岐阜大学自然科学技術研究科

知能理工学専攻

令和 5 年度学位論文

一意復号可能な Rabin-Williams 暗号プリミティブの実装および署名スキームの検討

三嶋研究室

学籍番号：1224525099

森 晴樹

指導教員： 三嶋 美和子 教授

目次

はじめに

1979 年, Michael Oser Rabin は Rabin 暗号を提案した. Rabin 暗号は鍵となる $N = pq$ の素因数分解の困難性に基づく公開鍵暗号スキームであり, 選択平文攻撃に対する耐性が証明されている. また Rabin 暗号は $e = 2$ の RSA 暗号であり, $e \geq 3$ の RSA 暗号の変形といえる. 暗号化指数 $e = 2$ を利用している点において, Rabin 暗号は RSA 暗号よりも計算量的に優れている. しかし, Rabin 暗号は RSA 暗号とは秘密鍵とするものが異なる. そのため復号アルゴリズムも異なり, 2 回の冪剰余計算と中国剰余定理 (CRT) の計算を行わねばならないうえ, 出力される 4 つの平文候補の中からもとの平文を一意に特定できない. このため Rabin 暗号の実用化には, 一意復号という課題が残されていた.

Rabin 暗号が発表された翌年, Williams は Rabin 暗号のバリエーションを提案した. この暗号スキームでも依然として復号プロセス中に 4 つの潜在的なメッセージ候補が生成されるが, 彼は特定の素数の選択 ($p \equiv 3 \pmod 8, q \equiv 7 \pmod 8$) とメッセージ空間の制約により, これらの候補から正しいメッセージを一意に特定することが可能になることを示した. また, Williams はこの暗号スキームの署名スキームへの応用も提案している. 詳細は 2.2 節で解説する.

1980 年以降, Williams によって改良された Rabin 暗号を応用した署名スキームは, RSA 署名の効率的なバリエーションの 1 つとして,「Rabin-Williams 公開鍵署名スキーム」として知られるようになった. 実際, Bernstein (2008) は "Variants of the Rabin-Williams public-key signature system have, since 1980, held the speed records for signature verification" と述べている.

その後も多くの研究者が, Rabin 暗号を実用的かつ実装可能にすることを目的として, 研究を行ってきた. しかしそのうちの多くは, 一意復号を実現するために, 次に挙げるような追加処理を必要とし, そのために Rabin 暗号の RSA 暗号に対する計算量的な優位性が失われている.

- Jacobi 記号の計算
- 追加情報の付加
- メッセージパディングの利用

Batten and Williams (2019) は, 追加処理なしに一意復号可能な Rabin 暗号のバリエーションを提案した. しかも, 彼らの提案スキームの復号処理は, これまでで最も効率的であった. 彼らのスキームは公開鍵として $N = p^2q$ を使用し, その安全性は N の素因数分解の困難性に依存している. 彼らはこのスキームを Rabin-Williams 署名を改良した ‘署名スキーム’ と位置づけている. しかし, 筆者が理解した限りではその内容は公開鍵暗号スキームの提案にとどまり, 署名スキームとしての提案にはなっていない. 本研究は, この先行研究に基づき, 次の 2 つについて取り組んだ.

1. Batten and Williams の公開鍵暗号スキームを Python で実装する.

2. Batten and Williams の公開鍵暗号スキームをデジタル署名に応用する方法について検討する.

本論文は 4 章で構成される. まず, 第 1 章では, 本研究で提案するデジタル署名スキームで用いる数学理論および暗号理論について解説する. 第 2 章では, Rabin (1979) が提案した Rabin 暗号および Williams (1980) が提案した Rabin-Williams 暗号に関する主要な関連研究について解説する. 第 3 章では, Batten and Williams (2019) が提案した Rabin-Williams 暗号のバリエーションを実装した Python ライブラリに関する詳細な仕様を与える. 最後に第 4 章で, Batten and Williams[3] の暗号スキームのデジタル署名スキームへの応用について議論する.

第1章　準備

この章では, 本研究で提案するデジタル署名スキームで用いる数学理論および暗号理論について概説する.

1.1　DSA 署名

デジタル署名とは, メッセージ作成者の否認防止を含めた, 作成者の認証を可能にする暗号技術である. デジタル署名の目的は, いわゆる手書きの署名や捺印と同じ機能を電子的に達成することといえる.

DSA (Digital Signature Algorithm) は, 1993 年に FIPS 186 として標準化された, **DSS** (Digital Signature Standard) の主要なアルゴリズムの 1 つであった. なお, 2023 年の FIPS 186-5 [6] では, DSA は新たにデジタル署名を行うことには推奨されないが, 標準策定以前に行われた署名の検証には引き続き利用可能とされている. 一般にデジタル署名スキームは, 鍵生成アルゴリズム (署名者が実行), 署名生成アルゴリズム (署名者が実行), 署名検証アルゴリズム (署名受信者が実行) の 3 つのアルゴリズムから構成される. DSA におけるこれら 3 つのアルゴリズムの手順を以下に述べる.

鍵生成

1. セキュリティパラメータとして整数 $L, N\,(L > N)$ を定める. FIPS 186-4 では以下の 4 つの値の組が規定されている.

$$(L, N) = (1024, 160), (2048, 224), (2048, 256), (3072, 256).$$

2. N ビットのランダムな素数 $q\,(2^{N-1} < q < 2^N)$, L ビットのランダムな素数 $p\,(2^{L-1} < p < 2^L)$ を選ぶ. ただし, q は $p-1$ を割り切る素数とする.
3. ランダムな整数 $a < p-1$ に対し,

$$g \equiv a^{\frac{p-1}{q}} \pmod{p}$$

   を計算する. ただし, $g = 1$ となる場合には, 再度 a を選択しなおし, Step 3 を実行する.
4. ランダムな整数 $x\,(1 < x < q)$ に対し,

$$y \equiv g^x \pmod{p}$$

   を計算する.

5. p, q, g は専用のパラメータとして, y は検証鍵として公開する. x は署名鍵として安全に管理する.

署名生成・検証には少なくとも N ビットの出力をもつ暗号学的ハッシュ関数が必要となる. 任意長のメッセージ M に対する署名は, パラメータ p, q, g, 秘密鍵 x, ハッシュ関数 H を用いて次のように生成する.

署名生成

1. $1 < k < q$ の範囲からランダムに整数 k を選ぶ.
2. $r \equiv (g^k \pmod p) \pmod q$ を計算する.
3. $s \equiv k^{-1}(H(M) + xr) \pmod q$ を計算する.
4. $r = 0$ もしくは $s = 0$ の場合, Step 1 まで戻り k を選び直し, 再計算を行う.
5. Step 3, 4 で計算した (r, s) のペアをメッセージ M に対する署名とし, M にとともに受信者に送付する.

メッセージ M に対する署名 (r, s) を検証するには, パラメータ p, q, g, 公開鍵 y, ハッシュ関数 H を用いて, 以下のアルゴリズムを実行する.

署名検証

1. $0 < r < q$ かつ $0 < s < q$ であるかを調べる. これ以外であれば署名は棄却する.
2. $w \equiv s^{-1} \pmod q$ を計算する.
3. $u_1 \equiv wH(M) \pmod q$ を計算する.
4. $u_2 \equiv rw \pmod q$ を計算する.
5. $v \equiv (g^{u_1} y^{u_2} \pmod p) \pmod q$ を計算する.
6. $r \equiv v \pmod q$ ならば署名を受理する. そうでなければ署名は不正とみなし, 棄却する.

正当なメッセージと署名の組 $(M, (r, s))$ に対し,

$$g^{u_1} y^{u_2} \equiv g^{u_1 + x u_2} \equiv g^{(H(M)+xr)w} \equiv g^k \pmod p$$

が成り立つ. これは

$$\begin{aligned} v &\equiv (g^{u_1} y^{u_2} \pmod p) \pmod q \\ &\equiv (g^k \pmod p) \pmod q \\ &\equiv r \pmod q \end{aligned}$$

であることを示している.

ここで, メッセージや署名に対し改ざんが行われたとしよう. 署名生成にはメッセージのハッシュ値 $H(M)$ が使われており, ハッシュ関数の理想的性質である衝突困難性により, $H(M) = H(M')$ となるような $M'(\neq M)$ を見つけることは困難である. 従って, 改ざんしたメッセージ M' のハッシュ値は $H(M') \neq H(M)$ となり, $v \equiv r \pmod{q}$ が成立しない. また, メッセージ M をそのままにして署名だけを改ざんしても, 署名 (r, s) には $H(M)$ が埋め込まれているので, やはり $v \equiv r \pmod{q}$ は成立しない. したがって, 署名検証アルゴリズムの Step 6 で署名が受理された場合, 署名 (r, s) に対してもメッセージ M に対しても改ざんは行われていないといえる.

1.2 RSA 署名

RSA を利用したデジタル署名スキームは, **メッセージ復元型**と**署名付録型** (単に付録型, あるいは署名添付型と呼ぶこともある) の 2 つに分類することができる. 本節では, まず最初に RSA 暗号のアルゴリズムを解説し, それを踏まえて 2 種類の RSA 署名のアルゴリズムを紹介する.

RSA 暗号は, 1977 年に Rivest, Shamir, Adleman によって提案された公開鍵暗号である. 公開鍵暗号スキームは, 鍵生成アルゴリズム (メッセージの受信者が実行), 暗号化アルゴリズム (メッセージの送信者が実行), 復号アルゴリズム (メッセージの受信者が実行) の 3 つのアルゴリズムから構成される. RSA 暗号におけるこれら 3 つのアルゴリズムを以下に示す.

鍵生成

1. 異なる 2 つの大きな素数 p, q をランダムに選び, $n = pq$ を計算する.
2. $\phi(n) = (p-1)(q-1)$ を計算し, $\phi(n)$ と互いに素な整数 e を選ぶ.
3. $\phi(n)$ を法とする e のモジュラ逆数 d を計算し, n, e を公開鍵として公開する. d は秘密鍵として安全に管理する.

暗号化

メッセージ M に対し

$$C = M^e \pmod{n}$$

を計算し, M の暗号文として C を受信者に送信する.

復号

暗号文 C に対し次式を計算することでメッセージ M を得る.

$$M \equiv C^d \pmod{n}.$$

上記のアルゴリズムがなぜ機能するかを示すために, Fermat の小定理 (定理 1.2.1) およびその一般化である Euler の定理 (定理 1.2.2) を紹介しておく.

定理 1.2.1 (Fermat の小定理)**.** p を素数, a を p と互いに素な整数とする. このとき, 以下の合同式が成り立つ.

$$a^{p-1} \equiv 1 \pmod{p}.$$

定理 1.2.2 (Euler の定理)**.** a, n を互いに素な整数とする. このとき, 以下の合同式が成り立つ.

$$a^{\phi(n)} \equiv 1 \pmod{n}.$$

ここで, ϕ は Euler のトーシェント関数を表す. すなわち, $\phi(n)$ は n と互いに素な正の整数の個数を表す.

