出金保留中

代替文書

最終ステータス

シリーズ/番号連邦情報処理標準 (FIPS) 186-5

タイトルデジタル署名規格 (DSS)

発行日2023 年 2 月 3 日

DOI https://doi.org/10.6028/NIST.FIPS.186-5

CSRC URL https://csrc.nist.gov/publications/detail/fips/186/5/final

追加情報FIPS 186-4 は、それに代わる改訂版 FIPS 186-5 の公開から 1 年後の 2024 年 2 月 3 日に正式に廃止されます。

NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY U.S. DEPARTMENT OF COMMERCE

FIPS パブ 186-4

連邦情報処理基準
出版物

デジタル署名標準 (DSS)

カテゴリ:コンピュータセキュリティ　　サブカテゴリ:暗号化

情報基盤研究室
米国国立標準技術研究所
ゲイサーズバーグ、メリーランド州 20899-8900
http://dx.doi.org/10.6028/NIST.FIPS.186-4
2013 年 7 月発行

米国商務省
キャメロン•F•ケリー、長官代理
米国国立標準技術研究所
パトリック•D•ギャラガー氏、標準技術担当商務次官兼ディレクター

序文

米国国立標準技術研究所 (NIST) の連邦情報処理標準出版物シリーズは、2002 年の連邦情報セキュリティ管理法 (FISMA) の規定に基づいて採用および公布された標準およびガイドラインに関する公式出版物シリーズです。

FIPS 出版物に関するコメントは歓迎されます。宛先は、100 Bureau Drive, Stop 8900, Gaithersburg, MD 20899-8900、国立標準技術研究所、情報技術研究所のディレクターに宛ててください。

チャールズ•ロマイン、監督
情報基盤研究室

抽象的な

この規格は、デジタル署名の生成に使用できる一連のアルゴリズムを指定します。
デジタル署名は、データに対する不正な変更を検出し、署名者の身元を認証するために使用されます。さらに、署名されたデータの受信者は、その署名が実際に署名者によって作成されたものであることを第三者に証明する証拠としてデジタル署名を使用できます。署名者は後で署名を簡単に否認できないため、これは否認防止として知られています。

キーワード:コンピュータ セキュリティ、暗号化、デジタル署名、連邦情報処理標準、公開キー暗号化。

連邦情報処理標準出版物 186-4

2013年7月

を発表

デジタル署名標準 (DSS)

連邦情報処理標準出版物 (FIPS PUBS) は、1996 年の情報技術管理改革法 (公法 104-106) の第 5131 条に基づく商務長官の承認を経て、国立標準技術研究所 (NIST) によって発行されます。および 1987 年のコンピュータ セキュリティ法 (公法 100-235)。

1. 規格の名前:デジタル署名規格 (DSS) (FIPS 186-4)。

2. 標準のカテゴリ:コンピュータ セキュリティ。サブカテゴリ。暗号化。

3. 説明:この標準は、書面による署名ではなくデジタル署名を必要とするアプリケーションのアルゴリズムを指定します。デジタル署名は、コンピュータではビット列として表現されます。デジタル署名は、署名者の身元とデータの整合性を検証できる一連のルールとパラメーターを使用して計算されます。デジタル署名は、保存されたデータと送信されたデータの両方に対して生成される場合があります。

署名の生成では、秘密キーを使用してデジタル署名を生成します。署名検証では、秘密キーに対応するが同じではない公開キーを使用します。各署名者は秘密鍵と公開鍵のペアを所有しています。公開鍵は公衆に知られる可能性があります。秘密鍵は秘密に保たれます。署名者の公開鍵を使用すれば、誰でも署名を検証できます。秘密鍵を所有するユーザーのみが署名生成を実行できます。

署名生成プロセスではハッシュ関数を使用して、署名対象のデータの圧縮バージョンを取得します。データの圧縮バージョンは、メッセージ ダイジェストと呼ばれることがよくあります。メッセージ ダイジェストはデジタル署名アルゴリズムに入力され、デジタル署名が生成されます。使用されるハッシュ関数は、Secure Hash Standard (SHS)、FIPS 180 で指定されています。FIPS承認のデジタル署名アルゴリズムは、 SHS で指定されている適切なハッシュ関数とともに使用されます。

デジタル署名は、署名されたデータとともに対象の検証者に提供されます。検証エンティティは、主張された署名者の公開鍵と、署名の生成に使用されたのと同じハッシュ関数を使用して署名を検証します。同様の手順を使用して、保存されたデータと送信されたデータの両方の署名を生成および検証できます。

4. 承認当局:商務長官。

5. 保守機関:商務省、国立標準技術研究所、情報技術研究所、コンピュータ セキュリティ部門。

6. 適用性:この基準は、米国法典第 10 編第 2315 条、または米国法典第 44 編第 3502 条 (2) の対象とならない機密扱いでない機密情報の保護を目的として、すべての連邦省庁に適用されます。この規格は、連邦省庁が運用する、または契約に基づいて運用される公開鍵ベースの署名システムの設計と実装に使用されます。この標準の採用と使用は、民間および営利組織が利用できます。

7. アプリケーション:デジタル署名アルゴリズムを使用すると、エンティティは署名されたデータの完全性と署名者の身元を認証できます。署名付きメッセージの受信者は、署名が実際に署名者によって作成されたものであることを第三者に証明する証拠としてデジタル署名を使用できます。署名者は後で署名を簡単に否認できないため、これは否認防止として知られています。デジタル署名アルゴリズムは、電子メール、電子資金移動、電子データ交換、ソフトウェア配布、データ ストレージ、およびデータ完全性の保証とデータ発信元の認証を必要とするその他のアプリケーションでの使用を目的としています。

8. 実装:デジタル署名アルゴリズムは、ソフトウェア、ファームウェア、ハードウェア、またはそれらの任意の組み合わせで実装できます。 NIST は、実装がこの規格のアルゴリズムに準拠しているかどうかをテストするための検証プログラムを開発しました。検証プログラムに関する情報は、 http://csrc.nist.gov/cryptval で入手できます。各デジタル署名アルゴリズムの例は、http://csrc.nist.gov/groups/ST/toolkit/examples.html で入手できます。

政府機関は、デジタル署名キーのペアを他の目的に使用してはならないことを通知されます。

9. その他の承認されたセキュリティ機能:この規格に準拠するデジタル署名の実装では、連邦政府の機密情報を保護するために承認された暗号アルゴリズム、暗号鍵生成アルゴリズム、および鍵確立技術を採用するものとします。承認された暗号化アルゴリズムと技術には、次のいずれかが含まれます。

a .連邦情報処理標準 (FIPS) で指定されている、

b. FIPS または NIST 勧告で採用されている、または

c. FIPS 140 の承認されたセキュリティ機能のリストに指定されています。

10. 輸出規制:特定の暗号化デバイスおよびそれらに関する技術データは、連邦輸出規制の対象となります。この標準を実装する暗号モジュールおよびそれらに関する技術データの輸出は、これらの連邦規制に準拠し、米国商務省産業安全保障局の許可を受けなければなりません。輸出規制に関する情報は、http://www.bis.doc.gov で入手できます。

11. 特許:この規格のアルゴリズムは、米国または外国の特許によって保護されている場合があります。

12. 実施スケジュール:この基準は商務長官の承認後直ちに発効します。アルゴリズムと暗号モジュールを検証するための移行戦略は、NIST のWeb ページ (http://csrc.nist.gov/groups/STM/cmvp/index.html) の「通知」に掲載されます。この移行計画は、連邦政府機関による、この標準の以前のバージョンへの準拠についてテストおよび検証されたモジュールから、暗号モジュール検証プログラムに基づく FIPS 186-4 への準拠についてテストおよび検証されたモジュールへの移行に対処します。この移行計画により、連邦政府機関やベンダーは FIPS 186-4 にスムーズに移行できるようになります。

13. 仕様:連邦情報処理標準 (FIPS) 186-4 デジタル署名標準 (添付)。

14. クロスインデックス:この規格では次の文書が参照されています。文書番号で特定のバージョンまたは日付が示されていない限り、指定された文書の最新バージョンが参照として使用されます。

ａ． FIPS PUB 140、暗号化モジュールのセキュリティ要件。

b. FIPS PUB 180 セキュア ハッシュ標準。

c. ANS X9.31-1998、金融サービス業界向けの可逆公開キー暗号化を使用したデジタル署名 (rDSA)。

d. ANS X9.62-2005、金融サービス業界向けの公開キー暗号化: 楕円曲線デジタル署名アルゴリズム (ECDSA)。

e. ANS X9.80、素数生成、素数性テスト、および素数証明書。

f.公開キー暗号化標準 (PKCS) #1、RSA 暗号化標準。

g. Special Publication (SP) 800-57、鍵管理に関する推奨事項。

h. Special Publication (SP) 800-89、デジタル署名アプリケーションの保証を取得するための推奨事項。

私。 Special Publication (SP) 800-90A、乱数生成に関する推奨事項
決定論的ランダム ビット ジェネレーターの使用。

j. Special Publication (SP) 800-102、デジタル署名の適時性に関する推奨事項。

k.特別出版物 (SP) 800-131A、移行: 移行に関する推奨事項
暗号化アルゴリズムとキーの長さの使用。

l. IEEE規格1363-2000、公開キー暗号化の標準仕様。

15. 資格:デジタル署名システムのセキュリティは、署名者の秘密鍵の機密性の維持に依存します。したがって、署名者は秘密鍵の開示を防ぐ必要があります。この標準の目的は、デジタル署名を生成するための一般的なセキュリティ要件を指定することですが、この標準への準拠は、次のことを保証するものではありません。

特定の実装は安全です。デジタル署名機能を実装するモジュールが安全な環境で設計および構築されていること を確認するのは実装者の責任です。
やり方。

同様に、この規格に準拠した実装を含む製品の使用は、その製品が使用されるシステム全体のセキュリティを保証するものではありません。各機関または部門の責任当局は、実装全体が許容可能なレベルのセキュリティを提供することを保証するものとします。

この種の規格は科学技術の進歩や革新に適応できる柔軟性を備えていなければならないため、この規格はその適切性を評価するために 5 年ごとに見直されます。

16. 免除手順:連邦情報セキュリティ管理法 (FISMA) では、商務長官によって義務付けられた連邦情報処理標準 (FIPS) に対する免除は認められていません。

17. 規格のコピーの入手先:この出版物は、 http://csrc.nist.gov/publications/ にアクセスすることで入手できます。他のコンピュータ セキュリティに関する出版物も同じ Web サイトで入手できます。

目次

連邦情報処理標準出版物 186-4

2013年7月

の仕様

デジタル署名標準 (DSS)

1. はじめに

この規格は、バイナリ データ (一般にメッセージと呼ばれる) の保護と、それらのデジタル署名の検証および検証に使用できるデジタル署名の生成方法を定義します。 3つの技術が承認されています。

(1) デジタル署名アルゴリズム (DSA) はこの規格で規定されています。仕様
ドメインパラメータの生成、公開鍵と秘密鍵のペアの生成、デジタル署名の生成と検証の基準が含まれます。

(2) RSA デジタル署名アルゴリズムは、American National Standard (ANS) で規定されています。
X9.31 および公開キー暗号化標準 (PKCS) #1。 FIPS 186-4 は、これらの標準のいずれかまたは両方の実装の使用を承認し、追加の要件を指定します。

(3) 楕円曲線デジタル署名アルゴリズム (ECDSA) は、ANS X9.62 で規定されています。
FIPS 186-4 は ECDSA の使用を承認し、追加の要件を指定します。
連邦政府が使用するために推奨される楕円曲線がここに提供されます。

この規格には、有効なデジタル署名に必要な保証を取得するための要件が含まれています。これらの保証を取得する方法は、NIST Special Publication (SP) 800-89、デジタル署名アプリケーションの保証を取得するための推奨事項で提供されています。

