実装している。

CAT Attack で用いる格子の基底は以下の (12) 式のようにになっている。

$$CATBase = \begin{pmatrix} s^{(1)} & s^{(2)} & s^{(3)} & \dots & s^{(j)} & 0 \\ N & 0 & 0 & \dots & 0 & 0 \\ 0 & N & 0 & \dots & 0 & 0 \\ 0 & 0 & N & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & N & 0 \\ a_i^{(1)} & a_i^{(2)} & a_i^{(3)} & \dots & a_i^{(j)} & N \cdot \frac{(j-1)}{j} \end{pmatrix} \quad (12)$$

この格子にLLL 基底簡約を行うことで、最短ベクトル  $R^i$  を得ることができる。

$$R^i = \left( r_i^{(1)}, r_i^{(2)}, \dots, r_i^{(j)}, -N \cdot \frac{(j-1)}{j} \right) \quad (13)$$

$R^i$  導出後、最終的に  $m = (r_i^{(1)} + a_i^{(1)}) \cdot (s^{(1)})^{-1} \bmod N$  を計算することで平文  $m$  を得られる。

## 4. 実装

### 4.1 実装環境

#### 4.1.1 使用言語

使用言語は、「プログラミング言語 C++」を使用した。そして、1024bit や 2048bit の演算の高速な処理実現のために、ライブラリとして NTL [11] と一部 GMP [12] を使用した。また、RSA 暗号の 1024bit と 2048bit の公開鍵や秘密鍵のデータは、OpenSSL [13] のコマンドから生成したものを使用している。

#### 4.1.2 使用機器

実装する攻撃アルゴリズムは、一般的に普及している程度のスペックで暗号文を解析することを想定しており、スパコン等の使用は想定していない。そのため、Jaist 所有のスパコンは使用せず、自分が普段使用するノートパソコンである MacBook Air 上で実装し、動作確認を行った。CPU は、第 5 世代 Core i7 2.2GHz/2 コアとなっている。

### 4.2 Manger Attack の実装

オブジェクト指向の考えを利用して、「MangerOracle」としての動作を行うオブジェクトと、そのオブジェクトを利用して MangerAttack アルゴリズムを進める「MangerAttacker」オブジェクトを生成し、その 2 者間でやり取りをさせることで攻撃対象暗号文を解読し、解読までにかかるクエリ数や時間を測定している。

### 4.3 CAT Attack (Parallel Manger Attack) の実装

4.2 節で実装したものを再利用し、CAT Attack でのパラレル Manger Attack のアルゴリズムを実装した。「MangerAttacker」と「Manger Oracle」のオブジェクトをパラレル数個 ( $j$  個) と、それらの Manger Attacker の出力から格子を作成し格子アルゴリズムを使用することで「m」を解析

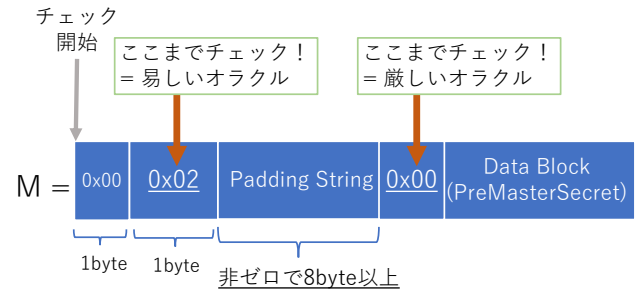


図4 実装した Bleichenbacher Oracle の種類

する「CAT Attacker」というオブジェクトを一つ生成し、それらのオブジェクト間でやり取りをすることで攻撃対象暗号文を解読できる。

パラレル数は最大 10 まで指定でき、それぞれのパラレルアタックでのクエリ数の上限も指定できる。それぞれの MangerAttacker はクエリ数の上限に達するとアルゴリズムを中断し、入力 of 初期値に関する不等式を出力する。CAT Attacker はパラレルの Manger Attacker からの出力の不等式から、CVP か SVP のための格子を作成し、格子アルゴリズムを使用して「m」を解析する。