RSA 暗号のアルゴリズムがなぜ機能するかは, 以下のようにして確かめられる ([9] 等参照).
(a) $\gcd(M, n) = 1$ のとき, (b) $\gcd(M, n) > 1$ のとき, の 2 つの場合に分けて考える.
(a) $\gcd(M, n) = 1$ のとき. Euler の定理 (定理 1.2.2) が $M^{\phi(n)} \equiv 1 \pmod{n}$ を保証するので, 以下が成立する.

$$C^d = M^{ed} = M^{k\phi(n)+1} \equiv M \pmod{n}. \tag{1.1}$$

ただし, k は $ed = k\phi(n) + 1$ を満たす正整数である.
(b) $\gcd(M, n) > 1$ のとき. 一般性を失うことなく $\gcd(M, p) = p, \gcd(M, q) = 1$ の場合を考える. このとき,

$$M^{k\phi(n)+1} - M \equiv 0 \pmod{p}$$

が成立つのは明らかである. また, Fermat の小定理 (定理 1.2.1) より, $M^{q-1} \equiv 1 \pmod{q}$ が成立つので

$$M^{k\phi(n)+1} - M \equiv 0 \pmod{q}$$

も成立つ. $\gcd(p, q) = 1$ であることから, ただちに

$$M^{k\phi(n)+1} - M \equiv 0 \pmod{pq}$$

を得る. すなわち, 式 (1.1) が成立する.
RSA 暗号の安全性は, 素因数分解問題の困難性に基づく. すなわち, $n = pq$ を素因数分解することは計算量的に困難であることが, RSA 暗号の安全性の根拠となっている.
ここからは単純化した 2 種類の RSA 署名について説明する.

メッセージ復元型

メッセージ復元型は, 署名からメッセージ全体または一部を復元可能とする方式であり, 署名検証の際にはメッセージを必要とせず, 署名対象データのフォーマットの正当性によって検証を行うものである. RSA 署名を利用したメッセージ復元型の署名方式としては, ISO/IEC 9796-2,3 [2][1] や PSS-R 署名 [4] 等が挙げられる.
メッセージ復元型のもっとも単純なアルゴリズムは, メッセージを RSA の秘密鍵 d でべき乗することで署名を生成するというものである. この方式における署名生成, 検証のアルゴリ

ズムを以下に示す. なお, 鍵生成アルゴリズム (署名者が実行) は RSA 暗号と全く同じであるため, 省略する. また, 攻撃がなければ署名として正しく機能することは, RSA 暗号が機能することと同様に証明できる.

署名生成

1. メッセージ M に対し, 署名者 (メッセージ作成者) の秘密鍵 d を用いて

$$S \equiv M^d \pmod{n}$$

   を計算する.
2. Step 1 で計算した S をメッセージ M に対する署名とし, 受信者に送付する.

署名検証

受信者は, 署名者の公開鍵 e を用いて

$$S^e \pmod{n}$$

を計算し, もっともらしいメッセージ M' が得られた場合, 署名を受理する. それ以外の場合, 署名を棄却する.

この単純な方式では, 署名の真正性 (実際には, メッセージの完全性) を正しく検証できない. なぜなら, 得られたメッセージのもっともらしさの評価は受信者の判断に委ねられているためである. したがって, 例えばメッセージ M が「銀行口座の残高を$1,000,000 に増やす」という内容であった場合, 受信者が「$1,000 に増やす」というメッセージを得たとしても, それは真正な署名として受理されてしまう. このような理由から, メッセージ復元型の利用は推奨されない. この問題を回避する方式が, 次に述べる付録型の RSA 署名である.

署名付録型

署名生成者が署名をメッセージとともに送信し, 署名検証者は受信したメッセージを利用して署名の正当性を検証する方式である.

付録型のもっとも単純なアルゴリズムは, メッセージのハッシュ値を生成し, そのハッシュ値を RSA の秘密鍵 d によりべき乗することで署名を生成するというものである. この方式における署名生成, 検証のアルゴリズムを以下に示す. なお, 鍵生成アルゴリズム (署名者が実行) は RSA 暗号と同様である.

署名生成

1. メッセージ M のハッシュ値 $h = H(M)$ を計算する. ただし, H はハッシュ関数である.

2. メッセージ作成者の秘密鍵 d を用いて

$$S \equiv h^d \pmod{n}$$

を計算する.

3. Step 2 で計算した S をメッセージ M に対する署名とし, M とともに受信者に送付する.

署名検証

受信者は, 署名者 (メッセージ作成者) の公開鍵 e を用いて以下の合同式が成立するか否かを検証する.

$$H(M) \equiv S^e \pmod{n}.$$

上記はもっとも単純化した付録型 RSA 署名のアルゴリズムであるが, RSA 署名において署名の対象となるデータ (署名対象データ) として, 単にメッセージのハッシュ値を利用するのではなく, メッセージになんらかの変換を加えたデータ, 乱数, 予め決められたパディングデータ等を結合させて生成する方式も提案されている. こうした方式としては, PKCS#1[5] や PSS 署名 [4] 等が挙げられる.

なお, DSA 署名 (1.1 節) は $H(M)$ が埋め込まれた値 s を送信するため付録型であり, DSA 署名の復元型は存在しない.

1.3 ハイブリッド暗号

ハイブリッド暗号とは, 公開鍵暗号と共通鍵暗号を組み合わせた暗号方式である. ハイブリッド暗号は, 公開鍵暗号の計算量の多さと共通鍵暗号の鍵配送の問題を解決するために考案された.

ハイブリッド暗号の流れを図 1.1 に示す. 送信者ボブは, 次の手順でメッセージとメッセージの暗号化に用いた共通鍵をアリスに送信する.

1. メッセージ暗号化用の共通鍵を生成する.
2. Step 1 で生成した共通鍵を用いてアリス宛のメッセージを暗号化する.
3. アリスの公開鍵を用いて, Step 1 で生成した共通鍵を暗号化する.
4. Step 3 で暗号化した共通鍵と Step 2 で暗号化したメッセージをアリスに送信する.

受信者アリスは, 以下の手順で共通鍵とメッセージを復号する.

5. 暗号化された共通鍵を自身の秘密鍵で復号する.
6. Step 5 で得られた共通鍵で暗号文を復号する.

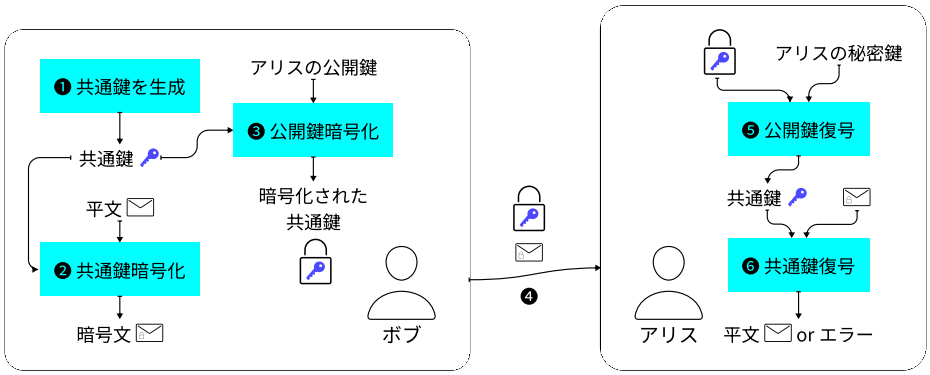

図 1.1: ボブがアリスにハイブリッド暗号を使用してメッセージを送信する流れ

1.4 平方剰余

本節では, 平方剰余の定義と性質を与える.

定義 1.4.1 (平方剰余)**.** p を奇素数, a を p で割り切れない整数とする. 合同式

$$x^2 \equiv a \pmod{p}$$

が整数解をもつとき, a は p を法として**平方剰余** (quadratic residue) であるという. 一方で, このような x が存在しないとき, a は p を法として**平方非剰余** (non-quadratic residue) であるという.

任意の正整数 a と奇素数 p に対し, a が $\mathbb{F}_p$ において平方剰余か否かを表す記号として, 以下で定義される **Legendre 記号**が知られている.

定義 1.4.2 (Legendre 記号)**.** 奇素数 p と $a \in \mathbb{N}$ に対し, 以下で定義される関数を Legendre 記号という.

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & a \text{ が } p \text{ を法として平方剰余かつ } p \nmid a, \\ -1 & a \text{ が } p \text{ を法として平方非剰余}, \\ 0 & p \mid a. \end{cases}$$

また, Legendre 記号の一般化である **Jacobi 記号**は次のように定義される.

定義 1.4.3 (Jacobi 記号)**.** 奇数 $n \geq 3$ と $a \in \mathbb{N}$ に対し, Jacobi 記号 $\left(\frac{a}{n}\right)$ は n の素因数に対応する Legendre 記号の積として定義される.

$$\left(\frac{a}{n}\right) = \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)^{\alpha_i}.$$

ただし, n の素因数分解を

$$n = \prod_{i=1}^{k} p_i^{\alpha_i}$$

とする. n が奇素数である場合, Jacobi 記号は Legendre 記号に一致する.

以下の規準を用いると, Legendre 記号は直ちに計算できる.

補題 1.4.4 (Eular 規準)**.** p を奇素数とする. $a \in \mathbb{Z}_p \backslash \{0\} (= \mathbb{Z}_p^*)$ に対し, 次式が成り立つ.

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}.$$

次に紹介する**平方剰余の相互法則**とその補充法則は, 整数 a が奇素数 p を法として平方剰余であるか否かを判定するための良く知られた法則である.

補題 1.4.5 (平方剰余の相互法則)**.** 異なる奇素数 p, q に対し,

$$\left(\frac{q}{p}\right)\left(\frac{p}{q}\right) = (-1)^{\frac{p-1}{2}\cdot\frac{q-1}{2}}$$

が成り立つ.

平方剰余の相互法則には 2 つの補充法則が知られている.

補題 1.4.6 (平方剰余の相互法則 (第 1 補充法則))**.** p を奇素数とする. このとき,

$$\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}}$$

が成り立つ. つまり,

$$\left(\frac{-1}{p}\right) = \begin{cases} 1 & (p \equiv 1 \pmod 4 \text{ のとき}), \\ -1 & (p \equiv 3 \pmod 4 \text{ のとき}). \end{cases}$$

証明. (i) $p = 4k+1$ $(k \in \mathbb{N})$ のとき, 補題 1.4.4 より,

$$\left(\frac{-1}{p}\right) \equiv (-1)^{\frac{p-1}{2}} = (-1)^{2k} = 1 \pmod{p}$$

が成り立つ.

(ii) $p = 4k+3$ $(k \in \mathbb{N} \cup \{0\})$ のときも同様に, 補題 1.4.4 より,

$$\left(\frac{-1}{p}\right) \equiv (-1)^{\frac{p-1}{2}} = (-1)^{2k+1} = -1 \pmod{p}$$

が成り立つ. 以上より, 主張は証明された. □

補題 1.4.7 (平方剰余の相互法則 (第 2 補充法則))**.** p を奇素数とする. このとき,

$$\left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}}$$

が成り立つ. つまり,

$$\left(\frac{2}{p}\right) = \begin{cases} 1 & (p \equiv \pm 1 \pmod 8 \text{ のとき}), \\ -1 & (p \equiv \pm 3 \pmod 8 \text{ のとき}). \end{cases}$$

証明. 一般に, 以下の式が成り立つことに注意する. なお, ${}_pC_j = {}_pC_{p-k}$ である.

$$(x^2+1)^p = \sum_{j=0}^{p} {}_pC_j x^{2j} = \sum_{j=0}^{\frac{p-1}{2}} {}_pC_j x^{2j} + \sum_{k=0}^{\frac{p-1}{2}} {}_pC_{p-k} x^{2(p-k)} = \sum_{j=0}^{\frac{p-1}{2}} {}_pC_j (x^{2j} + x^{2p-2j}).$$

これを x^p で割ると

$$(x+x^{-1})^p = \sum_{j=0}^{\frac{p-1}{2}} {}_pC_j (x^{p-2j} + x^{-(p-2j)}) \tag{1.2}$$

を得る.