2. 用語、頭字語、数学記号の用語集

2.1 用語と定義

承認された	FIPS 承認および/または NIST 推奨。 1) FIPS または NIST 勧告で指定されている、または 2) FIPS または NIST 勧告で採用されている、または 3) NIST 承認のセキュリティ機能のリストで指定されているアルゴリズムまたは技術。
ドメインパラメータの有効性の保証	ドメインパラメータが算術的に正しいという確信。
所持の保証	エンティティが秘密鍵および関連する鍵マテリアルを所有しているという確信。
公開鍵の有効性の保証	公開キーが算術的に正しいという確信。
ビット列	0 と 1 の順序付けされたシーケンス。左端のビットは文字列の最上位ビットです。右端のビットは文字列の最下位ビットです。
証明書	キー ペアとそのキー ペアの使用を許可された所有者を一意に識別するデータのセット。証明書には所有者の公開鍵と場合によってはその他の情報が含まれており、認証局 (つまり、信頼できる当事者) によってデジタル署名され、それによって公開鍵が所有者にバインドされます。
認証局 (カリフォルニア州)	証明書を発行し、PKI ポリシーへの準拠を徹底する責任を負う公開キー基盤 (PKI) 内のエンティティ。
主張された署名者	検証者の観点から見ると、主張された署名者は、デジタル署名を生成したとされるエンティティです。
デジタル署名	データの暗号化変換の結果。適切に実装されると、発信元の認証、データの整合性、および署名者の否認防止を検証するためのメカニズムが提供されます。
ドメイン パラメーター シード	ドメイン パラメーターの生成または検証プロセスの入力として使用されるビットの文字列。
ドメインパラメータ	暗号化アルゴリズムで使用されるパラメータ。通常、ユーザーのドメインに共通です。 DSA または ECDSA 暗号化キーのペアは、特定のドメイン パラメーターのセットに関連付けられます。

実在物	個人（個人）、組織、デバイス、またはプロセス。 「パーティー」と同じ意味で使用されます。
同等のプロセス	同じ値が各プロセスに (入力パラメーターとして、またはプロセス中に使用できる値として、またはその両方として) 入力されたときに、同じ出力が生成される場合、2 つのプロセスは同等です。
ハッシュ関数	任意の長さのビット列を固定長のビット列にマッピングする関数。承認されたハッシュ関数は FIPS 180 で指定されており、次の特性を満たすように設計されています。 1. (一方向) 以下の入力を見つけることは計算上不可能です。事前に指定された新しい出力にマッピングされ、 2. (衝突耐性) 同じ出力にマッピングされる 2 つの異なる入力を見つけることは計算上不可能です。
ハッシュ値	「メッセージダイジェスト」を参照してください。
署名対象者	将来的にデジタル署名を生成する予定のエンティティ。
鍵	暗号化アルゴリズムと組み合わせて使用され、その動作を決定するパラメータ。この規格に適用される例は次のとおりです。 1. データからのデジタル署名の計算、および 2. デジタル署名の検証。
キーペア	公開キーとそれに対応する秘密キー。
メッセージ	署名されたデータ。署名の検証および検証プロセスでは「署名済みデータ」とも呼ばれます。
メッセージダイジェスト	メッセージにハッシュ関数を適用した結果。 「ハッシュ値」とも呼ばれます。
否認防止	秘密鍵を所有する特定のエンティティ (つまり、署名者）。
所有者	キー ペアの所有者は、キー ペアの秘密キーの使用を許可されたエンティティです。
パーティー	個人（個人）、組織、デバイス、またはプロセス。 「エンティティ」と同じ意味で使用されます。
メッセージごとの秘密番号	各デジタル署名の生成前に生成される秘密の乱数。

公開鍵インフラストラクチャ (PKI)	公開鍵証明書を発行、維持、取り消すために確立されたフレームワーク。
素数生成シード	必要な特性を持つ素数を決定するために使用されるランダムなビットの文字列。
秘密鍵	非対称 (公開キー) 暗号アルゴリズムで使用される暗号キー。デジタル署名の場合、秘密キーは所有者に一意に関連付けられ、公開されません。秘密キーは、対応する公開キーを使用して検証できるデジタル署名を計算するために使用されます。
推定素数	確率的素数性テストに基づいて、素数であると考えられる整数。いわゆる確率素数が実際に合成される確率は無視できる程度であるべきです。
証明可能な素数	素数になるように構築された整数、または素数証明アルゴリズムを使用して素数になるように計算された整数。
擬似ランダム	プロセスまたはプロセスによって生成されるデータは、結果が決定的である場合には擬似ランダムであると言われますが、プロセスの内部動作が観察から隠されている限り実質的にランダムでもあります。暗号化の目的では、「効果的に」とは「意図されたセキュリティ強度の範囲内で」を意味します。
公開鍵	非対称 (公開キー) 暗号化アルゴリズムで使用され、秘密キーに関連付けられる暗号キー。公開キーは所有者に関連付けられており、公開される場合があります。デジタル署名の場合、公開キーは、対応する秘密キーを使用して署名されたデジタル署名を検証するために使用されます。
セキュリティの強さ	暗号化アルゴリズムまたはシステムを解読するために必要な作業量 (つまり、操作の数) に関連付けられた数値。セキュリティ レベルと呼ばれることもあります。
するものとします	この規格の要件を示すために使用されます。
すべき	強い推奨を示すために使用されますが、この規格の要件ではありません。
署名者	秘密キーを使用してデータのデジタル署名を生成するエンティティ。
署名の生成	デジタル署名アルゴリズムと秘密キーを使用してデータのデジタル署名を生成するプロセス。

署名の検証	デジタル署名の（数学的)検証と適切な保証の取得（公開鍵の有効性､秘密鍵の所有など）。
署名検証	デジタル署名アルゴリズムと公開鍵を使用して、データのデジタル署名を検証します。
署名されたデータ	デジタル署名が計算されたデータまたはメッセージ。 「メッセージ」もご覧ください。
購読者	認証局に証明書を申請して受け取ったエンティティ。
信頼できるサードパーティ (TTP)	所有者または検証者､あるいはその両方によって信頼されている､所有者および検証者以外のエンティティ。 「信頼できる当事者」と短縮されることもあります。
検証者	公開鍵を使用してデジタル署名の信頼性を検証するエンティティ。

2.2 頭字語

ANS	アメリカ国家規格。
カリフォルニア州	認証局。
DSA	デジタル署名アルゴリズム｡この規格に規定されている。
ECDSA	楕円曲線デジタル署名アルゴリズム。 ANS X9.62で規定されています。
FIPS	連邦情報処理標準。
NIST	米国国立標準技術研究所。
PKCS	公開キー暗号化標準。
PKI	公開鍵インフラストラクチャ。
RBG	ランダムビットジェネレーター。
RSA	Rivest､Shamir､Adleman によって開発されたアルゴリズム。 ANS X9.31 および PKCS #1 で指定されています。
社	安全なハッシュ アルゴリズム。 FIPS 180 で指定されています。
SP	NIST 特別出版物
TTP	信頼できるサードパーティ。

2.3 数学記号

モジュールn	整数 a を次で割ったときの一意の余り r、0 ≤ r ≤ (n – 1) 正の整数 n。たとえば、23 mod 7 = 2 となります。
b ≡ a mod n	b – a = kn となるような整数 k が存在します。同等に、mod n = b mod n。
カウンター	ドメイン パラメーター シードを使用して DSA ドメイン パラメーターを生成する場合に、 ドメイン パラメーター生成プロセスから得られるカウンター値。
d	1. RSA の場合、秘密キーの秘密署名指数。 2. ECDSA の場合、秘密キー。
domain_parameter_seed	ドメイン パラメータの生成に使用されるシード。
e	RSA 公開キーの公開検証指数。
g	DSA ドメイン パラメータの 1 つ。 g は GF(p)* の q 次巡回群の生成子です。つまり、 GF(p) の乗法群内の次数 q の要素です。
GCD (a、b)	整数 a と b の最大公約数。
ハッシュ(M)	承認されたハッシュ関数を使用したメッセージ M のハッシュ計算 (メッセージ ダイジ ェストまたはハッシュ値) の結果。
索引	g の生成で使用される値で、その意図された用途 (デジタル署名など) を示します。
k	DSA および ECDSA の場合、メッセージごとの秘密番号。
L	DSA の場合、パラメーター p の長さ (ビット単位)。
(L、N)	DSA キー ペアに関連付けられた長さパラメータのペア。L は p の長さ、N は q の長さです。
LCM (a、b)	整数 a と b の最小公倍数。
レン (a)	a の長さ (ビット単位)。整数 L、ただし 2L-1 ≤ a < 2L 。
M	デジタル署名アルゴリズムを使用して署名されたメッセージ。
メートル	ECDSA の場合、有限体 GF の次数 。
N	DSA の場合、パラメーター q の長さ (ビット単位)。

n	1. RSA の場合、係数。 n のビット長は次のようにみなされます。 鍵のサイズ。 2. ECDSA の場合、楕円曲線の基点の次数。 n のビット長が鍵のサイズとみなされます。
(n、d)	RSA 秘密キー。n は法、d は秘密署名指数です。
(n、e)	RSA 公開キー。n は法、e は公開検証指数です。
ンレン	RSA モジュラス n の長さ (ビット単位)。
p	1. DSA の場合、DSA ドメイン パラメータの 1 つ。素数 これはガロア体 GF(p) を定義し、GF(p) の演算で法として使用されます。 2. RSA の場合、法 n の素因数。
q	1. DSA の場合、DSA ドメイン パラメータの 1 つ。の主因数 p-1。 2. RSA の場合、法 n の素因数。
Q	ECDSA 公開キー。
r	DSA または ECDSA デジタル署名の 1 つのコンポーネント。 (r, s) の定義を参照してください。
(r、s)	DSA または ECDSA デジタル署名。r と s はデジタル署名コンポーネントです。
s	DSA または ECDSA デジタル署名の 1 つのコンポーネント。 (r, s) の定義を参照してください。
シーレン	ビット単位のdomain_parameter_seedの長さ。
SHA×(M)	M が SHA-x ハッシュ関数への入力である場合の結果。SHA-x は FIPS 180 で指定されています。
パッ	DSA 秘密キー。
y	DSA 公開キー。
⊕	同じ長さのビット文字列に対するビットごとの論理「排他的論理和」。各ビット列の対応するビットについて、結果は次のように決定されます: 0 ⊕ 0 = 0、0 ⊕ 1 = 1、1 ⊕ 0 = 1、または 1 ⊕ 1 = 0。 例: 01101 ⊕ 11010 = 10111
+	追加。

	乗算。
/	分割。
|| b	2 つの文字列 a と b の連結。 a と b は両方ともビット文字列であるか、両方ともバイト文字列です。
a	a の上限: a 以上の最小の整数。たとえば、 5 = 5、 5.3 = 6、 –2.1 = –2 となります。
a	の床。 a 以下の最大の整数。たとえば、 5 = 5、 5.3 = 5、および -2.1 です。 = -3。
|a|	a の絶対値。 |a| a < 0 の場合は – a;それ以外の場合は、単に a です。 たとえば、|2| = 2、および |–2| = 2。
[a、b]	a と b の間の整数の間隔 (a と b を含む)。たとえば、[1, 4] は整数 1、2、3、4 で構成されます。
{、a、b、…}	オプションの情報を示すために使用されます。
0x	16 進数の文字で表されるビット文字列の接頭辞。

3. 一般的な議論

デジタル署名は、書面による署名の電子的な類似物です。デジタル署名を使用すると、署名者が情報に署名したことを保証できます。さらに、デジタル署名を使用して、署名後に情報が変更されたかどうかを検出する (つまり、署名されたデータの完全性を検出する) こともできます。これらの保証は、データが送信で受信されたか、ストレージから取得されたかに関係なく取得できます。この規格の要件を満たす適切に実装されたデジタル署名アルゴリズムは、これらのサービスを提供できます。