### 4.4 Bleichenbacher Attack の実装

Manger Attack の実装と同じように、オブジェクト指向の考えを利用して、「BleichenbacherOracle」としての動作を行うオブジェクトと、そのオブジェクトを利用して Bleichenbacher Attack アルゴリズムを進める「BleichenbacherAttacker」オブジェクトを生成し、その 2 者間でやり取りをさせることで攻撃対象暗号文を解読し、解読完了までにかかるクエリ数や時間を測定している。本研究では、BleichenbacherOracle の実装において、以下の図 4 のような、2 種類のオラクルを実装した。受け取った暗号文に対応する平文をみて、16 進数表示で先頭の 2 バイトの「0002」かどうかのみを確認して返答するオラクル、仮に「易しいオラクル」命名する、と同様の平文に対して、先頭の 2 バイトが「0002」か確認した後、非ゼロが 8 バイト以上連続して存在し、その後どこかで「00」があるかどうかを確認して返答するオラクル、「厳しいオラクル」と命名する、ものの 2 つである。

## 5. 実装結果と考察

### 5.1 実装結果

#### Manger Attack と CAT Attack に関して

Manger Attack と CAT Attack (パラレル CAT Attack) の実装を行い、CAT Attack ではクエリ上限数別と鍵長別に 100 回の試行を行った。その結果を表 1 と表 2 にまとめる。

まず Manger Attack において一つのオラクルを利用し解

表 1 100 回のシミュレーション結果 ( $N = 1024 \text{ bit}$ )

| パラレル数<br>( $N=1024\text{bit}$ )         | 各オラクルの<br>上限クエリ数 | HNPの格子の<br>解読成功回数 | CATの格子<br>解読成功回数 | 平均時間 [s]<br>(1試行あたり) |
|-----------------------------------------|------------------|-------------------|------------------|----------------------|
| 10 個<br>CAT Attack<br>(Parallel Manger) | 225              | 6 / 100           | 6 / 100          | 2.69377              |
|                                         | 250              | 27 / 100          | 27 / 100         | 2.84215              |
|                                         | 275              | 42 / 100          | 42 / 100         | 3.21317              |
|                                         | 300              | 93 / 100          | 93 / 100         | 2.73724              |
| 5 個<br>CAT Attack<br>(Parallel Manger)  | 325              | 11 / 100          | 11 / 100         | 1.44026              |
|                                         | 350              | 39 / 100          | 39 / 100         | 1.62993              |
|                                         | 375              | 58 / 100          | 58 / 100         | 1.83912              |
|                                         | 400              | 89 / 100          | 89 / 100         | 1.73576              |
| 1 個<br>MangerAttack                     | 425              | 97 / 100          | 97 / 100         | 1.84671              |
|                                         | 1178<br>で解読完了    |                   |                  |                      |

表 2 100 回のシミュレーション結果 ( $N = 2048 \text{ bit}$ )

| パラレル数<br>( $N = 2048\text{bit}$ )       | 各オラクルの<br>上限クエリ数 | HNP格子の<br>解読成功回数 | CAT格子<br>解読成功回数 | 平均時間 [s]<br>(1試行あたり) |
|-----------------------------------------|------------------|------------------|-----------------|----------------------|
| 10 個<br>CAT Attack<br>(Parallel Manger) | 300              | 4 / 100          | 4 / 100         | 19.1799              |
|                                         | 325              | 21 / 100         | 21 / 100        | 19.4290              |
|                                         | 350              | 63 / 100         | 63 / 100        | 20.1005              |
|                                         | 375              | 99 / 100         | 99 / 100        | 21.5148              |
| 5 個<br>CAT Attack<br>(Parallel Manger)  | 525              | 12 / 100         | 12 / 100        | 14.3234              |
|                                         | 550              | 48 / 100         | 48 / 100        | 15.1788              |
|                                         | 575              | 89 / 100         | 89 / 100        | 14.6391              |
|                                         | 600              | 100 / 100        | 100 / 100       | 16.5872              |
| 1 個<br>MangerAttack                     | 2064<br>で解読完了    |                  |                 |                      |

読する方法は、ある 1024 ビットの公開鍵  $N$  のときの、解読までの総クエリ数は 1178 回であり、完了まで約 1 秒程度であった。そして、同じ 1024 ビットの公開鍵  $N$  の利用した CAT Attack の実装において、10 個の平行なオラクルを利用した時は、100 回のシミュレーションにおいて各オラクルの上限数が 300 の時 93% の正答率となった。5 個のオラクルを利用した時は、上限数が 425 の時に正答率が 97% となった。