また, 1 の 8 乗根を η とおくと (つまり $\eta^8 = 1$ が成り立つとすると),

$$\eta = e^{\frac{2\pi i}{8}} = \cos\frac{2\pi}{8} + i\sin\frac{2\pi}{8} = \frac{1}{\sqrt{2}} + i\frac{1}{\sqrt{2}}$$

である. η^{-1} は η の複素共役であり, $\eta^2 = i$ であることから, $\eta^3 = -1/\sqrt{2} + i(1/\sqrt{2})$ となることに注意すると,

$$\eta + \eta^{-1} = \sqrt{2}, \quad \eta^3 + \eta^{-3} = -\sqrt{2}$$

が成り立つ. $\eta^8 = 1$ より, 任意の $n \in \mathbb{N}$ に対し,

$$\eta^n + \eta^{-n} = \begin{cases} \eta + \eta^{-1} = \sqrt{2} & (n \equiv \pm 1 \pmod 8 \text{ のとき}), \\ \eta^3 + \eta^{-3} = -\sqrt{2} & (n \equiv \pm 3 \pmod 8 \text{ のとき}), \end{cases}$$

すなわち,

$$\eta^n + \eta^{-n} = (-1)^{\frac{n^2-1}{8}}\sqrt{2}$$

が成り立つ. 以上のことに注意し, 式 (1.2) の x に η を代入してから両辺を $\sqrt{2}$ で割ると,

$$\begin{aligned} \frac{(\eta+\eta^{-1})^p}{\sqrt{2}} &= \frac{1}{\sqrt{2}} \sum_{j=0}^{\frac{p-1}{2}} {}_pC_j (\eta^{p-2j} + \eta^{-(p-2j)}) \\ (\sqrt{2})^{p-1} &= \frac{1}{\sqrt{2}} \sum_{j=0}^{\frac{p-1}{2}} {}_pC_j (-1)^{\frac{(p-2j)^2-1}{8}}\sqrt{2} \\ 2^{\frac{p-1}{2}} &= \sum_{j=0}^{\frac{p-1}{2}} {}_pC_j (-1)^{\frac{(p-2j)^2-1}{8}} \equiv (-1)^{\frac{p^2-1}{8}} \pmod p \end{aligned} \tag{1.3}$$

を得る. よって, 式 (1.3) と補題 1.4.4 より,

$$\left(\frac{2}{p}\right) \equiv 2^{\frac{p-1}{2}} \equiv (-1)^{\frac{p^2-1}{8}} \pmod p$$

が成り立つ. 以上より, 主張は証明された. □

1.5 ブラム数を法とする 2 次合同式の解法

この節では, 後述する Batten and Williams[3] による暗号プリミティブに深く関与する, 2 次合同式の解法について述べる. その準備として, 最初に中国の剰余定理 (Chinese Remainder Theorem, CRT) を紹介しておく.

定理 1.5.1 (CRT)**.** $m_1, m_2, \ldots m_n$ を互いに素な 2 以上の自然数とし, $a_1, a_2, \ldots, a_n$ を任意の整数とする. このとき, 連立合同式

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \quad \vdots \\ x \equiv a_n \pmod{m_n} \end{cases} \tag{1.4}$$

を満たす $x \in \mathbb{Z}$ が $m = \prod_{i=1}^{n} m_i$ を法として一意に存在する.

ここで,

$$M_i = \frac{m}{m_i} = \prod_{j=1, j\neq i}^{n} m_j \quad (i = 1, 2, \ldots, n) \tag{1.5}$$

とおくと,

$$\gcd(M_i, m_i) = 1 \quad (i = 1, 2, \ldots, n)$$

が成り立つ. このとき, M_i の逆元 y_i が存在する. すなわち,

$$M_i y_i \equiv 1 \pmod{m_i} \quad (i = 1, 2, \ldots, n) \tag{1.6}$$

が成り立つ. なお, y_i は拡張 Eucrid アルゴリズムを用いて求めることができる.

以下は, CRT (定理 1.5.1) により, 連立合同式 (1.4) の解が求められることを示したよく知られた補題である.

補題 1.5.2 (連立合同式の解)**.** 連立合同式 (1.4) の解は次式で与えられる.

$$x \equiv \left(\sum_{i=1}^{n} a_i y_i M_i \right) \pmod{m}.$$

ここで, M_i, y_i はそれぞれ式 (1.5), 式 (1.6) で定義される値である.

証明. 式 (1.6) の両辺に a_i を掛けると,

$$a_i y_i M_i \equiv a_i \pmod{m_i} \quad (i = 1, 2, \ldots, n) \tag{1.7}$$

を得る. また, $i \neq j$ なる i, j について $\gcd(m_i, M_j) = m_i$ であるので,

$$a_j y_j M_j \equiv 0 \pmod{m_i} \quad (1 \leq i \neq j \leq n) \tag{1.8}$$

である. 式 (1.7) と式 (1.8) より, $i = 1, 2, \ldots, n$ について,

$$x \equiv a_i \equiv \sum_{j=1}^{n} a_j y_j M_j \pmod{m_i}$$

が成り立ち, CRT (定理 1.5.1) より,

$$x \equiv \sum_{j=1}^{n} a_j y_j M_j \pmod{m}$$

を得る. □

補題 1.5.2 を $n = 2$ のときに限定すると, 次の系が直ちに得られる.

系 1.5.3. 互いに素な 2 以上の自然数 s, t に関する連立合同式

$$\begin{cases} x \equiv a \pmod{s} \\ x \equiv b \pmod{t} \end{cases}$$

の解は

$$x \equiv a y_a t + b y_b s \pmod{st}$$

で与えられる. ただし, y_a, y_b は以下を満たす整数である.

$$y_a t \equiv 1 \pmod{s},$$
$$y_b s \equiv 1 \pmod{t}.$$

合成数 n が**ブラム数** (2 つの異なる $3 \pmod 4$ を満たす素因数に分解される合成数) の場合には, 2 次合同式の解の個数が定まることが知られている [7].

補題 1.5.4. $p, q \equiv 3 \pmod 4$ を異なる素数とし, $n = pq$ とする. $0 < a < n$ なる整数 a に対し,

$$x^2 \equiv a \pmod{n}$$

に解が存在するならば, それらすべての解は次式で与えられる.

$$x \equiv \pm(z q q_p^{-1} \pm w p p_q^{-1}) \pmod{n}. \tag{1.9}$$

ただし,

$$z \equiv a^{\frac{p+1}{4}} \pmod{p}, \quad w \equiv a^{\frac{q+1}{4}} \pmod{q},$$
$$q_p^{-1} \equiv q^{-1} \pmod{p}, \quad p_q^{-1} \equiv p^{-1} \pmod{q}$$

である.

証明. $x^2 \equiv a \pmod n$ より, $x^2 \equiv a \pmod p$ が成り立つ. $x^2 \equiv a \pmod p$ の解を $\pm z$ とすると, z は以下で与えられる.

$$z \equiv a^{\frac{p+1}{4}} \pmod{p}.$$

なぜなら, a は p を法とする平方剰余なので,

$$a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$$

が成り立つからである. この式の両辺に a を掛けると

$$a^{\frac{p+1}{2}} \equiv a \pmod{p}$$

となるので, $x^2 \equiv a \pmod{p}$ より, $x^2 \equiv a^{\frac{p+1}{2}} \pmod{p}$ が言える. よって,

$$x \equiv \pm a^{\frac{p+1}{4}} \pmod{p}$$

を得る. 同様に, $x^2 \equiv a \pmod{q}$ の解も導くことができる.

$x^2 \equiv a \pmod{q}$ の 解を $\pm w$ とすると, 2 次合同式 $x^2 \equiv a \pmod{n}$ の解は

$$\begin{cases} x \equiv z \pmod{p}, \\ x \equiv w \pmod{q}, \end{cases} \quad \begin{cases} x \equiv -z \pmod{p}, \\ x \equiv -w \pmod{q}, \end{cases}$$

$$\begin{cases} x \equiv -z \pmod{p}, \\ x \equiv w \pmod{q}, \end{cases} \quad \begin{cases} x \equiv z \pmod{p}, \\ x \equiv -w \pmod{q}, \end{cases}$$

の 4 通りのいずれかとなる. これらに CRT (定理 1.5.1) を適用すると, 式 (1.9) が得られる. □

以上により, CRT (定理 1.5.1) を用いてブラム数 n を法とした 2 次合同式 $x^2 \equiv a \pmod{n}$ の解を求めるアルゴリズムは, p と q の平方根を求めるアルゴリズムと同じ計算量で実行できることがわかる. また, 2 次合同式 $x^2 \equiv a \pmod{n}$ の解の個数は, a が n を法として平方剰余であるときには 4 個, そうでないときには 0 個となる.

第2章　関連研究

この章では, Rabin 暗号 [7] および Rabin-Williams 暗号 [8] に関する関連研究について解説する. Rabin 暗号は 1979 年に提案されてから今日まで様々な改良がなされており, それらの中で主要な先行研究を紹介する. 本研究では, 特に 2.3 節で解説する先行研究を参考にして署名スキームの提案を試みた.

2.1　Rabin 暗号

Rabin 暗号 [7] とは, 1979 年に Michael Oser Rabin が発表した, 選択平文攻撃で解読することと素因数分解問題を解くことが等価であることが証明された最初の公開鍵暗号である. Rabin 暗号は, 鍵となる合成数が素因数分解できない限り, 少なくとも選択平文攻撃による暗号解読に対して安全であることが証明されている. しかし, 選択暗号文攻撃に対しては安全でないことも証明されている. そのため, 証明可能安全性を有するという意味で暗号理論的な意義は大きいが, そのまま使用することは推奨されない.

Rabin 暗号は, 鍵生成 (メッセージの受信者が実行), 暗号化 (メッセージの送信者が実行), 復号 (メッセージの受信者が実行) の 3 つのアルゴリズムから構成される. 以下に, Rabin 暗号のアルゴリズムを示す.

鍵生成

1. 2 つの異なる素数 p と q を用意し, $n = pq$ と計算する.
2. n を公開鍵とし, p, q を秘密鍵とする.

暗号化

公開鍵 n を用いて, メッセージ M に対する暗号文を $C \equiv M^2 \pmod{n}$ と計算する.

なお, [7] では暗号化関数の出力を

$$C \equiv M(M + B) \pmod{n}$$

としているが, B は単純に 0 としても Rabin 暗号の本質を失わないため, 本稿では省略する.