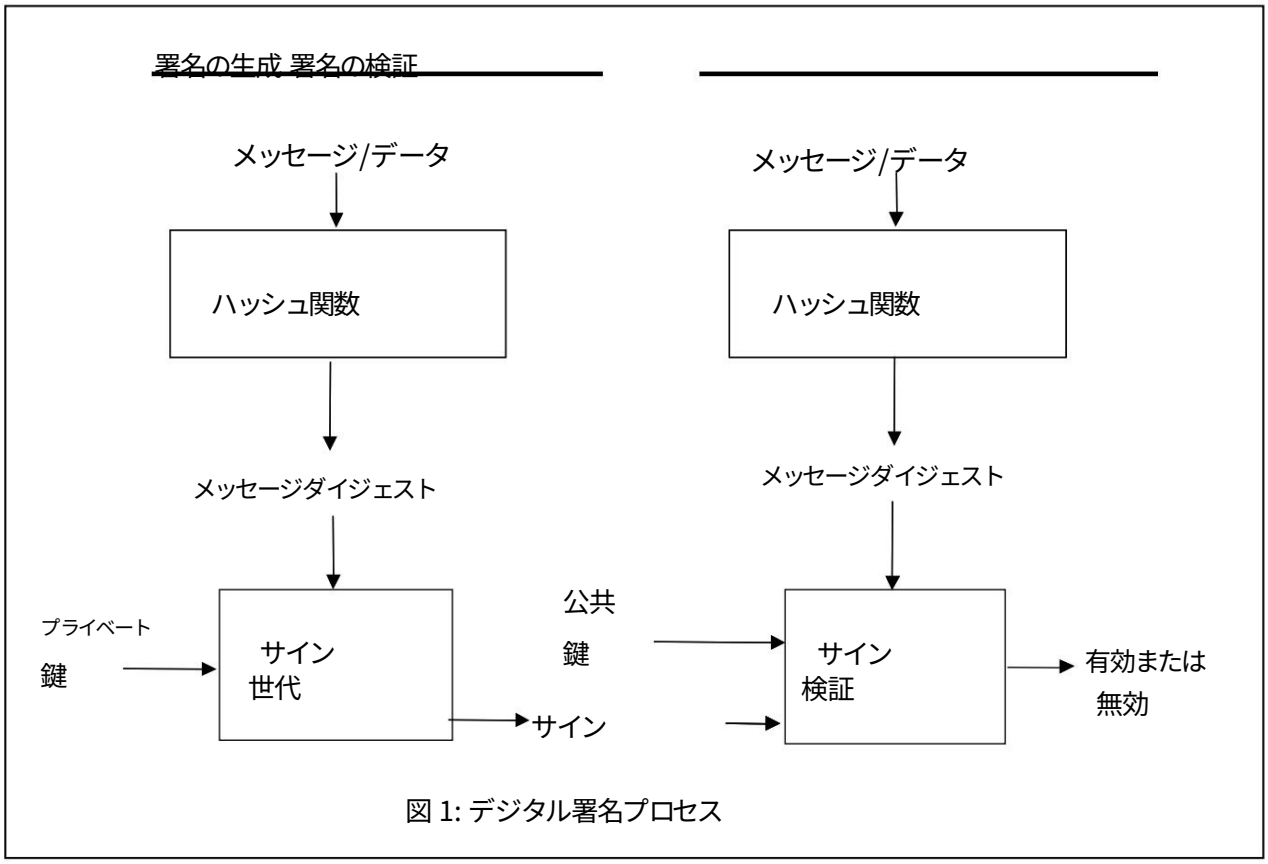

図 1: デジタル署名プロセス

デジタル署名アルゴリズムには、署名生成プロセスと署名検証プロセスが含まれます。署名者は生成プロセスを使用してデータのデジタル署名を生成します。検証者は検証プロセスを使用して署名の信頼性を検証します。各署名者は公開鍵と秘密鍵を持っており、その鍵ペアの所有者です。図 1 に示すように、秘密キーは署名生成プロセスで使用されます。キー ペアの所有者は、デジタル署名を生成するために秘密キーを使用する権限を与えられた唯一のエンティティです。他のエンティティがキー ペアの所有者であると主張し、秘密キーを使用して不正な署名を生成するのを防ぐために、

秘密鍵は秘密にしておく必要があります。承認されたデジタル署名アルゴリズムは、署名者の秘密鍵を知らない攻撃者が別のメッセージに対して署名者と同じ署名を生成することを防ぐように設計されています。つまり、署名は偽造できないように設計されています。この標準では、署名者または鍵ペアの所有者を指すために、多くの代替用語が使用されています。将来デジタル署名を生成する予定のエンティティは、意図された署名者と呼ばれる場合があります。署名されたメッセージの検証に先立って、署名者の実際の身元について十分な保証が得られるまで、署名者は主張された署名者と呼ばれます。

公開キーは署名検証プロセスで使用されます (図 1 を参照)。公開キーを秘密にしておく必要はありませんが、その完全性は維持する必要があります。公開鍵を使用すれば、誰でも正しく署名されたメッセージを検証できます。

署名生成プロセスと検証プロセスの両方で、メッセージ (つまり、署名されたデータ) は、承認されたハッシュ関数を使用してメッセージの固定長表現に変換されます。元のメッセージとデジタル署名の両方が検証者に利用可能になります。

検証者は、署名の検証に使用される公開鍵が、デジタル署名を生成したと主張するエンティティ (つまり、署名者と主張する) に属するという保証を必要とします。つまり、検証者は、署名者がデジタル署名の生成と検証に使用される公開鍵/秘密鍵ペアの実際の所有者であるという保証を必要とします。この保証を提供するために、所有者の身元と所有者の公開鍵のバインドが行われます。

検証者は、キー ペアの所有者が実際に秘密キーを所有していることの保証も必要とします。公開鍵に関連付けられており、公開鍵が数学的に正しい鍵であることを確認します。

これらの保証を取得することにより、検証者は、公開鍵を使用してデジタル署名が正しく検証できれば、そのデジタル署名が有効である (つまり、鍵ペアの所有者が実際にメッセージに署名した) という保証を得ることができます。デジタル署名の検証には、デジタル署名の (数学的) 検証と適切な保証の取得の両方が含まれます。このような保証が必要な理由は次のとおりです。

1. 署名者が、署名の検証に公開コンポーネントが使用される鍵ペアの実際の所有者であるという保証を検証者が得られない場合、署名の偽造の問題は、身元を偽るという問題に帰着します。たとえば、数学的に一貫した鍵ペアを所有している人は誰でもメッセージに署名し、署名者が米国大統領であると主張できます。検証者が、メッセージの署名を数学的に検証するために使用される公開鍵の所有者が実際に大統領であるという保証を必要としない場合、署名検証が成功すると、メッセージが署名以来変更されていないことが保証されますが、保証はされません。メッセージが大統領からのものであるという保証（つまり、検証者はデータの完全性を保証しますが、送信元の認証が不足しています）。

2. 署名の検証に使用される公開キーが数学的に有効でない場合、署名アルゴリズムの暗号強度を確立するために使用される引数は適用されない可能性があります。

その公開キーで検証できる署名を生成できるのは、所有者だけではない可能性があります。

3. 公開鍵インフラストラクチャが、鍵ペアの所有者が所有者の公開鍵に対応する秘密鍵の知識を証明しているという保証を検証者に提供できない場合、悪意のあるエンティティがその身元 (または別の当事者によって使用されている（または使用されている)公開鍵にバインドされた、想定されたアイデンティティ）。悪意のあるエンティティは、相手が署名した特定のメッセージの発信元を主張する可能性があります。あるいは、悪意のある組織が、特定のメッセージの署名を検証できるようにするという唯一の目的で選択された公開キーの所有権を取得した可能性もあります。

技術的には、デジタル署名アルゴリズムで使用される鍵ペアは、デジタル署名以外の目的 (鍵の確立など) にも使用できます。ただし、この標準で指定されているデジタル署名の生成と検証に使用される鍵ペアは、他の目的に使用してはならない。見る

詳細については、キーの使用法に関する SP 800-57 を参照してください。

この規格に従ってデジタル署名の生成または検証機能を有効にするには、多くの手順が必要です。

デジタル署名を生成するすべての関係者は、セクション 3.1 で説明されているように、初期セットアップ プロセスを実行するものとします。デジタル署名の生成は、セクション 3.2 で説明したように実行されます。デジタル署名の検証
検証はセクション 3.3 で説明されているように実行されます。

3.1 初期設定

図 2 は、次の手順を示しています。
デジタルを生成する前に実行される

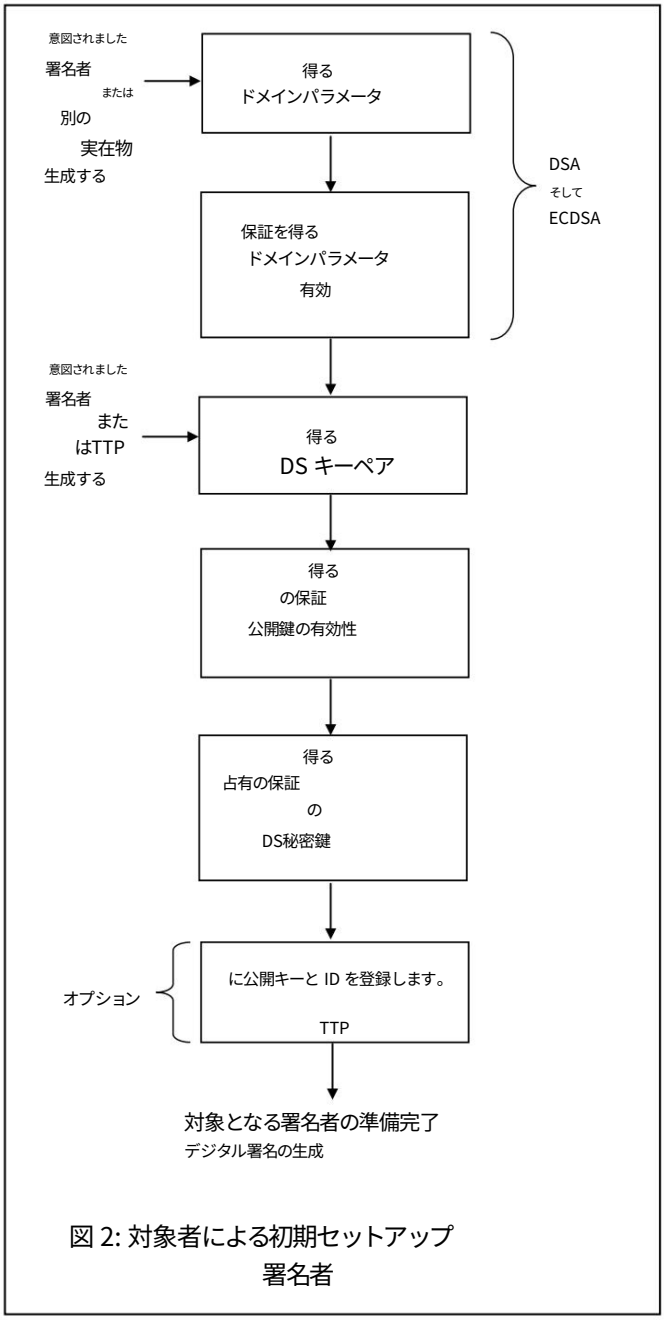

図 2: 対象者による初期セットアップ
署名者

署名者として行動することを意図した団体による署名。

DSA および ECDSA アルゴリズムの場合、意図された署名者は、まずドメイン パラメータ自体を生成するか、別のエンティティが生成したドメイン パラメータを取得することによって、適切なドメイン パラメータを取得する必要があります。ドメインパラメータのセットを取得したら、意図された署名者はそれらのドメインパラメータの有効性の保証を得る必要があります。この保証を取得するための承認された方法は、SP 800-89 で提供されます。 RSA アルゴリズムはドメイン パラメーターを使用しないことに注意してください。

意図された各署名者は、キー ペア自体を生成するか、信頼できる当事者からキー ペアを取得することによって、適切なデジタル署名アルゴリズムの指定に従って生成されたデジタル署名キー ペアを取得する必要があります。意図された署名者は、キー ペアの使用を許可されており、そのキー ペアの所有者です。信頼できる当事者が鍵ペアを生成する場合、たとえ信頼できる当事者が秘密鍵を知っていたとしても、その当事者が所有者になりすますことがないように信頼される必要があることに注意してください。

鍵ペアを取得した後、意図された署名者 (現在は鍵ペアの所有者) は、 (1) 公開鍵の有効性の保証、および (2) 関連する秘密鍵を実際に所有しているという保証を取得するものとします。これらの保証を取得するための承認された方法は、SP 800-89 で提供されます。

デジタル署名検証者は、署名者の身元を保証する必要があります。デジタル署名が生成および検証される環境に応じて、キー ペアの所有者 (つまり、意図された署名者) が公開キーを登録し、相互に信頼できる当事者との身元証明を確立する場合があります。たとえば、認証局 (CA) は、所有者の身元の証明が提供された後、所有者の公開キーと身元を含む資格情報に署名して証明書を作成できます。資格情報を認証し、証明書を配布するシステムは、この規格の範囲外です。身元証明を確立する他の手段（たとえば、身元証明書を公開鍵とともに将来の検証者に直接提供することによる）も許可されます。