1024bit の鍵長においては、10 個の平行での一つのオラクルへの上限クエリ数は通常の Manger Attack (平行なし) に必要なクエリ数の約 1/3 で 100% の解読成功可能となっている。そして、5 個の平行においては約 1/2 となっている。

#### Bleichenbacher Attack に関して

Bleichenbacher Attack を実装し、2 種類の Oracle 使用時の動作結果を表 3 にまとめる。

まず、1024bit 鍵長では「易しいオラクル」使用時に、解読完了まで約 7 万回のクエリ生成が必要になり、解読完了まで約 30 秒であった。一方で「厳しいオラクル」使用時は、クエリ数が 18 万となり約 1 分半で解読完了となった。次に、2048bit 鍵長では「易しいオラクル」使用時は、クエリ数が 14 万で約 8 分で解読でき、「厳しいオラクル」では、クエリ数が 32 万となり、解読完了までに約 17 分かかった。

表 3 Bleichenbacher Attack の 2 種類のオラクル使用時の結果

| 公開鍵 $N$ の<br>鍵長 | オラクルの<br>種類 | 解読完了までの<br>クエリ数 | 解読完了までの<br>時間 [s] |
|-----------------|-------------|-----------------|-------------------|
| 1024 bit        | 易しい         | 75,097          | 34.772            |
|                 | 厳しい         | 188,750         | 93.480            |
| 2048 bit        | 易しい         | 148,250         | 495.421           |
|                 | 厳しい         | 324,172         | 1033.412          |

## 5.2 考察

### Manger Attack と CAT Attack に関して

まず、2 つの表 2 と表 1 に共通して言えることとしては、Hidden Number Problem の格子を利用する方法と、CAT Attack での格子を利用する方法で解読成功回数がすべて一致したのは意外であった。前者の格子は CVP のための格子であり後者の格子は SVP のための格子であるため、CAT Attack 独自の改良がなされていると思ったからである。このことについて詳しく調べた結果、CAT Attack での用いられている格子は、CVP を解くアルゴリズムの一つである埋め込み法 [14] を利用するためであることがわかった。埋め込み法では、CVP の基底と目標ベクトルから SVP の形の基底を作る。その後、その基底に対して SVP を解くことで得られる最短ベクトルは、目標ベクトルに対する最近ベクトルとなっているという。そのため、CAT の格子では SVP を解くことで、CVP の解を得ているのである。

CAT Attack での格子を利用する場合では、最短ベクトルを求める際に、LLL アルゴリズムを行った後に、GaussSieve 等の SVP を解くためのアルゴリズムは使用しておらず、LLL の結果を使って平文  $m$  を計算している。それは、本研究での基底の次元 (パラレル数) は最大でも 10 次元であり、LLL アルゴリズムは 10 次元のような低い次元の基底に対して行ったときに、最短ベクトルも同時に出力してくれるからである。そのため、もし 100 次元などの高い次元つまり、仮に 100 パラレルで行った場合は LLL アルゴリズムだけでは求める数値は得られない可能性が高い。したがって、そのような場合は HNP の格子を使って、CVP を解くアルゴリズムを使用するのが良いと考えられる。

### Bleichenbacher Attack に関して

Bleichenbacher Attack の実装結果において完了までの時間を見たときに、自分の想像以上に解読完了までの時間がかかっていた。通常の Manger Attack では、1024bit の同じ公開鍵において約 2 秒程度で解読完了となっていたが、Bleichenbacher Attack では「易しいオラクル」において約 30 秒かかっていた。実装した途中出力を見たところ、Bleichenbacher Attack のアルゴリズムの中で最も時間がかかっていたのは、Step2.a と Step2.b であった。Step2.a ははじめの一回、Step2.b も多くて数回程度だが、それらだけで