復号

1. 暗号文 C と秘密鍵 p, q を用いて, 次式のように z, w を計算する.

$$z \equiv C^{\frac{p+1}{4}} \pmod{p},$$
$$w \equiv C^{\frac{q+1}{4}} \pmod{q}.$$

2. p を法とする q の逆元 q_p^{-1} と, q を法とする p の逆元 p_q^{-1} を計算する. すなわち, 次式を満たす q_p^{-1} と p_q^{-1} を計算する.

$$q_p^{-1} q \pmod{p} \equiv 1,$$
$$p_q^{-1} p \pmod{q} \equiv 1.$$

3. Step 1 で求めた z, w から定まる 4 組の (a, b)

$$(a, b) \in \{(z, w), (p - z, w), (z, q - w), (p - z, q - w)\}$$

に対し,

$$M \equiv a q_p^{-1} q + b p_q^{-1} p \pmod{pq}$$

を計算する.

復号アルゴリズムが示すように, M として 4 個の解が求まるが, 暗号化が単射ではないため, 一意に平文を求めることができないことに注意する. 4 個の解のうちどれがもとの平文であるかを特定するには, 付加情報を送信する等の追加処理が必要となる.

Rabin 暗号の例を示す.

例 2.1.1. メッセージ受信者は鍵生成のため, 異なる 2 つの素数として $p = 11, q = 19$ を選んだとする. この場合, $n = 209$ である.

メッセージ送信者は, 平文 $M = 123$ を以下のように暗号化する.

$$81 \equiv 123^2 \pmod{209}.$$

暗号文の受信者は次のように復号を行う. まず z と w を計算する.

$$z \equiv 81^{\frac{11+1}{4}} \equiv 9 \pmod{11},$$
$$w \equiv 81^{\frac{19+1}{4}} \equiv 9 \pmod{19}.$$

次に, 次式を満たす q_p^{-1} と p_q^{-1} を求める.

$$19 q_p^{-1} \equiv 1 \pmod{11},$$
$$11 p_q^{-1} \equiv 1 \pmod{19}.$$

これらはそれぞれ,

$$19 q_p^{-1} + 11k = 1,$$
$$11 p_q^{-1} + 19l = 1$$

に拡張 Eucrid アルゴリズムを適用することで求められる. ここで, k, l は整数である. まず q_p^{-1} を求める.

$$\begin{aligned} 19 &= 11 \times 1 + 8. \\ 11 &= 8 \times 1 + 3. \\ 8 &= 3 \times 2 + 2. \\ 3 &= 2 \times 1 + 1. \end{aligned}$$

剰余が 1 になったため, 過程を逆に辿る.

$$\begin{aligned} 1 &= 3 - 2 \times 1 \\ &= 3 - (8 - 3 \times 2) \\ &= 3 \times 3 - 8 \\ &= 3 \times (11 - 8 \times 1) - 8 \\ &= 3 \times 11 - 8 \times 4 \\ &= 3 \times 11 - (19 - 11 \times 1) \times 4 \\ &= 7 \times 11 - 4 \times 19. \end{aligned}$$

したがって, $q_p^{-1} = 7$ を得る. p_q^{-1} に関しても同様に計算する.

$$\begin{aligned} 11 &= 19 \times 0 + 11. \\ 19 &= 11 \times 1 + 8. \end{aligned}$$

ここで先と同じ形になるため, 途中の計算を省略する.

$$1 = 7 \times 11 - 4 \times 19.$$

から, $p_q^{-1} = 7$ を得る.

以上より, 復号アルゴリズムの Step 3 にある (a, b) の候補は次の 4 通りとなる.

$$(a, b) \in \{(9, 9), (2, 9), (9, 10), (2, 10)\}.$$

最後に q_p^{-1}, p_q^{-1} および (a, b) を利用して, M の候補を求める. まず $(a, b) = (9, 9)$ について,

$$M \equiv 9 \times 7 \times 19 + 9 \times 7 \times 11 \equiv 9 \pmod{209}.$$

残りの 3 つについても同様に計算する. $(a, b) = (2, 9)$ について,

$$M \equiv 2 \times 7 \times 19 + 9 \times 7 \times 11 \equiv 123 \pmod{209}.$$

$(a, b) = (9, 10)$ について,

$$M \equiv 9 \times 7 \times 19 + 10 \times 7 \times 11 \equiv 86 \pmod{209}.$$

$(a, b) = (2, 10)$ について,

$$M \equiv 2 \times 7 \times 19 + 10 \times 7 \times 11 \equiv 200 \pmod{209}.$$

よって, 平文 M の候補の集合は $\{9, 123, 86, 200\}$ である. このうち, $M = 123$ が求める平文であるが, Rabin 暗号ではこの特定まではできない.

2.2 Rabin-Williams 暗号および署名

Rabin が Rabin 暗号のアイデア [7] を発表した翌年に, Williams[8] は Rabin 暗号のアイデアを使用した一意復号可能な暗号スキームを提案し, 署名スキームへの応用も示唆した. [8] では, その解読がモジュラス n の因数分解と同等に困難あることも示されている.

2.2.1 暗号化および復号

ここでは, Rabin-Williams 暗号の鍵生成, 暗号化および復号を行うアルゴリズムを, [8] に基づき解説する.

アルゴリズム

メッセージの受信者は, 次の手順で鍵生成を行う.

鍵生成

1. 2 つの大きな素数 $p \equiv 3 \pmod 8, q \equiv 7 \pmod 8$ を選び, $R = pq$ を計算する.
2. 次式を満たすような e を選択する.

$$\gcd(e, \lambda(R)) = 1 \tag{2.1}$$

   ただし,

$$\lambda(R) = \mathrm{lcm}\,(p-1, q-1)$$

   である.
3. 合同式

$$ed \equiv m \pmod{\lambda(R)} \tag{2.2}$$

   を満たす d を計算する. ただし,

$$m = \frac{\frac{(p-1)(q-1)}{4} + 1}{2} \tag{2.3}$$

   である.
4. (R, e) を公開鍵として公開し, d を秘密鍵として秘匿する.

メッセージの送信者は, 次の手順で暗号化を行う.

暗号化

1. 公開鍵 (R, e) を用いて, 平文 $M \in \mathcal{M}$ に対し, 次式を計算する.

$$K = E(M) = E_2 \circ E_1(M).$$

ただし, 関数 E_1, E_2 は次の通りである.

$$E_1(M) = \begin{cases} 4(2M+1) & (\left(\frac{2M+1}{R}\right) = 1 \text{ のとき}), \\ 2(2M+1) & (\left(\frac{2M+1}{R}\right) = -1 \text{ のとき}), \end{cases}$$
$$E_2(N) \equiv N^{2e} \pmod{R}.$$

また, $\mathcal{M}$ は, 以下を満たす正の整数 M からなる集合である.

$$M \in \mathcal{M} \iff \begin{cases} 4(2M+1) < R & (\left(\frac{2M+1}{R}\right) = 1 \text{ のとき}), \\ 2(2M+1) < R & (\left(\frac{2M+1}{R}\right) = -1 \text{ のとき}). \end{cases}$$

2. Step 1 で計算した K を暗号文として出力する.

メッセージの受信者は, 次の手順で復号を行う.

復号化

暗号文 K, 秘密鍵 d, 公開鍵 R を用いて次式を計算することでメッセージ M を得る.

$$M = D(K) = D_1 \circ D_2(K).$$

ただし, 関数 D_1, D_2 は次の通りである.

$$D_1(L) = \begin{cases} \dfrac{\frac{L}{4} - 1}{2} & (L \equiv 0 \pmod 4 \text{ のとき}), \\ \dfrac{\frac{R-L}{4} - 1}{2} & (L \equiv 1 \pmod 4 \text{ のとき}), \\ \dfrac{\frac{L}{2} - 1}{2} & (L \equiv 2 \pmod 4 \text{ のとき}), \\ \dfrac{\frac{R-L}{2} - 1}{2} & (L \equiv 3 \pmod 4 \text{ のとき}), \end{cases}$$
$$D_2(K) \equiv K^d \pmod{R}.$$

Rabin-Williams 暗号が正しく機能することを示すため, 補題を用意しておく.

補題 2.2.1 ([8, Lemma 1])**.** p, q を $3 \pmod 4$ を満たす異なる素数とし, $R = pq$ とおく. このとき, $\left(\frac{M}{R}\right) = 1$ なる M に対し, 次式が成立する.

$$M^{\frac{(p-1)(q-1)}{4}} \equiv \pm 1 \pmod{R}.$$

証明. $\left(\frac{M}{R}\right) = 1$ なので, $\left(\frac{M}{p}\right) = \left(\frac{M}{q}\right)$ である. 2 つの場合に分けて考える.

(i) $\left(\frac{M}{p}\right) = \left(\frac{M}{q}\right) = 1$ のとき,

$$M^{\frac{(p-1)}{2}} \equiv 1 \pmod{p},$$
$$M^{\frac{(q-1)}{2}} \equiv 1 \pmod{q}$$

なので,

$$(M^{\frac{p-1}{2}})^{\frac{q-1}{2}} = M^{\frac{(p-1)(q-1)}{4}} \equiv 1 \pmod{p},$$
$$(M^{\frac{q-1}{2}})^{\frac{p-1}{2}} = M^{\frac{(p-1)(q-1)}{4}} \equiv 1 \pmod{q}$$

である. よって,

$$M^{\frac{(p-1)(q-1)}{4}} \equiv 1 \pmod{R}$$

となる.

(ii) $\left(\frac{M}{p}\right) = \left(\frac{M}{q}\right) = -1$ のとき,

$$M^{\frac{(p-1)}{2}} \equiv -1 \pmod{p},$$
$$M^{\frac{(q-1)}{2}} \equiv -1 \pmod{q}$$

である. 仮定より $\frac{p-1}{2}, \frac{q-1}{2}$ は奇数なので,

$$M^{\frac{(p-1)(q-1)}{4}} \equiv -1 \pmod{p},$$
$$M^{\frac{(p-1)(q-1)}{4}} \equiv -1 \pmod{q}$$

を得る.

よって題意は示された. □

定理 2.2.2 ([8, Theorem 1]). p, q を以下を満たす素数とし, $R = pq$ とおく.

$$p \equiv 3 \pmod{8}, \qquad q \equiv 7 \pmod{8}.$$

また e, d は式 (2.1), 式 (2.2) を満たすとする. 以下のように定義される関数 E_1, E_2, D_1, D_2

$$E_1(M) = \begin{cases} 4(2M+1) & (\left(\frac{2M+1}{R}\right) = 1 \text{ のとき}), \\ 2(2M+1) & (\left(\frac{2M+1}{R}\right) = -1 \text{ のとき}), \end{cases}$$
$$E_2(N) \equiv N^{2e} \pmod{R},$$
$$D_1(L) = \begin{cases} \frac{\frac{L}{4}-1}{2} & (L \equiv 0 \pmod 4 \text{ のとき}), \\ \frac{\frac{R-L}{4}-1}{2} & (L \equiv 1 \pmod 4 \text{ のとき}), \\ \frac{\frac{L}{2}-1}{2} & (L \equiv 2 \pmod 4 \text{ のとき}), \\ \frac{\frac{R-L}{2}-1}{2} & (L \equiv 3 \pmod 4 \text{ のとき}), \end{cases}$$
$$D_2(K) \equiv K^d \pmod{R}$$

に対して, $M \in \mathcal{M}$ ならば次式が成立する.

$$D_1 \circ D_2 \circ E_2 \circ E_1(M) = M. \tag{2.4}$$

ただし, $\mathcal{M}$ は以下を満たす正の整数 M からなる集合である.

$$M \in \mathcal{M} \iff \begin{cases} 4(2M+1) < R & (\left(\frac{2M+1}{R}\right) = 1 \text{ のとき}), \\ 2(2M+1) < R & (\left(\frac{2M+1}{R}\right) = -1 \text{ のとき}). \end{cases}$$

証明. $N = E_1(M)$ とおく. E_1 の定義から N は偶数であり, $0 < N < R$ である. 最初に, $\left(\frac{N}{R}\right)$ について考える.