3.2 デジタル署名の生成

図 3 は、意図された署名者 (つまり、デジタル署名を生成するエンティティ) によって実行されるステップを示しています。

デジタル署名を生成する前に、承認された適切なハッシュ関数を使用して、署名される情報に対してメッセージ ダイジェストを生成する必要があります。

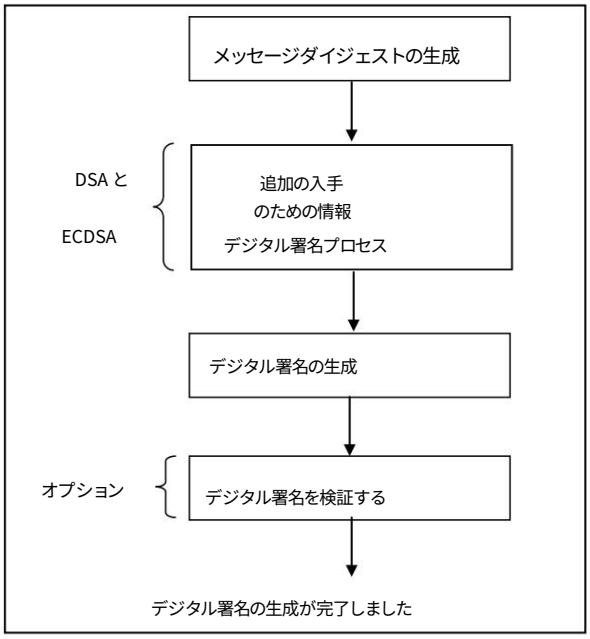

図 3: デジタル署名の生成

使用するデジタル署名アルゴリズムに応じて、追加情報を取得する必要があります。たとえば、 DSA および ECDSA については、メッセージごとのランダムな秘密番号が取得されます。

選択されたデジタル署名アルゴリズム、署名秘密鍵、メッセージ ダイジェスト、およびデジタル署名プロセスに必要なその他の情報を使用して、デジタル署名はこの規格に従って生成されます。

署名者はオプションで、署名検証プロセスを使用してデジタル署名を検証できます。
および関連する公開キー。このオプションの検証は、他の方法では検出されない署名生成の計算エラーを検出するための最終チェックとして機能します。この検証は、価値の高いメッセージに署名する場合、複数のユーザーが署名を検証することが予想される場合、または検証者がかなり後で署名を検証する場合に賢明である可能性があります。

3.3 デジタル署名の検証と検証

図 4は、検証者 （例えば、署名されたデータおよび関連するデジタル署名の対象受信者）によって実行されるデジタル署名の検証および検証プロセスを示している。この図は、成功した検証および検証プロセスを示していることに注意してください (つまり、エラーは検出されません)。

デジタル署名を検証するために、検証者は、主張された身元に基づいて、主張された署名者の公開鍵を取得する必要があります。 DSA または ECDSA がデジタル署名の生成に使用されている場合、検証者はドメイン パラメータも取得する必要があります。公開鍵およびドメインパラメータは、たとえば、信頼できる当事者 （たとえば、CA）によって作成された証明書から、または主張された署名者から直接取得できます。メッセージダイジェストは、デジタル署名生成プロセス中に使用されたのと同じハッシュ関数を使用して、署名が検証されるデータ上で (つまり、受信したデジタル署名上ではなく) 生成されます。適切なデジタル署名アルゴリズム、ドメイン パラメータ (該当する場合)、公開鍵、および新しく計算されたメッセージ ダイジェストを使用して、受信したデジタル署名がこの規格に従って検証されます。検証プロセスが失敗した場合、データが正しいかどうかについて推論することはできません。指定された公開キーと指定された署名形式を使用した場合、そのデータのデジタル署名を検証できないことだけがわかります。

検証されたデジタル署名を有効なものとして受け入れる前に、検証者は、 (1) 署名者の主張する身元の保証、(2) ドメイン パラメータ (DSA および ECDSA の) の有効性の保証、(3) 署名者の有効性の保証を得る必要があります。公開鍵、および (4) 主張された署名者が、署名の生成時にデジタル署名の生成に使用された秘密鍵を実際に所有しているという保証。検証者がこれらの保証を取得するための方法は、SP 800-89 で提供されます。ドメインパラメータの有効性の保証は、初期セットアップ中に取得されている可能性があることに注意してください (セクション 3.1 を参照)。

検証および保証プロセスが成功した場合、デジタル署名および署名されたデータは有効であるとみなされます。ただし、検証または保証プロセスが失敗した場合、デジタル署名は無効であると見なされます。組織のポリシーは、無効なデジタル署名に対して取られる措置を管理するものとします。このようなポリシーは、この規格の範囲外です。

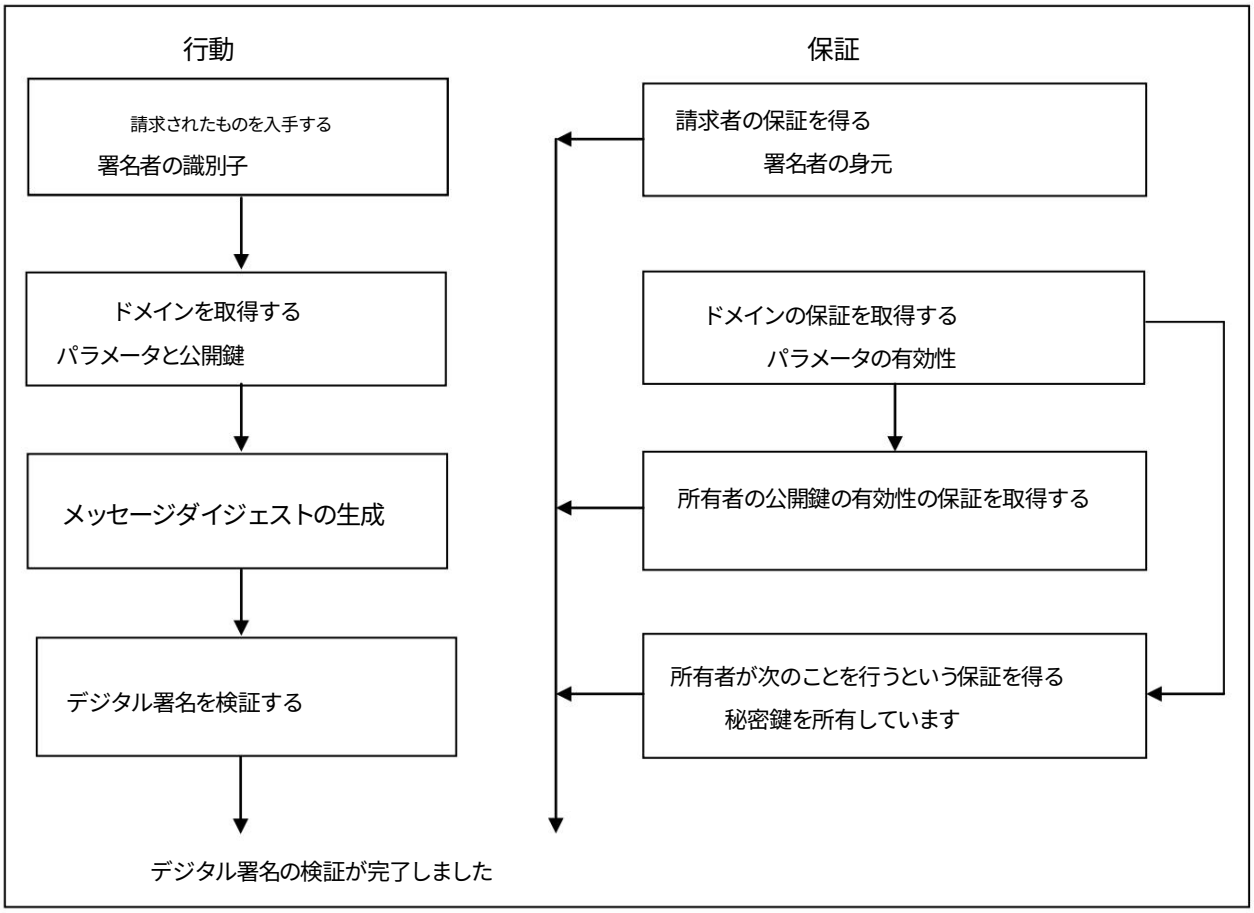

図 4: デジタル署名の検証と検証

4 デジタル署名アルゴリズム (DSA)

4.1 DSA パラメータ

DSA デジタル署名は、ドメイン パラメーターのセット、秘密キーx、メッセージごとの秘密番号k、署名されるデータ、およびハッシュ関数を使用して計算されます。デジタル署名は、同じドメイン パラメーター、デジタル署名の生成に使用された秘密鍵xに数学的に関連付けられた公開鍵y 、検証対象のデータ、および署名の生成中に使用されたのと同じハッシュ関数を使用して検証されます。これらのパラメータは次のように定義されます。

pは素数係数、 $2^{L-1} < p < 2^{L}$、 Lはpのビット長です。 Lの値はセクション 4.2 に記載されています。

q は(p – 1)の素約数です。ここで、 $2^{N-1} < q < 2^{N}$、 Nはqのビット長です。 Nの値
セクション 4.2 で説明されています。

g GF(p) の乗法群内の次数 q の部分群の生成子 (1 < g となる)
< p.

x秘密にしておく必要がある秘密鍵。 xはランダムまたは擬似ランダムに生成される
$0 < x < q$のような整数。つまり、 xは [1, q–1] の範囲内にあります。

y は公開鍵で、 $y = g^{x} \bmod p$ です。

k各メッセージに固有の秘密番号。 kはランダムまたは擬似ランダムに生成された整数で、$0 < k < q$、つまりk
の範囲は [1, q–1] になります。

4.2 DSA のパラメータ サイズとハッシュ関数の選択

この規格では、 LとNのペア(それぞれpとqのビット長) について次の選択肢を指定しています。

L = 1024、 N = 160

L = 2048、 N = 224

L = 2048、 N = 256

L = 3072、 N = 256

連邦政府機関は、これらの選択肢の 1 つ以上を使用してデジタル署名を生成するものとします。

FIPS 180 で指定されている承認済みのハッシュ関数は、デジタル署名の生成中に使用されます。 DSA デジタル署名プロセスに関連するセキュリティ強度は、(L, N) ペアのセキュリティ強度と使用されるハッシュ関数のセキュリティ強度の最小値を超えません。使用するハッシュ関数のセキュリティ強度と(L, N) ペアのセキュリティ強度の両方が、要求されるセキュリティ強度を満たすか、それを超えている必要があります。

デジタル署名プロセス。各(L, N) ペアとハッシュ関数のセキュリティ強度はSP 800-57 で提供されます。

SP 800-57は、デジタル署名の生成のための所定の期間における望ましいセキュリティ強度に応じた適切な(L, N) ペアの選択に関する情報を提供します。
署名された情報がその情報の予想される有効期間全体にわたって保護される(L, N) ペアが選択されます。たとえば、5 年間保護する必要がある情報に対してデジタル署名が 2009 年に生成され、特定の(L, N) ペアが
2010 年以降無効になった場合、残っているより大きな(L, N) ペアが使用されます。情報を保護する必要がある全期間有効です。

(L, N) ペアのセキュリティ強度とデジタル署名の生成に使用されるハッシュ関数のセキュリティ強度は、より強力なハッシュ関数を使用するという参加エンティティ間での合意がない限り、同じであることが推奨されます。ハッシュ関数の出力の長さがN (つまり、q のビット長)より大きい場合、ハッシュ関数出力ブロックの左端のNビットは、生成または生成中にハッシュ関数出力を使用する計算に使用されます。デジタル署名の検証。 (L, N) ペアよりも低いセキュリティ強度を提供するハッシュ関数は、通常は使用しないでください。これは、デジタル署名プロセスのセキュリティ強度がハッシュ関数によって提供されるレベル以下に低下するためです。

認証局 (CA) 以外の連邦政府機関は、最初の 3 つの(L, N) ペア (つまり、(1024, 160)、(2048, 224)、および (2048, 256) ペア) のみを使用する必要があります。 CA は、その加入者が使用する(L, N) ペア以上の(L, N) ペアを使用する必要があります。たとえば、加入者が (2048, 224) ペアを使用している場合、CA は(2048, 224)、(2048, 256)、または (3072, 256) ペアのいずれかを使用します。この規則の例外として考えられるのは、CA 間の相互認証、デジタル署名以外の目的での鍵の認証、ある鍵のサイズやアルゴリズムから別の鍵のサイズやアルゴリズムへの移行などです。詳細については、SP 800-57 を参照してください。

4.3 DSA ドメインパラメータ

DSA では、デジタル署名の生成と検証に使用される秘密鍵と公開鍵のペアが、特定のドメイン パラメータのセットに関して生成されることが必要です。これらのドメイン パラメータは、ユーザーのグループに共通であり、公開されている場合があります。ドメイン パラメータのセットのユーザー (つまり、署名者と検証者の両方) は、それらを使用する前にその有効性を保証するものとします (セクション 3 を参照) 。ドメインパラメータは公開情報である可能性がありますが、
特定のキー ペアとそのドメイン パラメータのセットの間の正しい対応関係が、そのキー ペアを使用するすべての当事者に対して維持されるように管理されます。一連のドメインパラメータは、
長期間固定されたままになります。

DSA のドメイン パラメーターは、整数p、 q、 g であり、オプションで、 pとqの生成に使用されたdomain_parameter_seedおよびcounter (つまり、ドメイン パラメーターの完全なセットは(p, q, g {, domain_parameter_seed,カウンター})) 。