全体の7割ほどの時間を費やしていることがわかった。

Manger Attack に比べて大きく時間がかかる考えうる理由の1つとしては、Bleichenbacher Attack のアルゴリズムでは PKCS Comforming とならなかったときにカウントアップ、つまり値を1ずつ増やして、次の PKCS Comforming となる値を見つけており、Manger Attack では、PKCS Comforming にならなかったときには値と定数の積を次の値としているため、解読完了が早くなっていると思われる。

また、Bleichenbacher Attack と Manger Attack でのオラクルが OK となるためのチェックに 1byte 差がある。つまり、Manger Attack では先頭の 1byte のチェックのみで、Bleichenbacher Attack の易しいオラクルのチェックは先頭の 2byte のチェックである。そのため、最大で8ビット、2の8乗倍のクエリ数の差が発生することが予想できる。

これらの理由から、Bleichenbacher Attack と Manger Attack は同じ公開鍵出会っても、終了時間に大きく差が発生するようだ。

最後に、利用したクエリ別で見るとやはり「厳しいオラクル」を用いるとクエリ数も時間もかかることがわかった。表3を見ても「易しいオラクル」利用時よりも、約2倍以上のクエリ数と、時間が必要になっている。

ここで、Bleichenbacher Attack で興味深いことは、エラーに強いということだ。実装では「厳しいオラクル」と「易しいオラクル」に対して攻撃を行ったが、そのどちらもクエリ数に差はあれど正しく復号することができた。一方で、Manger Attack は、そのような2種のオラクルで攻撃を正しく遂行することはできず、完璧なオラクルが要求されている。そのため、Bleichenbacher Attack はより実践的なアルゴリズムとなっていることがわかる。

## 6. おわりに

本研究では、SSL/TLS のハンドシェイクプロトコルでの RSA 暗号利用での脆弱性とその攻撃手法について論じ、その後それらのアルゴリズムの実装を行った。実際の TLS サーバーにアタックすることは叶わなかったため、オブジェクト指向の考えを用いてパディングオラクルと同様の処理をするオブジェクトを生成し、それを利用することで暗号文を解読していくようなアルゴリズムの実装を行った。実際のサーバに対して本研究での攻撃を行うとしたら、成功率や時間などは異なってくると思われる。

また、論文内では説明できなかったが、パラレル Bleichenbacher Attack も実装している。結果としては、例えば 1024bit の公開鍵に 15 パラレルでは各パラレル平均 22 万クエリで 88 秒で解読成功した。

今後の課題としては、今回の攻撃を実際のサーバーで行っても正しく動作するか試すことや、HNP の論文をより深く解析し、より多くのパラレル数で行う場合での検証を行いたいと思う。

謝辞 本研究は JSPS 科研費 JP20K23322 と JP19K11960 の助成を受けたものです。

## 参考文献

- [1] Tibor Jager, Saqib A. Kakvi, and Alexander May. On the security of the PKCS#1 v1.5 signature scheme. pp. 1195–1208, 2018.
- [2] Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Alfredo Pironti, and Pierre-Yves Strub. Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS. pp. 98–113, 2014.
- [3] Serge Vaudenay. Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS... pp. 534–546, 2002.
- [4] B. Kaliski. PKCS # 1: RSA Encryption Version 1.5, RFC2313. March 1998.
- [5] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. pp. 1–12, 1998.
- [6] James Manger. A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS #1 v2.0. pp. 230–238, 2001.
- [7] Eyal Ronen, Robert Gillham, Daniel Genkin, Adi Shamir, David Wong, and Yuval Yarom. The 9 lives of bleichenbacher's CAT: New cache ATtacks on TLS implementations. pp. 435–452, 2019.
- [8] Dan Boneh and Ramarathnam Venkatesan. Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. pp. 129–142, 1996.
- [9] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, Vol. 261, No. ARTICLE, pp. 515–534, 1982.
- [10] László Babai. On lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, Vol. 6, No. 1, pp. 1–13, 1986.
- [11] Victor Shoup. NTL : A Library for doing Number Theory. Available at <http://www.shoup.net>, 2016 / 1 / 30.
- [12] The GNU Multiple Precision arithmetic library. <https://gmplib.org/>.
- [13] The OpenSSL Project. OpenSSL: The open source toolkit for SSL/TLS. [www.openssl.org](http://www.openssl.org), April 2003.
- [14] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, Vol. 12, No. 3, pp. 415–440, 1987.