(i) $\left(\frac{2M+1}{R}\right) = 1$ の場合, $N = 4(2M+1)$ である. $4(< R)$ が平方剰余であることは明らかであり, 前提より $2M+1$ は平方剰余なので N は平方剰余である.

(ii) $\left(\frac{2M+1}{R}\right) = -1$ の場合, $N = 2(2M+1)$ である. 前提より, $2M+1$ は平方非剰余であり, $p \equiv 3 \pmod 8, q \equiv -1 \pmod 8$ であることに注意すると, 平方剰余の相互法則の第 2 補充法則 (補題 1.4.7) より $\left(\frac{2}{R}\right) = \left(\frac{2}{p}\right)\left(\frac{2}{q}\right) = -1$ であるから, 2 も平方非剰余である. よって N は平方剰余である.

したがって, (i), (ii) どちらの場合も $\left(\frac{N}{R}\right) = 1$ である. 補題 2.2.1 および式 (2.3) より

$$L = D_2 \circ E_2(N) \equiv N^{2ed} \equiv N^{2m} \equiv N^{\frac{(p-1)(q-1)}{4}+1} \equiv \pm N \pmod R$$

であり, $0 < L < R$ である. N は偶数であり R は奇数であるから, L が偶数ならば $L = N$ であり, L が奇数ならば $L = R - N$ である.

最後に, L を M に変換する関数 D_1 について, L の値を場合分けして考える.

(i) $L \equiv 0 \pmod 4$ の場合, L は偶数なので $L = N \equiv 0 \pmod 4$ である. したがって, $E_1(M)$ が取ることのできる値は

$$L = N = E_1(M) = 4(2M+1)$$

のみである. この式を M について変形すると, 次式を得る.

$$M = \frac{\left(\frac{L}{4} - 1\right)}{2} = D_1(L).$$

(ii) $L \equiv 1 \pmod 4$ の場合, L は奇数であり, $R = pq \equiv 1 \pmod 4$ であるから, $L = R - N$ より $N = R - L \equiv 0 \pmod 4$ である. したがって, (i) と同様の議論により, 次式を得る.

$$M = \frac{\left(\frac{R-L}{4} - 1\right)}{2} = D_1(L).$$

(iii) $L \equiv 2 \pmod 4$ の場合, L は偶数なので $N = L \equiv 2 \pmod 4$ である. したがって, $E_1(M)$ が取ることのできる値は

$$L = N = E_1(M) = 2(2M+1)$$

のみである. この式を M について変形すると, 次式を得る.

$$M = \frac{\left(\frac{L}{2} - 1\right)}{2} = D_1(L).$$

(iv) $L \equiv 3 \pmod 4$ の場合, L は奇数であり, $R \equiv 1 \pmod 4$ であるから, $L = R - N$ より $N = R - L \equiv 2 \pmod 4$ である. したがって, (iii) と同様の議論により, 次式を得る.

$$M = \frac{\left(\frac{R-L}{2} - 1\right)}{2} = D_1(L).$$

以上より, $M \in \mathcal{M}$ ならば式 (2.4) が成立することが示された. □

注 2.2.3 ([8, Corollary])**.** 定理 2.2.2 における式 (2.4) は,

$$D_1 \circ E_2 \circ D_2 \circ E_1(M) = M$$

と書き換えることができる. なぜなら,

$$E_2 \circ D_2(K) \equiv (K^d)^{2e} \equiv K^{2ed} \equiv D_2 \circ E_2(K) \pmod R$$

が成り立つからである.

2.2.2 署名

Williams は [8] で, 2.2.1 節で紹介した暗号スキームを署名スキームに応用することも提案している.

署名者アリスが署名検証者ボブにメッセージ M に署名を付けて送信する手順は以下の通りである.

鍵生成

1. アリスは, 2 つの大きな素数 $p_A \equiv 3 \pmod 8, q_A \equiv 7 \pmod 8$ を選び, $R_A = p_A q_A$ を計算する. また, 式 (2.1) を満たすような e を e_A, 式 (2.2) を満たすような d を d_A, 式 (2.3) を満たすような m を m_A として計算する.
2. いま, アリスは以下のように定義される関数 $E_{1A}, E_{2A}, D_{1A}, D_{2A}$ を知っている.

$$E_{1A}(M) = \begin{cases} 4(2M+1) & \left(\left(\frac{2M+1}{R_A}\right) = 1 \text{ のとき}\right), \\ 2(2M+1) & \left(\left(\frac{2M+1}{R_A}\right) = -1 \text{ のとき}\right), \end{cases}$$

$$E_{2A}(N) \equiv N^{2e_A} \pmod{R_A},$$

$$D_{1A}(L) = \begin{cases} \dfrac{\frac{L}{4} - 1}{2} & (L \equiv 0 \pmod 4 \text{ のとき}), \\ \dfrac{\frac{R_A - L}{4} - 1}{2} & (L \equiv 1 \pmod 4 \text{ のとき}), \\ \dfrac{\frac{L}{2} - 1}{2} & (L \equiv 2 \pmod 4 \text{ のとき}), \\ \dfrac{\frac{R_A - L}{2} - 1}{2} & (L \equiv 3 \pmod 4 \text{ のとき}), \end{cases}$$

$$D_{2A}(K) \equiv K^{d_A} \pmod{R_A}.$$

アリスは, 署名検証鍵として関数 (D_{1A}, E_{2A}) を公開し, 署名鍵として関数 (E_{1A}, D_{2A}) を秘匿する.

3. ボブは 2 つの大きな素数 $p_B \equiv 3 \pmod 8, q_B \equiv 7 \pmod 8$ を選び, $R_B = p_B q_B$ を計算する. また, 式 (2.1) を満たすような e を e_B, 式 (2.2) を満たすような d を d_B, 式 (2.3) を満たすような m を m_B として計算する.
4. いま, ボブは以下のように定義される関数 $E_{1B}, E_{2B}, D_{1B}, D_{2B}$ を知っている.[1]

$$E_{1B}(M) = \begin{cases} 4(2M+1) & (\left(\frac{2M+1}{R_B}\right) = 1 \text{ のとき}), \\ 2(2M+1) & (\left(\frac{2M+1}{R_B}\right) = -1 \text{ のとき}), \end{cases}$$

$$E_{2B}(N) \equiv N^{2e_B} \pmod{R_B},$$

$$D_{1B}(L) = \begin{cases} \dfrac{\frac{L}{4}-1}{2} & (L \equiv 0 \pmod 4 \text{ のとき}), \\ \dfrac{\frac{R_B-L}{4}-1}{2} & (L \equiv 1 \pmod 4 \text{ のとき}), \\ \dfrac{\frac{L}{2}-1}{2} & (L \equiv 2 \pmod 4 \text{ のとき}), \\ \dfrac{\frac{R_B-L}{2}-1}{2} & (L \equiv 3 \pmod 4 \text{ のとき}), \end{cases}$$

$$D_{2B}(K) \equiv K^{d_B} \pmod{R_B}.$$

ボブは, 暗号化鍵として関数 E_B を公開し, 復号鍵として関数 D_B を秘匿する.

署名生成

1. アリスはメッセージ $M \in \mathcal{M}$ に対する署名

$$S = D_{2A} \circ E_{1A}(M)$$

を計算する. ただし, $\mathcal{M}$ は以下を満たす正の整数 M からなる集合である.

$$M \in \mathcal{M} \iff \begin{cases} 4(2M+1) < R_A & (\left(\frac{2M+1}{R_A}\right) = 1 \text{ のとき}), \\ 2(2M+1) < R_A & (\left(\frac{2M+1}{R_A}\right) = -1 \text{ のとき}). \end{cases}$$

2. アリスはボブの公開鍵を用いて, Step 1 で生成した署名 S の暗号文

$$K = E_B(S)$$

を計算する.

3. アリスはボブに対し, 平文 M と暗号文 K を送信する.[2]

[1] これらの関数の定義は, R_B を R_A に差し替えれば, それぞれ $E_{1A}, E_{2A}, D_{1A}, D_{2A}$ に一致する.
[2] [8] ではメッセージ M を送信するステップは明記されていないが, 実際には M も署名 S とともに送信し, 署

署名検証

1. ボブは受信した暗号文 K を次式にしたがって復号する.

$$L = D_B(K).$$

2. ボブは次式が成立するか否かにより, メッセージ M がアリスにより作成された (アリスが署名した) か否かを検証し, 成立する場合には M を受理する.

$$D_{1A} \circ E_{2A}(L) = M.$$

以上の手順により, アリスはボブに対しメッセージ M に対する署名を送信できる. ボブは M に対する署名 S がアリスにしか計算できないことから, M がアリスからの正当なメッセージであることを確認できる.

なお, Willams が提案した上記の署名スキームが機能することは, 注 2.2.3 から得られる次の等式からわかる.

$$D_{1A} \circ E_{2A} \circ D_B(K) = D_{1A} \circ E_{2A} \circ D_{2A} \circ E_{1A}(M) = M.$$

2.3 一意復号可能な Rabin-Williams 暗号

2019 年に Batten and Williams[3] は, 効率的に一意復号可能な Rabin-Williams 暗号のバリエーションを提案した. また, [3] のスキームを破ることは, $N = p^2q$ を素因数分解することと同等であることを示した.

Batten and Williams[3] のスキームは, 鍵生成 (メッセージの受信者が実行), 暗号化 (メッセージの送信者が実行), 復号 (メッセージの受信者が実行) の 3 つのアルゴリズムから構成される. 以下に, [3] のアルゴリズムを示す.

鍵生成

1. $3 \pmod 4$ を満たす 2 つの異なる素数 p, q を選ぶ.
2. $N = p^2q$ を計算する.
3. $2 \leq s < \sqrt{q}$ なる正整数 s を選ぶ.
4. $B = \frac{pq}{s}$ を計算する.
5. (N, B) を公開鍵として公開し, (p, q) を秘密鍵として秘匿する.

暗号化

公開鍵 (N, B) を受信し, メッセージ $M \in \mathcal{M}$ に対し, N を用いて暗号文 C を次のように

名検証において検証者が等式を評価することで付録型署名を成立させていると思われる.

計算する.

$$C \equiv M^2 \pmod{N}.$$

ただし, $\mathcal{M}$ は, 以下を満たす正の整数 M からなる集合である.

$$\mathcal{M} = \{M \in \mathbb{Z}^+ \mid M < B, \gcd(M, pq) = 1\}.$$

復号

1. 次式の 4 つの解 X を求める.

$$C \equiv X^2 \pmod{pq}.$$

2. $M = \min(\chi)$ を復号文とする. ここで, χ は, Step 1 で求めた 4 つの解の集合である.

一意復号可能な Rabin-Williams 暗号が正しく機能することを保証する系を導くために, 次の補題と定理を用意する.