4.3.1 ドメインパラメータの生成

ドメイン パラメータは、信頼できる第三者 (CA などの TTP) または TTP 以外のエンティティによって生成される場合があります。ドメインパラメータの有効性の保証は、キーペアの生成、デジタル署名の生成、またはデジタル署名の検証の前に取得されるものとします (セクション 3 を参照) 。

整数pおよびq は、付録 A.1 の指定に従って生成されます。生成プロセスへの入力は、 LとNの選択された値です(セクション 4.2 を参照)。生成プロセスの出力は、 pとqの値であり、オプションで、domain_parameter_seedとcounterの値も得られます。

ジェネレータg は、付録 A.2 の指定に従って生成されます。

ドメインパラメータの生成時に使用されるハッシュ関数のセキュリティ強度
(L, N) ペアに関連付けられたセキュリティ強度以上である必要があります。これは、デジタル署名プロセスに使用できるハッシュ関数よりも制限が厳しいことに注意してください (セクション 4.2 を参照)。

4.3.2 ドメインパラメータ管理

各デジタル署名キーのペアは、 (たとえば、公開キーに関連付けられたドメイン パラメーターを識別する公開キー証明書によって) ドメイン パラメーターの 1 つの特定のセットに正しく関連付けられるものとします。ドメイン パラメータは、セットが非アクティブ化されるまで (セットが不要になった場合)、不正な変更から保護されます。同じドメイン パラメータが複数の目的に使用される場合があります (たとえば、同じドメイン パラメータがデジタル署名と鍵の確立の両方に使用される場合があります)。ただし、ジェネレータgに異なる値を使用すると、

ある目的のために生成されたキー ペアが、別の目的に誤って (うまく) 使用されるリスクを軽減します。

4.4 キーペア

各署名者は、互いに数学的に関連する秘密鍵xと公開鍵yという鍵ペアを持っています。秘密キーは、デジタル署名が生成される一定の期間 (つまり、秘密キーの暗号化期間) のみ使用されます。関連する秘密鍵を使用して生成されたデジタル署名を検証する必要がある限り、公開鍵は使用し続けることができます (つまり、公開鍵は、関連する秘密鍵の暗号化期間を超えて使用し続けることができます)。詳細については、SP 800-57 を参照してください。

4.4.1 DSA キー ペアの生成

デジタル署名キーのペアxおよびyは、一連のドメイン パラメーター(p、 q、 g {、 domain_parameter_seed、counter}) に対して生成されます。 xとyの生成方法は付録 B.1 に記載されています。

4.4.2 キーペアの管理

キー ペアの保護に関するガイダンスは SP 800-57 で提供されます。デジタル署名を安全に使用できるかどうか
は、次のようにエンティティのデジタル署名キー ペアの管理に依存します。

1. ドメインパラメータの有効性は、鍵ペアの生成、またはデジタル署名の検証および妥当性確認の前に保証されるものとします (セクション 3 を参照) 。
2. 各キー ペアは、キー ペアが使用されるドメイン パラメータに関連付けられます。
   生成されました。
3. キーペアは、ドメインを使用して署名を生成および検証するためにのみ使用されます。
   そのキーペアに関連付けられたパラメータ。
4. 秘密鍵は、この標準で指定されている署名生成にのみ使用され、秘密に保たれます。公開キーは署名の検証にのみ使用され、公開される場合があります。
5. 意図された署名者は、デジタル署名を生成するために秘密鍵を使用する前またはそれと同時に、秘密鍵を所有していることを保証するものとします (セクション 3.1 を参照) 。
6. 秘密キーは、不正なアクセス、開示、変更から保護されなければなりません。
7. 公開鍵は、不正な変更（置き換えを含む）から保護されなければなりません。
   たとえば、CA によって署名された公開キー証明書は、そのような保護を提供する場合があります。
8. 検証者は、公開鍵とその関連ドメイン間のバインディングを保証されるものとする。
   パラメータとキーペアの所有者 (セクション 3 を参照)。
9. 検証者は、信頼できる方法で公開鍵を取得する必要があります（たとえば、エンティティが信頼する CA によって署名された証明書から、またはエンティティが検証者によって信頼され、エンティティが検証者によって信頼され、認証できる場合は、意図された署名者または主張された署名者から直接） 。検証される署名情報のソース）。
10. 検証者は、主張された署名者が鍵ペアの所有者であること、および
    所有者は、署名の生成時にデジタル署名の生成に使用された秘密鍵 (つまり、デジタル署名の検証に使用される公開鍵に関連付けられた秘密鍵) を所有していました (セクション 3.3 を参照)。
11. 署名者と検証者は、公開鍵の有効性を保証するものとします（「公開鍵」を参照） 。
    セクション 3.1 および 3.3)。

4.5 DSA メッセージごとの秘密番号

新しい秘密乱数k は、署名生成プロセス中に使用する各デジタル署名の生成に先立って生成されます。この秘
密番号は、不正な開示や変更から保護されます。

k^{-1} は、 qを法とする乗算に関するkの逆乗です。つまり、$0 < k < q$ $^{-1}$

$1 = (k\,k) \bmod^{-1} q$。この逆は、署名生成プロセスに必要です (セクション 4.6 を参照)。 kからkを導出する手法は付録 C.1 に記載されています。

署名される$^{-1}$メッセージの知識は計算に必要ないため、 kおよびk $^{-1}$ は事前に計算されてもよい。 kとk $^{-1}$ が事前に計算される場合、その機密性と完全性は保護されません。

メッセージごとの秘密番号の生成方法は、付録 B.2 に記載されています。

4.6 DSA 署名の生成

意図された署名者は、セクション 3.1 に規定されている保証を有するものとします。

Nをqのビット長とします。 min(N, outlen)が正の整数Nの最小値を表すものとします。
およびoutlen。outlen はハッシュ関数出力ブロックのビット長です。

メッセージMの署名は、次の方程式に従って計算される数値rとsのペアで構成されます。

$r = (g^{k} \bmod p) \bmod q$。

z = Hash(M)の左端のmin(N, outlen)ビット。

$s = (k^{-1}(z + xr)) \bmod q$。

sを計算するとき、 Hash(M)から取得された文字列z は整数に変換されます。変換ルールは付録 C.2 に記載されています。

k、 p、 q 、 gが利用可能なときはいつでもrを計算できることに注意してください。たとえば、ドメインパラメータp、 q 、 gが既知で、 kが事前に計算されているとき (セクション 4.5 を参照)、 rも事前に計算される可能性があります。署名されるメッセージの知識は計算には必要ないため、計算されます。
rの。事前に計算されたk、 k^{-1} そしてr値はプライベートと同じ方法で保護されます。
sが計算されるまでキーx を押し続けます (SP 800-57 を参照)。

rとsの値は、 r = 0 かs = 0かを決定するためにチェックされます。r = 0またはs = 0 の場合、 kの新しい値が生成され、署名が再計算されます。署名が適切に生成された場合、 r = 0 またはs = 0になる可能性は非常に低いです。

署名(r, s) はメッセージとともに検証者に送信されます。

4.7 DSA 署名の検証と検証

署名検証は、署名者の公開鍵を使用して、任意の当事者 (つまり、署名者、対象受信者、またはその他の当事者) によって実行できます。署名者は、署名されたメッセージを目的の相手に送信する前に、計算された署名が正しいことを検証したい場合があります。

受信者。意図された受信者 (またはその他の当事者) は、署名を検証してその信頼性を判断します。

署名されたメッセージの署名を検証する前に、ドメイン パラメータ、主張された署名者の公開鍵と ID が認証された方法で検証者に利用可能になるものとします。公開キーは、たとえば、信頼できるエンティティ (たとえば CA) によって署名された証明書の形式で、または公開キーの所有者との直接の会議で取得できます。

M '、 r'、およびs'を、それぞれM、 r、およびsの受信バージョンとする。 yを主張された署名者の公開鍵とする。 Nを qのビット長とします。また、 min(N, outlen)が正の整数Nとoutlenの最小値を表すものとします。ここで、 outlenはハッシュ関数出力ブロックのビット長です。

署名検証プロセスは次のとおりです。

1. 検証者は、 0 < r' < qおよび 0 < s' < qであることを確認します。いずれかの条件に違反した場合、署名は無効として拒否されます。
2. ステップ 1 の 2 つの条件が満たされる場合、検証者は以下を計算します。

   $w = (s')^{-1}$ モッドq。

   z = Hash(M) の左端のmin(N, outlen)ビット。

   u1 = (zw) mod q。

   u2 = ((r')w) mod q。

   $v = (((g)^{u1} (y)^{u2}) \text{ モッド} p) \text{ モッド} q$。

   (s)-1 (つまり、 s' mod qの逆数)を導出する手法が付録 C.1 に提供されています。

   Hash(M)から取得した文字列z は整数に変換されます。変換ルールは付録 C.2 に記載されています。

3. v = rの場合、その後、署名が検証されます。 M= M、 r= r、 sのとき、 v = rであることの証明について = s、付録 E を参照してください。
4. v がrに等しくない場合、その場合、メッセージまたは署名が変更されたか、署名者の生成プロセスにエラーがあった可能性があります。または、詐欺師 (主張された署名者の公開鍵に関連付けられた秘密鍵を知らなかった人) がメッセージを偽造しようとした可能性があります。サイン。署名は無効とみなされます。データが有効かどうかについて推論することはできません。署名を検証するために公開キーを使用した場合、そのデータの署名が正しくないということのみがわかります。
5. 署名を有効なものとして受け入れる前に、検証者は、以下に指定されている保証を得る必要があります。
   セクション3.3。

組織のポリシーによって、無効なデジタル署名に対して実行されるアクションが規定される場合があります。このようなポリシーは、この規格の範囲外です。デジタル署名されたメッセージの適時性の判断に関するガイダンスは、SP 800-102、デジタル署名の適時性に関する推奨事項で取り上げられています。

5. RSA デジタル署名アルゴリズム

デジタル署名の生成と検証における RSA アルゴリズムの使用は、American National Standard (ANS) X9.31 および Public Key Cryptography Standard (PKCS) #1 で規定されています。
これらの標準はそれぞれ RSA アルゴリズムを使用していますが、デジタル署名が生成される ANS X9.31 および PKCS #1 データの形式は、アルゴリズムを構成する詳細が異なります。
交換不可。

5.1 RSA キーペアの生成

RSA デジタル署名キーのペアは、デジタル署名の計算に使用される RSA 秘密キーと、デジタル署名の検証に使用される RSA 公開キーで構成されます。デジタル署名に使用される RSA キー ペアは、 1 つのデジタル署名スキーム (例、ANS X9.31、RSASSA-PKCS1 v1.5 または RSASSA-PSS、セクション 5.4 および 5.5 を参照) にのみ使用されます。さらに、RSA デジタル署名キー ペアは、他の目的 (キーの確立など) に使用してはなりません。

RSA 公開キーは、2 つの正の素数pとq (つまり、 n = pq) の積である法 n と公開キー指数e で構成されます。したがって、RSA 公開キーは値のペア(n, e) であり、デジタル署名を検証するために使用されます。 RSA キー ペアのサイズは、一般にビット単位の法nの長さ(nlen)であると考えられます。

対応する RSA 秘密鍵は、同じ法nと秘密鍵指数dで構成されます。
それはnと公開鍵指数eに依存します。したがって、RSA 秘密キーは値のペア(n, d) であり、デジタル署名の生成に使用されます。中国剰余定理 (CRT) を使用して(n, d)を表す別の方法が許可されていることに注意してください。

デジタル署名プロセスにセキュリティを提供するために、2 つの整数pとq、および秘密キーの指数dは秘密に保たれます。法nと公開鍵指数e は誰にでも知られる可能性があります。これらの値の保護に関するガイダンスは、SP 800-57 で提供されます。

この規格では、モジュラスの長さ (nlen) の 3 つの選択肢 (1024、2048、および 3072 ビット) を指定しています。連邦政府機関は、これらの選択肢の 1 つ以上を使用してデジタル署名を生成するものとします。

FIPS 180 で指定されている承認済みのハッシュ関数は、キー ペアとデジタル署名の生成中に使用されます。 RSA キー ペアの生成中に (この規格で指定されているように) 使用される場合、ハッシュ関数出力ブロックのビット単位の長さは、モジュラス n のビット長に関連付けられたセキュリティ強度を満たすか、それを超えていなければなりません(SP 800-57 を参照) ） 。

RSA デジタル署名プロセスに関連付けられたセキュリティ強度は、モジュラスのビット長に関連付けられたセキュリティ強度と、使用されるハッシュ関数のセキュリティ強度の最小値を超えません。デジタル署名プロセス中に使用されるモジュラス長およびハッシュ関数ごとのセキュリティ強度は、SP 800-57 で提供されます。使用されるハッシュ関数のセキュリティ強度とビット長に関連付けられたセキュリティ強度の両方

法nの値は、デジタル署名プロセスに必要なセキュリティ強度を満たすか、それを超えていなければなりません。

参加エンティティ間でより強力なハッシュ関数を使用するという合意が形成されていない限り、モジュラスのセキュリティ強度とハッシュ関数のセキュリティ強度を同じにすることが推奨されます。モジュラスのビット長に関連付けられたセキュリティ強度よりも低いセキュリティ強度を提供するハッシュ関数は、通常は使用しないでください。デジタル署名プロセスのセキュリティ強度がハッシュによって提供されるレベル以下に低下するためです。関数。

CA 以外の連邦政府機関は、 SP 800-57 に示されている期間中、最初の 2 つの選択肢 (つまり、 nlen = 1024 または 2048) のみを使用する必要があります。 CA は、長さnlenがその加入者が使用するモジュラス以上のモジュラスを使用する必要があります。たとえば、加入者がnlen = 2048 を使用している場合、CA はnlen ≥ 2048を使用する必要があります。SP 800-57 は、n のビット長の選択に関する詳細情報を提供します。この規則の例外として考えられるのは、CA 間の相互認証、デジタル署名以外の目的での鍵の認証、ある鍵のサイズやアルゴリズムから別の鍵のサイズやアルゴリズムへの移行などです。

RSA 鍵ペアの生成基準は、付録 B.3.1 に記載されています。

RSA パラメータがランダムに生成される場合 (つまり、素数pとq、およびオプションで公開鍵指数e)、承認されたランダム ビット ジェネレータを使用して生成されます。ランダムビット発生器によって生成される（擬似）ランダムビットは、 ＲＳＡパラメータを生成するためのシードとして使用されるものとする（例えば、（擬似)乱数は素数生成シードとして使用される）。
素数生成シードは秘密に保たれるか、法nが計算されるときに破棄されます。素数生成シードが（たとえば、RSA 法n を再生成するため、または生成された素因数pおよびqがこの標準に従って生成されたという証拠として)保持される場合、シードは秘密にされ、保護されるものとします。この保護の強度は、関連する秘密キーに必要な保護と (少なくとも) 同等でなければなりません。

5.2 キーペアの管理

デジタル署名を安全に使用できるかどうかは、エンティティのデジタル署名キー ペアの管理に依存します。 RSA のキー ペア管理要件は、セクション 4.4.2 の要件 4 ～ 11 に記載されています。ドメイン パラメーターとキーペアの関係を扱うセクション 4.4.2 の最初の 3 つの要件は、RSA には適用されないことに注意してください。

5.3 保証

意図された署名者は、セクション 3.1 に規定されている保証を有するものとします。デジタル署名を有効なものとして受け入れる前に、検証者はセクション 3.3 で指定されている保証を得る必要があります。

5.4 ANS X9.31

ANS X9.31、金融サービス業界向けの可逆公開キー暗号化を使用したデジタル署名(rDSA) は、金融サービス認定標準委員会 X9 によって米国規格協会向けに開発されました。 ANS X9.31 および関連正誤表のコピーの入手方法については、 http://www.x9.orgを参照してください。以下の説明は、1998 年に承認された ANS X9.31 のバージョンに基づいています。

プライベート素因数pおよびqの生成方法は、付録 B.3 に記載されています。

ANS X9.31 では、法nの長さは256 ビット単位で許可されます。
最小 1024 ビット。 FIPS 186-4 への準拠を主張する実装には、セクション 5.1 で指定された 1 つ以上の係数サイズが含まれなければなりません。

ANS X9.31 には、デジタル署名を生成する 2 つの方法が含まれています。公開署名検証指数eが奇数の場合、デジタル署名アルゴリズムは一般に RSA として知られています。公開署名検証指数e が偶数の場合、デジタル署名アルゴリズムは一般に Rabin-Williams として知られています。この規格 (FIPS 186-4) は RSA の使用を採用していますが、Rabin-Williams の使用は採用していません。

署名検証中に、データ構造IR'からハッシュ値H(M)'が抽出されます。
次のいずれかによって達成されます。

- トレーラー情報の 2 バイトの直前にあるデータ構造IR'のハッシュレン バイトを選択します。ここで、ハッシュレンは、パディングの長さに関係なく、使用されるハッシュ関数のバイト単位の長さです。または

- ハッシュ値H(M)' がパディングの最後のバイトに対する位置によって選択された場合
  (つまり、0xBA)。これには、ハッシュ値の後に、予期されるトレーラー値を含む 2 バイトだけが続くかどうかのチェックが含まれます。

ANS X9.31 には、乱数生成に関する付録が含まれています。ただし、ANS X9.31 の実装では、承認された乱数生成方法を使用する必要があります。

ANS X9.31 の付録では、セキュリティと実装に関する考慮事項についての有益な議論が提供されています。

5.5 PKCS #1

RSA 暗号化標準である Public-Key Cryptography Standard (PKCS) #1 は、 RSA アルゴリズムを使用してデータを暗号化および署名するためのメカニズムを定義します。 PKCS #1 v2.1 では、RSASSA-PKCS1-v1.5 と RSASSA-PSS という 2 つのデジタル署名プロセスと対応する形式が指定されています。どちらの署名スキームも使用が承認されていますが、PKCS #1 v2.1 で指定されているものを超える追加の制約が課されます。

(a) RSA 鍵ペアを生成する実装では、次の基準と方法を使用する必要があります。
これらの鍵ペアを生成するには付録 B.3、

(b) 承認されたハッシュ関数のみを使用するものとします。

(c)法 n を形成するには、 2 つの素因数pおよびqのみを使用します。

(d) 乱数は、承認されたランダム ビット ジェネレーターを使用して生成されるものとします。

(e) RSASSA-PSS の場合:

- nlen = 1024 ビット (つまり、128 バイト) で、承認されたハッシュ関数出力ブロックの出力長が512 ビット (つまり、64 バイト) の場合、ソルト(sLen)の長さ (バイト単位) は0 を満たさなければなりません。 ≤ sLen ≤ hLen – 2、
- それ以外の場合、ソルト(sLen)の長さ (バイト単位) は0 ≤ sLen ≤ hLenを満たす必要があります。

ここで、hLen はハッシュ関数出力ブロックの長さ (バイト単位) です。

(f) RSASSA-PKCS-v1.5 の場合、エンコードされたメッセージからハッシュ値が復元される場合
EM はデジタル署名 1 の検証中に、ハッシュ値の抽出を行う必要があります。
次のいずれかによって達成されます。

- ハッシュのサイズに基づいて、EMの右端 (最下位) ビットを選択する
  パディングの長さに関係なく、使用される関数、または
- ハッシュ値がパディングの最後のバイトに対する位置によって選択される場合、ハッシュ値が右端 (最下位) に位置するかどうかのチェックが含まれます。
  EMのバイト数(つまり、エンコードされたメッセージのハッシュ値の後には他の情報は続きません)。

注: PKCS #1 は、1991 年に RSA Laboratories によって最初に開発され、複数のバージョンとして改訂されました。 FIPS 186-4 の承認時点では、PKSC #1 の 3 つのバージョン (バージョン 1.5、バージョン 2.0、およびバージョン 2.1) が利用可能でした。この規格はバージョン 2.1 のみを参照します。

[1] PKCS #1、v2.1 では、ハッシュ値を比較するための 2 つの方法が提供されています。エンコードされたメッセージEMとEM' を比較する方法と、エンコードされたメッセージのデコードからハッシュ値を抽出する方法です (PKCS #1、v2.1 の注を参照)。 1)。
上記のステップ (f) は後者の場合に適用されます。

6. 楕円曲線デジタル署名アルゴリズム (ECDSA)

ANS X9.62 (金融サービス業界向け公開キー暗号化: 楕円曲線デジタル署名標準(ECDSA)) は、金融サービス認定標準委員会 X9 によって米国規格協会向けに開発されました。 ANS X9.62 のコピーの入手に関する情報は、http://www.x9.org で入手できます。以下の説明は、2005 年に承認された ANS X9.62 のバージョンに基づいています。ANS X9.62 のこのバージョンは、

この規格の実施スケジュールに記載されている移行期間を条件として使用されます。

ANS X9.62 は、楕円曲線デジタル署名アルゴリズム (ECDSA) を使用したデジタル署名の生成および検証の方法を定義しています。デジタル署名の生成および検証中に使用されるドメイン パラメータの生成に関する仕様も、ANS X9.62 に含まれています。 ECDSA は、DSA の楕円曲線に似たものです。 ECDSA キーは、他の目的 (キーの確立など) に使用してはなりません。

6.1 ECDSA ドメインパラメータ

ECDSA では、デジタル署名の生成と検証に使用される秘密鍵と公開鍵のペアが、特定のドメイン パラメーターのセットに関して生成されることが要求されます。これらのドメイン パラメータは、ユーザーのグループに共通であり、公開されている場合があります。ドメインパラメータは長期間固定されたままになる場合があります。

ECDSA のドメイン パラメーターの形式は(q, FR, a, b {, domain_parameter_seed}, G, n, h) です。ここで、 qはフィールド サイズです。 FRは使用される基底を示します。 aとbは、曲線の方程式を定義する 2 つのフィールド要素です。 Domain_parameter_seedはドメイン パラメーター シードであり、楕円曲線が検証可能な方法でランダムに生成された場合に存在するオプションのビット文字列です。 Gは曲線上の素数次数の基点です (つまり、 G = (xG, yG))。 nは点 G の次数、 hは余因子 (曲線の次数を n で割ったものに等しい) です。

6.1.1 ドメインパラメータの生成

この規格では、 nの 5 つの範囲を指定しています(表 1 を参照)。各範囲の最大余因子サイズも指定されています。 ANS X9.62 では、ドメイン パラメータのセットにおける余因子hの指定はオプションですが、この規格 (つまり、FIPS 186-4) に準拠する実装では、ドメイン パラメータのセットにおける余因子hを指定する必要があることに注意してください。表 1 は、補因子 h の最大サイズを示しています。

表 1: ECDSA セキュリティ パラメータ

nのビット長	最大 補因子(h)
160～223	210
224～255	214
256 - 383	216
384 - 511	224
≥ 512	232

ECDSA は、有限体GFpと有限体 GF という2 つの算術フィールドに対して定義されます。フィールドGFp 、 pは奇数 。のために
の素数である必要があります。

NIST 推奨曲線は、この規格の付録 D (つまり、FIPS 186-4) に記載されています。
3 種類の曲線が提供されています。

1. プライムフィールド上の曲線。P-xxx として識別されます。
2. B-xxx として識別されるバイナリ フィールド上の曲線、および
3. K-xxx として識別されるコブリッツ曲線、

ここで、xxx はフィールド サイズのビット長を示します。

あるいは、ECDSA ドメイン パラメータは、ANS X9.62 で指定されているように生成することもできます。 ECDSA ドメイン パラメータが生成されるとき (つまり、NIST 推奨曲線が使用されないとき)、 Gの値は標準的に (検証可能なランダムで) 生成される必要があります。 FIPS 180 で指定されている承認済みのハッシュ関数は、 ECDSA ドメイン パラメータの生成中に使用されます。
これらのドメインパラメータを生成する際、使用されるハッシュ関数のセキュリティ強度は、
nのビット長に関連付けられたセキュリティ強度を満たすか、それを超えています(以下の脚注 2 を参照)。

ドメインの生成時には、FIPS 180 で指定されている承認済みのハッシュ関数が必要です
パラメーター。使用されるハッシュ関数のセキュリティ強度は、 nのビット長に関連付けられたセキュリティ強度以上である必要があります。 nの範囲とハッシュ関数のセキュリティ強度は SP 800-57 で提供されます。 nのビット長に伴うセキュリティ強度とハッシュ関数のセキュリティ強度は同じにすることを推奨します。

---

[2] NIST 推奨曲線は、このガイダンスの策定前に、当時利用可能で唯一承認されたハッシュ関数であった SHA-1 を使用して生成されました。 SHA-1 は生成時に安全であると考えられ、曲線は公開され、SHA-1 はそれらの曲線を検証するためにのみ使用されるため、NIST 推奨曲線は依然として安全であり、連邦政府の使用に適していると考えられています。

ただし、より強力なハッシュ関数を使用するという参加エンティティ間での合意がなされている場合を除きます。 nのビット長に関連付けられているセキュリティ強度よりも低いセキュリティ強度を提供するハッシュ関数は使用されません。ハッシュ関数の出力の長さがnのビット長より大きい場合、ハッシュ関数出力ブロックの左端のnビットは、デジタル署名の生成または検証中にハッシュ関数出力を使用するあらゆる計算に使用されます。 。