補題 2.3.1 ([3, Lemma 1])**.** p, q を素数とし, s を $1 < s < \sqrt{q}$ であるような正の整数とする.

$$\mathcal{M} = \{M \in \mathbb{Z}^+ \mid M < \frac{pq}{s}, \gcd(M, pq) = 1\} \tag{2.5}$$

としたとき, $M \in \mathcal{M}$ ならば, $pq - M \notin \mathcal{M}$ である.

証明. 背理法により示す. $M \in \mathcal{M}$ に対し, $pq - M < \frac{pq}{s}$ と仮定する. これを変形すると $pq - \frac{pq}{s} < M$ を得る. $\mathcal{M}$ の定義から,

$$pq - \frac{pq}{s} < M < \frac{pq}{s}$$

でなければならない. 不等式の両端を引き合わせて両辺に s を乗じ, 整理すると,

$$\begin{aligned} spq - pq &< pq \\ spq &< 2pq \\ s &< 2 \end{aligned}$$

を得る. しかしこれは仮定 $2 \leq s < \sqrt{q}$ に矛盾する. よって補題の主張は示された. □

系 2.3.2 ([3, p. 10])**.** 補題 2.3.1 の条件のもとでは, $C \equiv X^2 \pmod{pq}$ の解のうち $\mathcal{M}$ に含まれるのは高々 2 つである.

定理 2.3.3 ([3, Theorem 1])**.** p, q を $3 \pmod 4$ を満たす 2 つの異なる素数とし, s を $1 \leq s < \sqrt{q}$ かつ $sp > 2q$ を満たす整数とする. $\mathcal{M}$ は補題 2.3.1 の式 (2.5) の定義通りであるとして, $M \in \mathcal{M}$ に対し,

$$C \equiv M^2 \pmod{p^2 q} \tag{2.6}$$

とする. このとき, M は

$$C \equiv X^2 \pmod{pq} \tag{2.7}$$

の唯一の根である.

証明. $pq \mid p^2q$ であるから, M は式 (2.7) の解である.

補題 1.5.4 より, 式 (2.7) は 4 つの解をもつ. それらを M_1, M_2, M_3, M_4 とおく. これらはいずれも $0 < M_i < pq$ の範囲にあり, 互いに異なる. M はこれら M_i のうちのどれか 1 つである. 以下に, M_i $(i = 1, 2, 3, 4)$ のうち, $\frac{pq}{s}$ よりも小さく, 式 (2.7) と 式 (2.6) の両方を満たすものが一意に定まることを背理法により示す.

M_i のうち 2 つが $\frac{pq}{s}$ 未満であると仮定し, $\frac{pq}{s} > M_i > M_j > 0$ とする. $M_i \equiv M_j \pmod{p^2q}$ であることは,

$$p^2q \mid (M_i - M_j)(M_i + M_j)$$

を意味する. 変数 $\alpha \in \{1, -1\}$ を導入し, 3 つの場合に分けて考える.

(i) $p^2q \mid (M_i - M_j)$ または $p^2q \mid (M_i + M_j)$ のとき.
$p^2q \mid (M_i - M_j)$ は, p^2q を法として M_i と M_j が等しいことを意味するため, $M_i > M_j$ に矛盾する.
$p^2q \mid (M_i + M_j)$ は, M_i と M_j が p^2q を法として互いの反数であること, すなわち $M_i = p^2q - M_j$ であることを意味する. M_i, M_j ともに $\frac{pq}{s}$ より小さいので,

$$p^2q = M_i + M_j < \frac{2pq}{s}$$

でなければならない. すなわち, $sp < 2$ でなければならないが, これは仮定 $sp > 2q$ に矛盾する.

(ii) $pq \mid (M_i + \alpha M_j)$ かつ $p \mid (M_i - \alpha M_j)$ のとき.
p が両方の除数に含まれており, $(M_i + \alpha M_j) + (M_i - \alpha M_j) = 2M_i$ であるから,

$$p \mid 2M_i$$

でなければならない. $\mathcal{M}$ の定義式 (2.5) より $\gcd(M_i, pq) = 1$ であり, かつ $p \neq 2$ である. したがって $p \nmid 2M_i$ であるため, この場合は成立しない.

(iii) $p^2 \mid (M_i + \alpha M_j)$ かつ $q \mid (M_i - \alpha M_j)$ のとき.
$M_i + \alpha M_j = kp^2$ を満たす $k \in \mathbb{Z}_{\geq 0}$ が存在するので,

$$0 \leq kp^2 \leq |M_i| + |M_j|$$

が成り立つ. さらに仮定 $sp > 2q$ と合わせて

$$0 \leq kp^2 \leq |M_i| + |M_j| < \frac{2pq}{s} < p^2 \tag{2.8}$$

がいえる. 式 (2.8) から次式を得る.

$$kp^2 < p^2.$$

したがって $k = 0$ であり, $M_i + \alpha M_j = kp^2$ の仮定から

$$M_i = -\alpha M_j$$

となるが, $\alpha \in \{1, -1\}$ は上式を満たさない.

以上より, 主張は証明された. □

定理 2.3.3 の証明から, $C \equiv X^2 \pmod{pq}$ の 4 つの解はすべて pq 未満であり, そのうち M と一致するものだけが $\frac{pq}{s}$ 未満であることがわかる. よって次の系が得られる. この系が, Batten and Williams[3] のスキームが正しく機能することを示していることになる.

系 2.3.4 ([3, p. 12])**.** p, q を $3 \pmod 4$ を満たす素数とし, s を $1 \leq s < \sqrt{q}$ かつ $sp > 2q$ を満たす整数とする. $\mathcal{M}$ は補題 2.3.1 の式 (2.5) の定義通りであるとして, $M \in \mathcal{M}$ に対し,

$$C \equiv M^2 \pmod{p^2 q}$$

とする. このとき, M は

$$C \equiv X^2 \pmod{pq}$$

の最小の根である.

第3章　一意復号可能なRabin-Williams暗号プリミティブの実装

筆者の知る限り, 2.3 節で述べた一意復号可能な Rabin-Williams 暗号 [3] の実装は存在しない. そこで, 本研究では Python によるライブラリを作成した. この章では, 本研究で作成した Python による一意復号可能な Rabin-Williams 暗号プリミティブのライブラリについて述べる. ライブラリを構成する主なファイル構成を表 3.1 に示す.

表 3.1: ライブラリの構成

ファイル名	機能
`main.py`	ライブラリのインターフェースを提供する.
`unique_rabin_williams.py`	鍵生成, 暗号化, 復号を行う.
`utils.py`	ユーティリティ関数を提供する.

ライブラリの依存関係を表 3.2 に示す.

表 3.2: ライブラリの依存関係

パッケージ名	用途
`argparse`	コマンドライン引数の解析を行う.
`logging`	ログ出力を行う.
`random`	乱数生成を行う.
`sympy`	素数判定を行う.
`math`	数学関数を提供する.
`typing`	型ヒントを提供する.

各モジュール, クラス, 関数, メソッドについて説明する.

module main

Unique Rabin-Williams 暗号プリミティブを実行するためのインターフェース. ユーザーはコマンドライン引数を通じて鍵の生成, 暗号化, および復号を行うことができる.

main(*argv: list[str]*) → None

コマンドライン引数を解析し, ユーザー入力に基づいて鍵生成, 暗号化, または復号を実行する. ユーザーが p, q, s, M を指定した場合, 入力を検証し, それに基づいて暗号化と復号を行う. 引数を指定しない場合は自動的に鍵を生成し, 暗号化と復号の例を示す.

引数:

- argv: list[str] = None - コマンドライン引数のリスト. 以下のオプションが利用可能:
  - -p - 素数 p を指定する.
  - -q - 素数 q を指定する.
  - -s - 整数 s を指定する.
  - -M - メッセージ M を指定する.

  引数が指定されていない場合, すべてのパラメータが自動生成される.

使用例:

```
# 鍵とメッセージを指定して暗号化･復号
python main.py --p 1187 --q 2351 --s 4 --M 500000
# 鍵を自動生成してデフォルトメッセージを暗号化･復号
python main.py
```

ソースコード:

```
def main():
    args = parse_arguments()

    if args.p and args.q and args.s and args.M:
        p, q, s, M = args.p, args.q, args.s, args.M
        validate_inputs(p, q, s, M)
        N = p * q * q
    else:
        N, p, q, s = UniqueRabinWilliamsKeyGenerator.generate_keys
            ()
        M = 500000 # Example message
        print(f"Generated N: {N}, p: {p}, q: {q}, s: {s}")

    D = UniqueRabinWilliamsEncryptor.encrypt(M, N)
    print(f"Encrypted message: {D}")

    decrypted_message = UniqueRabinWilliamsDecryptor.decrypt(D, p,
         q)
    print(f"Decrypted message: {decrypted_message}")
```

parse_arguments() → argparse.Namespace

コマンドライン引数を解析し, 解析された引数を Namespace オブジェクトとして返す.
戻り値: argparse.Namespace - コマンドライン引数に基づいて設定された属性を含むオブジェクト.
使用例:

```
args = parse_arguments()
```

ソースコード:

```
def parse_arguments():
    parser = argparse.ArgumentParser(description="Unique Rabin-
        Williams Cryptosystem")
    parser.add_argument("--p", type=int, help="Prime number p")
    parser.add_argument("--q", type=int, help="Prime number q")
    parser.add_argument("--s", type=int, help="Integer s")
    parser.add_argument("--M", type=int, help="Message M")
    return parser.parse_args()
```

validate_inputs(*p: int, q: int, s: int, M: int*) → bool

入力された鍵とメッセージが暗号システムの要件を満たしているかを検証する. 要件を満たしていない場合, エラーメッセージとともに ValueError を発生させる.
引数:

- p: int - 素数 p.
- q: int - 素数 q.
- s: int - 整数 s.
- M: int - メッセージ M.

戻り値: bool - 入力が有効な場合は True, そうでない場合は ValueError が発生.
使用例:

```
validate_inputs(3, 11, 2, 20)
```

ソースコード:

```
def validate_inputs(p: int, q: int, s: int, M: int) -> bool:
    if not isprime(p) or not isprime(q):
        raise ValueError("Both p and q must be prime numbers.")
    if p % 4 != 3 or q % 4 != 3:
        raise ValueError("Both p and q must be congruent to 3 mod
            4.")
    if not 1 < s < q**0.5:
        raise ValueError("s must be greater than 1 and less than
            sqrt(q).")
    N = p * p * q
```

```
    if not 0 < M < N / s:
        raise ValueError("M must be in the range (0, N/s).")
    if not utils.is_coprime(M, N):
        raise ValueError("M and N must be coprime.")
    return True
```

続いて, unique_rabin_williams.py に含まれる 3 つのクラス UniqueRabinWilliamsKeyGenerator, UniqueRabinWilliamsEncryptor および UniqueRabinWilliamsDecryptor とそのメソッドについて説明する.

class UniqueRabinWilliamsKeyGenerator

鍵生成を行うクラス. 鍵をメッセージ送信者が選択せず, 暗号-復号処理に加えライブラリに鍵生成も任せる場合に使用する.

static generate_keys() → Tuple[int, int, int, int]

鍵生成を行う.