通常、CA は、評価されたセキュリティ強度が、その加入者が使用するnのビット長に関連付けられた評価されたセキュリティ強度以上であるビット長nを使用する必要があります。たとえば、加入者が評価されたセキュリティ強度が 112 ビットでビット長nを使用している場合、CA は評価されたセキュリティ強度が 112 ビット以上であるビット長nを使用する必要があります。 SP 800-57 には、n のビット長の選択に関する詳細情報が記載されています。この規則の例外として考えられるのは、CA 間の相互認証、デジタル署名以外の目的での鍵の認証、ある鍵のサイズやアルゴリズムから別の鍵のサイズやアルゴリズムへの移行などです。ただし、これらの例外についてはさらなる分析が必要です。

6.1.2 ドメインパラメータ管理

各 ECDSA 鍵ペアは、 (たとえば、公開鍵に関連付けられたドメイン パラメータを識別する公開鍵証明書によって) ドメイン パラメータの 1 つの特定のセットに正しく関連付けられるものとします。ドメイン パラメータは、セットが非アクティブ化されるまで (セットが不要になった場合)、不正な変更から保護されます。同じドメイン パラメータが複数の目的に使用される場合があります (たとえば、同じドメイン パラメータがデジタル署名と鍵の確立の両方に使用される場合があります)。ただし、異なるドメイン パラメーターを使用すると、ある目的で生成されたキー ペアが別の目的で誤って (正常に) 使用されるリスクが軽減されます。

6.2 秘密鍵/公開鍵

ECDSA キー ペアは、特定の ECDSA ドメイン パラメーターのセットに関連付けられた秘密キーdと公開キーQで構成されます。 d、 Q 、およびドメインパラメータは数学的に相互に関連しています。秘密キーは通常、一定期間 (つまり、暗号化期間) 使用されます。公開鍵は、デジタル署名が有効である限り使用され続ける可能性があります。
関連する秘密鍵を使用して生成された鍵は検証される必要があります (つまり、公開鍵は、関連する秘密鍵の暗号化期間を超えて使用され続ける可能性があります)。詳細については、SP 800-57 を参照してください。

ECDSA キーは、 ECDSA デジタル署名の生成と検証にのみ使用されます。

6.2.1 キーペアの生成

デジタル署名キーのペアdおよびQは、一連のドメイン パラメーター(q、FR、a、b {、
ドメインパラメータシード}、 G、n、 h)。 dとQの生成方法は付録 B.4 に記載されています。

6.2.2 キーペアの管理

デジタル署名の安全な使用は、セクション 4.4.2 で指定されているエンティティのデジタル署名キー ペアの管理に依存します。

6.3 秘密番号の生成

新しい秘密乱数k は、署名生成プロセス中に使用する各デジタル署名の生成に先立って生成されます。この秘密番号は、不正な開示や変更から保護されます。メッセージごとの秘密番号の生成方法は、付録 B.5 に記載されています。

k^{-1}は、nを法とする乗算に関するkの逆乗です。つまり、$0 < k < n$
$1 = (k\ k^{-1}) \bmod n$。この逆は、署名生成プロセスに必要です。 kからkを導出する手法は付録 C.1 に記載されています。

署名されるk^{-1}メッセージの知識は計算に必要ないため、 kおよびk は事前に計算されてもよい。 kとk が事前に計算される場合、その機密性と完全性は保護されます。

6.4 ECDSA デジタル署名の生成と検証

ECDSA デジタル署名(r、 s)は、 ANS X9.62 の規定に従って、以下を使用して生成されます。

1. セクション 6.1.1 に従って生成されるドメインパラメータ
2. セクション 6.2.1 の指定に従って生成される秘密キー
3. セクション 6.3 で指定されているように生成されるメッセージごとの秘密番号。
4. 以下で説明する承認済みのハッシュ関数、および
5. 承認されたランダム ビット ジェネレーター。

ECDSA デジタル署名は、署名生成時に使用されたのと同じドメイン パラメータとハッシュ関数を使用して、ANS X9.62 の規定に従って検証されます。

FIPS 180 で指定されている承認済みのハッシュ関数は、デジタル署名の生成中に使用されます。 ECDSA デジタル署名プロセスに関連付けられたセキュリティ強度は、 nのビット長に関連付けられたセキュリティ強度と、使用されるハッシュ関数のセキュリティ強度の最小値を超えません。ハッシュのセキュリティ強度

使用される関数とnのビット長に関連付けられたセキュリティ強度は、デジタル署名プロセスに必要なセキュリティ強度を満たすか、それを超えていなければなりません。 nのビット長の範囲および各ハッシュ関数のセキュリティ強度は、SP 800-57 で提供されます。

参加エンティティ間でより強力なハッシュ関数を使用するという合意が形成されていない限り、 nのビット長に関連するセキュリティ強度とハッシュ関数のセキュリティ強度は同じであることが推奨されます。ハッシュ関数の出力の長さがnのビット長より大きい場合、ハッシュ関数出力ブロックの左端のnビットは、デジタル署名の生成または検証中にハッシュ関数出力を使用するあらゆる計算に使用されます。 。ハッシュ関数よりもセキュリティ強度が低いハッシュ関数。

nのビット長に関連付けられたセキュリティ強度は、通常は使用すべきではありません。これは、デジタル署名プロセスのセキュリティ強度がハッシュ関数によって提供されるレベル以下に低下するためです。

6.5 保証

意図された署名者は、セクション 3.1 に規定されている保証を有するものとします。署名を有効なものとして受け入れる前に、検証者はセクション 3.3 で指定されている保証を得る必要があります。

付録 A: FFC ドメイン パラメータの生成と検証

有限体暗号 (FFC) は、有限体数学を使用して離散対数暗号を実装する方法です。この規格で指定されている DSA は、FFC の一例です。 SP 800-56A で指定されている Diffie-Hellman および MQV キー確立アルゴリズムは、FFC として実装することもできます。

FFC のドメイン パラメーターは、値のセット(p、 q、 g {、 domain_parameter_seed、 counter}) で構成されます。この付録では、FFC ドメイン パラメータp、 q 、 gの生成と、明示的なドメイン パラメータ検証の実行のための手法を指定します。生成プロセス中に、 domain_parameter_seedとcounterの値が取得されます。

A.1 FFC 素数pおよびqの生成

このセクションでは、セクション 4.1 および 4.2 で指定された基準を満たす素数pおよびqを生成する方法を提供します。これらの素数を生成する際には、これらの方法の 1 つが使用されます。付録 A.1.1 では、ランダムな候補整数を生成し、確率的アルゴリズムを使用してそれらの素数をテストする方法が提供されています。付録 A.1.2 では、構築された整数が素数であることが保証されるように、より小さい整数から整数を構築する 2 番目の方法が提供されています。

生成、検証、テストのプロセス中に、ビット文字列と整数の間の変換が必要になります。付録 C.2 では、これらの変換のメソッドを提供します。

A.1.1 推定素数の生成と検証

この規格の以前のバージョンには、SHA-1 および確率的手法を使用してドメイン パラメータpおよびqを生成する方法が含まれていました。この方法は現在承認されていません
ドメインパラメータ生成用。ただし、以前に生成されたドメインパラメータを検証するために、このメソッドの検証プロセスが付録 A.1.1.1 に提供されています。

確率的手法を使用した素数pとqの生成と検証の方法は、付録 A.1.1.2 で提供されており、承認されたハッシュ関数の使用に基づいています。このメソッドは、確率素数を生成するために使用されます。このメソッドの検証プロセスは、付録 A.1.1.3 に記載されています。

確率的手法では、ハッシュ関数と任意のシード(domain_parameter_seed) を使用します。
任意のシードは、たとえば、ユーザーのお気に入りの数値、または承認された乱数発生器によって出力される乱数または疑似乱数など、何でもあり得ます。ドメインパラメータシード
必要な間隔でpおよびqの候補のシーケンスを決定し、確率的素数性テストを使用して素数性をテストします (付録 C.3 を参照)。このテストでは、候補が素数ではない (つまり、合成整数である) か、または「おそらく素数である」(つまり、合成整数が素数であると宣言される確率が非常に低い) であるかを判断します。 pとqは、

素数性テストに合格した最初の候補セットになります。 domain_parameter_seedに注意してください。

は、同じ方法を使用して生成されるドメイン パラメータの一意のセットごとに一意である必要があります。

A.1.1.1 SHA-1 を使用して生成された推定素数pおよびqの検証
この規格の以前のバージョンで指定されている

この素数検証アルゴリズムは、この標準の以前のバージョンで指定された素数生成アルゴリズムによって生成された素数pおよびqを検証するために使用されます。このアルゴリズムには、素数生成アルゴリズムから出力されたp、 q、 domain_parameter_seed 、およびcounterの値が必要です。

SHA1()を FIPS 180 で指定された SHA-1 ハッシュ関数とします。このメソッドのpとqを検証するには、次のプロセスまたは同等のプロセスを使用します。

入力：

1.p 、 q　　生成された素数pとq。

2.domain_parameter_seed pとqを生成するために使用されたシード。

3.カウンター　　生成時に決定されたカウント値。

出力：

1.ステータス　　検証プロシージャから返されるステータス。ステータスはVALIDまたはINVALIDのいずれかです。

プロセス：

1. (len (p) ≠ 1024) または(len (q) ≠ 160) の場合は、 INVALID を返します。

2. (カウンタ> 4095) の場合は、 INVALID を返します。

3.seedlen = len (domain_parameter_seed)。

4. (seedlen < 160) の場合は、 INVALID を返します。

5. computed_q = SHA1(domain_parameter_seed) ⊕ SHA1((domain_parameter_seed + 1) mod 2seedlen)。

6. computed_qの最初と最後のビット(つまり、159 番目と0 番目のビット)を 1 に設定します。

7. 付録 C.3 で指定されているように、 computed_qが素数であるかどうかをテストします。(computed_q ≠ q) または(computed_q が素数ではない)場合は、 INVALID を返します。

8.オフセット= 2。

9. i = 0の場合、 doをカウンターします

9.1　j = 0 ～ 6 の場合、次のようにします。

Vj = SHA1((domain_parameter_seed + offset + j) mod 2seedlen)。

9.2W = V0 + (V1 * 2160) + (V2 * 2320) + (V3 * 2480) + (V4 * 2640) + (V5 * 2800) +

((V6 mod 263) 2960)。

9.3 X = W + 21023。 コメント: 0 ≤ W < 2L–1 。

9.4 c = X mod 2q。

9.5 computed_p = X – (c – 1)。コメント: computed_p ≡ 1 (mod 2q)。

9.6 (computed_p < 21023)の場合は、ステップ 9.8 に進みます。

9.7 付録 C.3 で指定されているように、 computed_pが素数であるかどうかをテストします。
computed_pが素数であると判断された場合は、ステップ 10 に進みます。

9.8 オフセット=オフセット+ 7。

10. ((i ≠ counter)または(computed_p ≠ p) または(computed_pが素数ではない)) の場合、return 無効。

11.有効を返します。

A.1.1.2承認されたハッシュ関数を使用した確率素数pおよびqの生成

この方法は承認されたハッシュ関数を使用しており、素数pの生成に使用できます。
qは任意のアプリケーション (デジタル署名や鍵の確立など) を表します。ハッシュ関数のセキュリティ強度は、 (L, N) ペアに関連付けられたセキュリティ強度以上である必要があります。

また、 seedlenビットの任意のdomain_parameter_seedも使用されます。ここで、 seedlenはN以上でなければなりません。

生成プロセスでは、素数である可能性が非常に高い整数pとqのセットが返されます。付録 A.1.1.3 の検証プロセスを使用して素数が正しく生成されたことを別のエンティティが検証するには、 domain_parameter_seedの値と素数の生成に使用されたカウンタも返され、検証エンティティが利用できるようにする必要があります。 domain_parameter_seedとcounter を秘密にしておく必要はありません。 Hash()を選択したハッシュ関数とし、 outlen を出力ブロックのビット長とし、 outlen はN以上であるものとします。

このメソッドのpとqを生成するには、次のプロセスまたは同等のプロセスが使用されます。

入力：

1.L _ 素数pの希望の長さ(ビット単位)。

2.N _ 素数qの希望の長さ(ビット単位)。

3.シーレン ドメインパラメータシードの必要な長さ。 seedlen はN以上である必要があります。

出力：

1.ステータス 生成プロシージャから返されるステータス。ここで、ステータスは

有効または無効のいずれかです。 INVALIDがステータスとして返された場合、他の出力パラメータの値が返されないか、無効な値(ゼロまたはヌル文字列など) が返されます。