戻り値: Tuple[int, int, int, int] - モジュロ N, 秘密値 p, q, パラメータ s

使用例:

```
N, p, q, s = UniqueRabinWilliamsKeyGenerator.validate_inputs()
```

ソースコード:

```
@staticmethod
def generate_keys() -> Tuple[int, int, int, int]:
    p = generate_prime(3)
    q = generate_prime(3)
    s = randint(2, int(q**0.5))
    N = p**2 * q
    return N, p, q, s
```

class UniqueRabinWilliamsEncryptor

暗号化を行うクラス.

static encrypt(*M: int, N: int*) → int

引数:

- M: int - メッセージ M.
- N: int - モジュロ N.

戻り値: int - 暗号化されたメッセージ D.
使用例:

```
D = UniqueRabinWilliamsEncryptor.encrypt(20, 21)
```

ソースコード:

```
@staticmethod
def encrypt(M: int, N: int) -> int:
    return pow(M, 2, N)
```

class UniqueRabinWilliamsDecryptor

復号を行うクラス.

static decrypt(*D: int, p: int, q: int*) → int

引数:

- D: int - 暗号化されたメッセージ D.
- p: int - 素数 p.
- q: int - 素数 q.

戻り値: int - 復号されたメッセージ M.
使用例:

```
M = UniqueRabinWilliamsDecryptor.decrypt(20, 3, 7)
```

ソースコード:

```
@staticmethod
def decrypt(D: int, p: int, q: int) -> int:
    R1 = pow(D, (p + 1) // 4, p)
    R2 = pow(D, (q + 1) // 4, q)
    logger.debug(f"R1: {R1}, R2: {R2}")
    M = [chrem([R1, R2], [p, q]), chrem([R1, -R2], [p, q]),
        chrem([-R1, R2], [p, q]), chrem([-R1, -R2], [p, q])]
    logger.debug(f"M: {M}")
    print()
    return min(M)
```

最後に, 関数群 utils.py に含まれる関数について説明する.

module utils

一般的な数学および暗号関連のユーティリティ関数を提供するモジュール.

generate_prime(*mod4: int*) → int

4 で割った剰余が mod4 と等しい素数を生成する.
引数:

- mod4: int - 生成する素数の 4 で割った剰余.

戻り値: int - 生成された素数.
使用例:

```
p = generate_prime(3)
```

ソースコード:

```
def generate_prime(mod4: int) -> int:
    while True:
        p = random.randint(2**10, 2**11) # or a suitable range
        if p % 4 == mod4 and isprime(p):
            return p
```

chrem(*R: list[int], M: list[int]*) → int

補題 (1.5.4) を用いて連立合同式を解く.
引数:

- R: list[int] - 剰余のリスト.
- M: list[int] - 法のリスト.

戻り値: int - 連立合同式の解 $x \equiv R_i \pmod{M_i}$.
使用例:

```
solution = chrem([2, 3, 2], [3, 5, 7])
```

ソースコード:

```
def chrem(R: list[int], M: list[int]) -> int:
    assert len(R) == len(M)
    n = len(R)
    M_prod = 1
    for i in range(n):
        M_prod *= M[i]
    M_i = [M_prod // M[i] for i in range(n)]
    M_i_inv = [pow(M_i[i], -1, M[i]) for i in range(n)]
    X = 0
    for i in range(n):
        X += R[i] * M_i[i] * M_i_inv[i]
    return X % M_prod
```

is_coprime(*a: int, b: int*) → bool

2 つの数が互いに素であるか否かを判定する.
引数:

- a: int - 第一の数.
- b: int - 第二の数.

戻り値: bool - 2 つの数が互いに素の場合は True, そうでない場合は False.
使用例:

```
coprime = is_coprime(15, 28)
```

ソースコード:

```
def is_coprime(a: int, b: int) -> bool:
    return gcd(a, b) == 1
```

quadratic_residue(*a: int, p: int*) → bool

Fermat の小定理を利用し, 数 a が素数 p を法とした平方剰余であるかどうかを判定する.
引数:

- a: int - 判定する数 a.
- p: int - 素数 p.

戻り値: bool - a が p を法とした平方剰余の場合は True, そうでない場合は False.
使用例:

```
residue = quadratic_residue(5, 11)
```

ソースコード:

```
def quadratic_residue(a: int, p: int) -> bool:
    return pow(a, (p - 1) // 2, p) == 1
```

legendre_symbol(*a: int, p: int*) → int

Legendre 記号を計算する.
引数:

- a: int - 判定する数 a.
- p: int - 素数 p.

戻り値: int - Legendre 記号の値. a が p の平方剰余の場合は 1, 平方非剰余の場合は-1, a と p が互いに素でない場合は 0.
使用例:

```
symbol = legendre_symbol(5, 11)
```

ソースコード:

```
def legendre_symbol(a: int, p: int) -> int:
    if math.gcd(a, p) != 1:
        return 0
    if quadratic_residue(a, p):
        return 1
    else:
        return -1
```

第4章　署名スキームの検討

Batten and Williams[3] は, 彼らの提案スキームを Rabin-Williams 署名を改良した署名スキームであると位置付けている. しかし, 筆者が理解した限りでは, その内容は公開鍵暗号スキームの提案にとどまり, 署名スキームとしての提案にはなっていない. そこで本研究では, [3] の公開鍵暗号スキームのデジタル署名への応用を検討した.

本章では, 本研究で検討したデジタル署名スキームについて議論する.

4.1　付録型 Rabin-Williams 署名

本研究で検討したスキームを構成する鍵生成アルゴリズム (署名者が実行), 署名生成アルゴリズム (署名者が実行), 署名検証アルゴリズム (署名検証者が実行) を以下に示す. これ以降, 検討した署名スキームを**付録型 Rabin-Williams 署名**と呼ぶことにする.

鍵生成

1. $3 \pmod 4$ を満たす 2 つの異なる素数 p, q を選ぶ.
2. $N = p^2 q$ を計算する.
3. 署名検証鍵として N を公開し, 署名鍵として (p, q) を秘匿する.

署名生成

1. 署名鍵 (p, q) を用いて, メッセージ $M \in \mathcal{M}$ に対し

$$X^2 \equiv M \pmod{pq} \tag{4.1}$$

を満たす 4 つの X を, 次式を利用して求める.

$$X \equiv \pm(zqq_p^{-1} \pm wpp_q^{-1}) \pmod{pq}.$$

ただし,

$$\begin{aligned} z &\equiv M^{\frac{p+1}{4}} \pmod p, & w &\equiv M^{\frac{q+1}{4}} \pmod q, \\ q_p^{-1} &\equiv q^{-1} \pmod p, & p_q^{-1} &\equiv p^{-1} \pmod q \end{aligned}$$

であり, メッセージ集合 $\mathcal{M}$ は以下を満たす M からなる集合である.

$$\mathcal{M} = \{M \in \mathbb{Z}^+ \mid \left(\frac{M}{pq}\right) = 1, \gcd(M, pq) = 1\}. \tag{4.2}$$

2. $S = \min(\chi)$ をメッセージ M に対する署名 S とし, メッセージ M とともに受信者に送付する. ここで, χ は Step 1 で求めた 4 つの解の集合である.

署名検証

1. 署名 S, 署名検証鍵 N, メッセージ M を受信し, 次式を計算する.

$$M' \equiv S^2 \pmod{N}.$$

2. Step 1 で求めた M' と受信したメッセージ M を比較し, $M' = M$ ならば署名を受理し, そうでなければ棄却する.

この署名スキームは, [3] の暗号スキームにおける秘密鍵を署名鍵として, 公開鍵を署名検証鍵として用いることを基本的なアイデアとして, [3] の暗号化アルゴリズムを署名検証アルゴリズムに, 復号アルゴリズムを署名アルゴリズムに置き換えたものである.

メッセージ集合が式 (4.2) のように定義される理由を補足しておく. まず, 一つ目の条件

$$\left(\frac{M}{pq}\right) = 1$$

は, 式 (4.1) が解をもつことと同値である. また, 式 (2.5) の (暗号スキームにおける) メッセージ M に関する条件

$$\gcd(M, pq) = 1$$

を付録型 Rabin-Williams 署名のアルゴリズムに適用すると, この条件は

$$\gcd(S, pq) = 1 \tag{4.3}$$

という, 署名 S に対する条件と見なすことができる. 式 (4.3) を満たすためには, S の 2 乗で表される (付録型 Rabin-Williams 署名における) メッセージ M も pq と互いに素である必要がある. この条件が式 (4.2) の 2 つ目の条件

$$\gcd(M, pq) = 1$$

である. したがって, メッセージ集合 $\mathcal{M}$ を式 (4.2) のように定義している.

付録型 Rabin-Williams 署名の例を以下に示す.

例 4.1.1. アリスは, 次のように鍵を生成する. $p = 1187, q = 2351$ を選び, $N = p^2q = 3312486119$ を計算する. 検証鍵 N を公開し, 署名鍵 (p, q) を秘匿する. アリスはボブに対してメッセージ $M = 1024$ に署名したいとする.

次にアリスは, 次のように署名を生成する. $X^2 \equiv M \pmod{pq}$ を満たす X を補題 1.5.4 を用いて求める. まず, 合同式の解を導出するための各値を計算する.

$$z \equiv 1024^{\frac{1187+1}{4}} \equiv 1155 \pmod{1187},$$
$$w \equiv 1024^{\frac{2351+1}{4}} \equiv 32 \pmod{2351}.$$

次に, 法 p での q の逆元 q_p^{-1} と 法 q での p の逆元 p_q^{-1} を求める. これらはそれぞれ整数 k, l を用いて,

$$qq_p^{-1} + kp = 2351q_p^{-1} + 1187k = 1,$$
$$pp_q^{-1} + lq = 1187p_q^{-1} + 2351l = 1$$

に拡張 Eucrid アルゴリズムを適用することで計算できる. すると

$$q_p^{-1} = 258,$$
$$p_q^{-1} = 1840$$

が得られる. これらと z, w を用いて 4 つの X を求める.

$$X_1 \equiv 1155 \times 2351 \times 258 + 32 \times 1187 \times 1840 \equiv 249238 \pmod{2790637},$$
$$X_2 \equiv 1155 \times 2351 \times 258 - 32 \times 1187 \times 1840 \equiv 2790605 \pmod{2790637},$$
$$X_3 \equiv -1155 \times 2351 \times 258 + 32 \times 1187 \times 1840 \equiv 32 \pmod{2790637},$$
$$X_4 \equiv -1155 \times 2351 \times 258 - 32 \times 1187 \times 1840 \equiv 2541399 \pmod{2790637}.$$

求まった解 $X \in \{249238, 2790605, 32, 2541399\}$ のうち, 最小のものは $X = 32$ である. したがって, M に対する署名を $S = 32$ とし, M とともにボブに送付する.

ボブは, まず $M' = S^2 \pmod{N}$ を計算する.

$$M' \equiv 32^2 \equiv 1024 \pmod{3312486119}$$

$M' = M$ であるため, ボブは署名を受理する.

4.2 実験

付録型 Rabin-Williams 署名が機能するか否かを確認するため, Python を用いて実装し, 実験を行った.

$p = 1187, q = 2351$ とした. また, メッセージ集合 $\mathcal{M}$ は次のとおりとした.

$$\mathcal{M} = \{M \in \mathbb{Z}^+ \mid 1 < M < 100000, \left(\frac{M}{pq}\right) = 1, \gcd(M, pq) = 1\}.$$

なお, 実行時間の観点から, $M < 100000$ とした. メッセージ空間に含まれる各メッセージに対して署名を生成し, 署名を検証する. このとき, 署名の検証に成功したメッセージの個数および失敗したしたメッセージの個数を記録した.

実験の実行結果を表 4.1 に示す. $M < 100000$ とした場合, 署名の検証に成功したメッセージは 340 個, 失敗したメッセージは 24663 個であった. また, $M < 1000$ とした場合, 署名の検証に成功したメッセージは 31 個, 失敗したメッセージは 238 個であった.

表 4.1: 署名の検証に成功および失敗したメッセージの個数

メッセージ M の上限	メッセージ空間の大きさ	成功	失敗
100000	25003	340	24663
1000	269	31	238

実験結果のうち M の値が小さいものについて, 各 M に対する署名の値と検証結果を表 4.2 に示す. また, $M < 1000$ で検証に成功したケースのみを表 4.3 に示す.

表 4.2: M に対する署名の値と検証結果

M	署名	検証
1	1	True
3	536112	False
4	2	True
9	3	True
10	469740	False
11	346495	False
12	1072224	False
14	409894	False
16	4	True
23	882999	False
25	5	True
27	342994	False
30	191455	False
33	1212916	False
35	273236	False
36	6	True
38	762019	False
40	939480	False
42	369808	False
43	137491	False
44	41763	False
48	472887	False
49	7	True
56	602567	False
62	1041668	False
64	8	True
69	368890	False
75	110077	False
81	9	True
90	1096537	False
92	592055	False

表 4.3: 検証に成功したメッセージ ($M < 1000$)

M	署名	検証
1	1	True
4	2	True
9	3	True
16	4	True
25	5	True
36	6	True
49	7	True
64	8	True
81	9	True
100	10	True
121	11	True
144	12	True
169	13	True
196	14	True
225	15	True
256	16	True
289	17	True
324	18	True
361	19	True
400	20	True
441	21	True
484	22	True
529	23	True
576	24	True
625	25	True
676	26	True
729	27	True
784	28	True
841	29	True
900	30	True
961	31	True

4.3 考察

実験結果から, 提案手法は署名の検証に成功するメッセージの割合が低く, 署名スキームとしてこのままでは実用には耐えられないことがわかった. ここでは, 提案手法の署名の検証に失敗する原因を考察する.

表 4.2 および表 4.3 を見ると, 署名の検証に成功するメッセージ M は, 以下のように表されるメッセージのみであることがわかる.

$$M = x^2.$$

つまり, 法で見るまでもなく pq 未満の平方数でしか署名検証は成功していない. このことは, メッセージ空間をこのような数のみに限定すれば, 提案手法が失敗することはなくなることを意味する. しかし, このような数のみをメッセージ空間とすると, メッセージ M に対し署名は常に次のように計算することができてしまう.

$$S = M^{\frac{1}{2}}.$$

この計算は署名鍵 p, q を必要としないため, S がメッセージ作成者の真正性を保証することにならない. したがって, このような数のみを平文空間としても, 署名スキームとして機能しない.

pq のような特定の法において平方剰余となる数をメッセージとした場合, なぜ失敗するのかを考察する. まず, 署名生成時に次のような式を計算する.

$$X^2 \equiv M \pmod{pq}. \tag{4.4}$$

一方, 署名検証時には次のような剰余計算を行う.

$$M' \equiv S^2 \pmod{p^2q}.$$

ここで, M' は署名 S に対応するメッセージである. このとき, X を S に変換する式 (4.4) は法 pq でみており, $pq < p^2q$ であるため失われた情報は復元できない. したがって, X が pq より大きい場合, 署名検証に失敗していると考えられる.

このスキームをどのように改良すれば署名検証の失敗を避けられるかを考える.

1. 署名鍵と検証鍵を入れ替える

このスキームでは, Batten and Williams[3] の暗号スキームにおける秘密鍵 pq を署名鍵, 公開鍵 p^2q を署名検証鍵として使用している. しかし, この対応が問題の原因の一つになっている. そこで, 署名鍵と検証鍵を入れ替えることを考える. すなわち, p^2q を署名鍵, pq を検証鍵とする. このアプローチでは署名鍵 $<$ 検証鍵 に起因する署名検証失敗問題 (すなわちメッセージ M の復元失敗) は解決する. しかし [3] の復号アルゴリズムでは, pq に加え p, q を必要とする. p, q が知られてしまうと署名鍵 p^2q が知られてしまうため, このアプローチでは署名鍵の秘匿性が失われてしまう. 例えば, 何らかの方法で [3] の復号アルゴリズムから p, q の必要性を取り除くことができれば, この問題の解決に近づく可能性がある. このようなスキームの修正については今後の課題としたい.

2. 署名検証に pq を使用する

署名検証に用いる法を p^2q ではなく pq を使用することを考える. 修正した署名スキームを以下に示す. なお, 4.2 節に与えた付録型 Rabin-Williams 署名のアルゴリズムと異なるのは, 鍵

生成アルゴリズムの Step 2 と, 署名生成アルゴリズムの Step 2 である.

鍵生成

1. $3 \pmod 4$ を満たす 2 つの異なる素数 p, q を選ぶ.
2. $N = pq$ を計算する.
3. 署名検証鍵として N を公開し, 署名鍵として (p, q) を秘匿する.

署名生成

1. 署名鍵 (p, q) を用いて, メッセージ $M \in \mathcal{M}$ に対し

$$X^2 \equiv M \pmod N \tag{4.5}$$

を満たす 4 つの X を, 次式を利用して求める.

$$X \equiv \pm(zqq_p^{-1} \pm wpp_q^{-1}) \pmod N.$$

ただし,

$$\begin{aligned} z &\equiv a^{\frac{p+1}{4}} \pmod p, \quad w \equiv a^{\frac{q+1}{4}} \pmod q, \\ q_p^{-1} &\equiv q^{-1} \pmod p, \quad p_q^{-1} \equiv p^{-1} \pmod q \end{aligned}$$

であり, メッセージ集合 $\mathcal{M}$ は以下を満たす M からなる集合である.

$$\mathcal{M} = \{M \in \mathbb{Z}^+ \mid \left(\frac{M}{pq}\right) = 1, \gcd(M, pq) = 1\}. \tag{4.6}$$

2. Step 1 で求めた 4 つの解から任意の X をメッセージ M に対する署名 S とし, メッセージ M とともに受信者に送付する.

署名検証

1. 署名 S, 署名検証鍵 N, メッセージ M を受信し, 次式を計算する.

$$M' \equiv S^2 \pmod N. \tag{4.7}$$

2. Step 1 で求めた M' と受信したメッセージ M を比較し, $M' = M$ ならば署名を受理し, そうでなければ棄却する.

式 (4.5) のように求めた S について式 (4.7) を計算するため, $M' = M$ が成立することは明らかである (4.2 節でのプログラムを変更して再度同様の実験も行い, 4.2 節で扱った全てのメッ

セージに対して署名検証に成功したことを確認した). この改良を行うことで署名検証の失敗は避けられるが, この方法は本質的に Rabin[7] が提案したスキームと同じであり, Rabin 署名スキームに帰着することになる.

以上の考察から, [3] はデジタル署名に応用するという観点では, 単純な方法で署名スキームに変換することは難しいことが明らかとなった. 署名の検証鍵 N を p^2q ではなく pq とすることで署名スキームとして成立させることは可能であるが, この場合は Rabin 署名スキームに帰着することになるため, 新たな署名スキームとは言えない. Batten and Williams[3] は署名スキームを提案したと主張しているが, 本研究ではその主張を否定的に確認したことになる.

おわりに

本論文は 4 章で構成される. まず, 第 1 章では, 本研究で提案するデジタル署名スキームで用いる数学理論および暗号理論について解説した. 第 2 章では, Rabin[7] が提案した Rabin 暗号および Williams[8] が提案した Rabin-Williams 暗号に関する主要な関連研究について解説した. 第 3 章では, Batten and Williams[3] が提案した Rabin-Williams 暗号のバリエーションを実装した Python ライブラリに関する詳細な仕様を与えた. 最後に第 4 章で, Batten and Williams[3] の暗号スキームのデジタル署名スキームへの応用について議論した.

本研究では, Batten and Williams[3] が提案した Rabin-Williams 暗号のバリエーションの実装を Python ライブラリとして提供した. また, そのライブラリを用いて, Rabin-Williams 暗号のデジタル署名スキームへの応用を検討し, 特に署名鍵と検証鍵の関係における課題を明らかにした. これらの課題を解決するためには, 暗号スキームと署名スキームの非対称性を克服するアルゴリズムを考案しなければならない. 一方で, Python ライブラリの開発は, 今後これらの課題を解決する際の強力なツールとなるはずである.

謝辞

本論文は筆者である森が岐阜大学大学院自然科学技術研究科知能理工学専攻知能情報学領域に在籍中の研究成果をまとめたものです. 本研究は多くの方々のご指導, ご協力のもと行われており, その方々の助力なくして, 本研究は成立しませんでした. ここに深く感謝申し上げます.

筆者の指導教員である三嶋美和子教授には, 研究の提案からその詳細, 執筆活動にわたるまで, 多くの助言と細やかなご指導をいただきました. また, 研究, 開発に必要な設備を提供していただき, よりよい方向へと進むよう多くの助言をいただきました. ここに心から感謝の意を表します.

同研究室に配属された中島健太氏, 階戸弾氏の 2 人には本研究に関して様々な助言をいただきました. 苦楽を共にした同窓生の 2 人に, 感謝いたします.

最後になりましたが, 本研究に関して多くの示唆をもたらしてくれた三嶋研究室の後輩, 大変優秀なアシスタントの ChatGPT と GitHub Cpilot, そして激励をくれたたくさんの暇な友人達に感謝いたします.

令和 6 年 2 月 5 日
岐阜大学大学院自然科学技術研究科知能理工学専攻
森 晴樹

参考文献

[1] ISO/IEC 9796-3:2006 - Information technology — Security techniques —Digital signature schemes giving message recovery — Part 3: Discrete logarithm based mechanisms. https://www.iso.org/standard/42228.html, September 2006. Accessed: 2024-01-15.

[2] ISO/IEC 9796-2:2010 - Information technology — Security techniques —Digital signature schemes giving message recovery — Part 2: Integer factorization based mechanisms. https://www.iso.org/standard/54788.html, December 2010. Accessed: 2024-01-15.

[3] L. M. Batten and H. C. Williams. Unique rabin-williams signature scheme decryption. Cryptology ePrint Archive, Paper 2019/915, 2019.

[4] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures - HOw to Sign with RSA and Rabin. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, Vol. 1070 of *Lecture Notes in Computer Science*, pp. 399–416. Springer, 1996.

[5] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch. PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017, November 2016.

[6] National Institute of Standards and Technology. FIPS 186-5 Digital Signature Standard (DSS), February 2023. Supersedes FIPS 186-4.

[7] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, USA, 1979.

[8] H. C. Williams. A modification of the RSA public-key encryption procedure (Corresp.). *IEEE Transactions on Information Theory*, Vol. 26, No. 6, pp. 726–729, 1980.

[9] 神保 雅一編著. 暗号とセキュリティ (新インターユニバーシティ). オーム社, 2010.