2.p 、 q 生成された素数pとq。

3.ドメインパラメータシード

(オプション) pおよびqの生成に使用されたシード。

4.カウンター (オプション) 生成中に決定されたカウント値。

プロセス：

1. (L, N) ペアが許容可能な(L, Nペア)のリストに含まれていることを確認します (セクション 4.2 を参照)。もしペアがリストにない場合は、 INVALID を返します。

2. (seedlen < N) の場合は、 INVALID を返します。 3. n =

L outlen – 1。

4. b = L – 1 – (n outlen)。

5. 任意のseedlenビットのシーケンスを、 domain_parameter_seedとして取得します。

6. U =ハッシュ(domain_parameter_seed) mod 2N–1 7. q = 2N–1 +

U + 1 – (U mod 2)。

8. 付録 C.3 で指定されているように、 qが素数であるかどうかをテストします。

9. qが素数でない場合は、ステップ 5 に進みます。

10.オフセット= 1。

11.カウンタ = 0 ～ (4L – 1) の場合、次のようにします。

11.1 j = 0 ～ nの場合、 Vj = Hash

((domain_parameter_seed + offset + j) mod 2seedlen) を実行します。

11.2 W = V0 + (V1 2outlen) + … + (Vn–1 2(n–1) outlen) + ((Vn mod 2b) 2n outlen)。

11.3 X = W + 2L–1コメント: 0 ≤ W < 2L–1 ;したがって、 2L–1 ≤ X < 2L

11.4 c = X mod 2q。

11.5 p = X – (c – 1)。 コメント: p ≡ 1 (mod 2q)。

11.6 (p < 2L–1) の場合、ステップ 11.9 に進みます。

11.7 付録 C.3 で指定されているように、 pが素数であるかどうかをテストします。

11.8 pが素数であると判定された場合、 VALIDとp、 q 、およびの値を返します。
(オプション) domain_parameter_seed と counterの値。

11.9オフセット=オフセット+ n + 1。 コメント:オフセットを増分します。次に、ステップ 11 のループの一部として、カウンターをインクリメントします。カウンタ< 4Lの場合は、ステップ 11.1 ～ 11.8 を繰り返します。

12. 手順 5 に進みます。

A.1.1.3を使用して生成された推定素数pおよびqの検証
承認されたハッシュ関数

この素数検証アルゴリズムは、整数pおよびq が付録 A.1.1.2 に示されている素数生成アルゴリズムによって生成されたことを検証するために使用されます。検証アルゴリズムには、素数生成アルゴリズムから出力されたp、 q、 domain_parameter_seed 、およびcounterの値が必要です。 Hash( )をpとqの生成に使用されるハッシュ関数とし、 outlenとします。
は出力ブロック長になります。

このメソッドのpとqを検証するには、次のプロセスまたは同等のプロセスを使用します。

入力：

1.p 、 q 生成された素数pとq。

3.domain_parameter_seed pと_ _ q.

4.カウンター 生成時に決定されたカウント値。

出力：

1.ステータス 検証プロシージャから返されるステータス。ステータスはVALIDまたはINVALID のいずれかです。

プロセス：

1. L =レン(p)。

2. N = len (q)。

3. (L, N) ペアが許容可能な(L, N) ペアのリストに含まれていることを確認します(セクション 4.2 を参照)。ペアがリストにない場合は、 INVALID を返します。

4. (カウンター> (4L – 1)) の場合は、 INVALID を返します。

5.seedlen = len (domain_parameter_seed)。

6. (seedlen < N) の場合は、 INVALID を返します。

7. U =ハッシュ(domain_parameter_seed) mod 2N–1 。

8. computed_q = 2N–1 + U + 1 – (U mod 2)。

9. 付録 C.3 で指定されているように、 computed_qが素数であるかどうかをテストします。 If (computed_q ≠ q) または(computed_qが素数ではない) の場合は、 INVALID を返します。

10. n = L outlen – 1.

11. b = L – 1 – (n outlen)。

12.オフセット= 1。

13. i = 0の場合、 doをカウンターします

13.1 j = 0 ～ nの場合､次のことを行います

Vj =ハッシュ((domain_parameter_seed + offset + j) mod 2seedlen)。

13.2 W = V0 + (V1 2outlen) + …+ (Vn–1 2(n–1) outlen) + ((Vn mod 2b) 2n outlen)。

13.3 X = W + 2L–1 。

13.4 c = X mod 2q。

13.5計算済み_p = X – (c – 1)。

13.6 (computed_p < 2L–1) の場合､ステップ 13.9 に進みます。

13.7 付録 C.3 で指定されているように、 computed_pが素数であるかどうかをテストします。

13.8 computed_pが素数であると判断された場合は､ステップ 14 に進みます。

13.9オフセット=オフセット+ n + 1。

14. ((i ≠ counter)または(computed_p ≠ p) または(computed_pが素数ではない)) の場合､return 無効。

15.有効を返します。

A.1.2 証明可能な素数pおよびqの構築と検証

素数は､素数であることが保証されるように生成できます。 pとqを生成する次のアルゴリズムは､承認されたハッシュ関数と任意のシード(firstseed)を使用してpを構築します。
およびq を必要な間隔で入力します｡ハッシュ関数のセキュリティ強度は、 (L, N) ペアに関連付けられたセキュリティ強度以上である必要があります。

任意のシードは､たとえば､ユーザーのお気に入りの数値､乱数発生器から出力される乱数または擬似乱数など､任意のシードにすることができます｡一意のドメイン パラメータのセットを生成するには、 firstseedが一意である必要があることに注意してください｡候補素数は､候補が素数であるかどうかを証明する決定論的素数性テストを使用して素数性についてテストされます｡結果として得られるpとq は素数であることが保証されます。

A.1.2.1 Shawe-Taylor アルゴリズムを使用した素数pおよびqの構築

生成されたドメイン パラメータのセットごとに､少なくともseedlenビットの任意の初期シード(firstseed)が決定されます｡ここで、 seedlen ≧ Nとなります。

生成プロセスでは､素数であることが保証されている整数pとqのセットが返されます｡のために

素数が正しく生成されたことを検証するための別のエンティティ (付録 A.1.2.2 の検証プロセスを使用)、firstseed の値、 2 つの計算されたシード(pseedおよびqseed) 、および素数の生成に使用されたカウンター(pgen_counterおよびqgen_counter)を検証エンティティが利用できるようにする必要があります。シードとカウンターを秘密にしておく必要はありません。 DSA のドメイン パラメーターはセクション 4.3 で(p, q, g {, domain_parameter_seed, counter}) として識別されます。 Shawe-Taylor アルゴリズムを使用してpとq を生成する場合、 domain_parameter_seedは 3 つのシード値(firstseed、pseed、 qseed) で構成され、カウンターはカウンター値のペア(pgen_counterとqgen_counter) で構成されます。

Hash( )を選択したハッシュ関数 (付録 A.1.2 を参照) とし、 outlenをそのハッシュ関数の出力ブロックのビット長とします。

A.1.2.1.1 最初のシードを取得する

この構築的なメソッドのファーストシードを生成するには、次のプロセスまたは同等のプロセスを使用します。

入力：

1.N _	qの長さ (ビット単位)。
2.シーレン	firstseed の長さ (seedlen ≥ N) 。

出力：

1.ステータス	生成プロシージャから返されるステータス。ステータスはSUCCESSまたはFAILUREのいずれかです。 FAILUREが返された場合は、 firstseed値が提供されないか、無効な値が返されます。
2.ファーストシード	最初に生成されたシード。

プロセス：

1. 最初のシード= 0。
2. N が許容可能な(L, N) ペアのリストに含まれていることを確認します(セクション 4.2 を参照)。そうでない場合は、FAILURE を返します。
3. (seedlen < N) の場合は、 FAILURE を返します。
4. 最初のシード< 2N–1の間 。

   任意のseedlenビットのシーケンスをfirstseedとして取得します。
5. SUCCESSと firstseed の値を返します。

注: このルーチンは、付録 A.1.2.1.2 の構成素数生成手順の最初に組み込むことができます。ただし、これはこの仕様では行われていないため、

付録 A.1.2.2 の検証プロセスは、構成素数生成プロシージャを呼び出して、入力としてfirstseedの値を提供することもできます。

A.1.2.1.2 構成素数の生成

この構築的な方法のpとqを生成するには、次のプロセスまたはそれと同等のプロセスが使用されます。

入力：

1.L _ pの要求された長さ(ビット単位)。

2.N _ qの要求された長さ(ビット単位)。

3.ファーストシード 使用される最初のシード。これは、付録 A.1.2.1.1 に指定されているように取得されました。

出力：

1.ステータス 生成プロシージャから返されたステータス。ステータスはSUCCESSまたはFAILUREのいずれかです。 FAILUREが返された場合は、他の値が返されないか、無効な値が返されます。

2.p 、 q 要求された素数。

3. pseed、 qseed (オプション) pとqの生成に使用された計算されたシード値。 pとqを生成するシード全体は、 firstseed、 pseed 、 qseedで構成されます。

4. pgen_counter、 qgen_counter

(オプション) 生成中に決定されたカウント値。

プロセス：

1. (L, N) ペアが許容可能な(L, N) ペアのリストに含まれていることを確認します(セクション 4.2 を参照)。もしペアがリストにない場合は、 FAILURE を返します。

コメント: Shawe-Taylor ランダムを使用します。付録 C.6 の prime ルーチンを使用して、ランダムな素数を生成します。

2. Nを長さとして、 firstseed をinput_seedとして使用し、付録 C.6 のランダム素数生成ルーチンを使用してq、 qseed 、およびqgen_counter を取得します。 FAILUREが返された場合は、FAILUREを返します。

3.長さとして「L / 2 + 1」を使用し、 input_seedとしてqseedを使用し、付録 C.6 のランダム プライム ルーチンを使用して、 p0 、pseed、およびpgen_counter を取得します。 FAILUREが返された場合は、FAILUREを返します。

4.反復= L / outlen –1。

5. old_counter = pgen_counter。

コメント:区間[2L－1 , 2L]で(擬似) ランダムxを生成します。

6. x = 0。

7. For i = 0 から反復を行う

x = x + (ハッシュ(pseed + i) 2 私*アウトレン)。

8. pseed = pseed +反復+ 1。 9. x = 2L－1

+ (x mod 2L–1)。

コメント:区間[2L－1 , 2L]で素数の候補 p を生成します。

10. t = x / (2q p0) 。

11. (2tq p0 + 1) > 2Lの場合、すると、 t = 2L－1 / (2q p0) となります。

12. p = 2tq p0 + 1。

13. pgen_counter = pgen_counter + 1。

コメント: pの素数をテストします。区間 [2, p–2] 内の整数aを選択します。

14. a = 0

15. For i = 0 から反復はdo

a = a + (ハッシュ(pseed + i) 2 私*アウトレン)。

16. pseed = pseed +反復回数+ 1。

17. a = 2 + (a mod (p–3))。

18. z = a 2tq mod p.

19. If ((1 = GCD(z–1, p)) および (1 = z p0 mod p)) の場合は、 SUCCESSと、p、 q 、および (オプションで) pseed、qseed、pgen_counter、およびqgen_counterの値を返します。

20. (pgen_counter > (4L + old_counter)) の場合、 FAILURE を返します。

21. t = t + 1。

22. 手順 11 に進みます。

A.1.2.2ショー テイラー アルゴリズムを使用して構築されたDSA 素数pおよびqの検証

firstseed、 pseed、 qseed、 pgen_counter 、およびqgen_counterの値が保存され、次のアルゴリズムで使用するために提供されている場合、付録 A.1.2.1.2 で説明されている方法によって生成された素数pおよびqの検証を実行できます。

この構築的な方法のpとqを検証するには、次のプロセスまたは同等のプロセスを使用します。

入力：

1.p 、 q　　検証される素数。

2. firstseed、pseed、 qseed pおよびqの生成に使用されたシード値。

3. pgen_counter、 qgen_counter

生成中に決定されたカウント値。

出力：

1.ステータス　　検証手順から返されるステータス。ステータスはSUCCESSまたはFAILUREのいずれかです。

プロセス：

1. L =レン(p)。

2. N = len (q)。

3. (L, N) ペアが許容可能な(L, N) ペアのリストに含まれていることを確認します(セクション 4.2 を参照)。もしペアがリストにない場合は、 FAILURE を返します。

4. (firstseed < 2N–1) の場合は、 FAILURE を返します。

5. (2N ≤ q) の場合は、 FAILURE を返します。

6. (2L ≤ p) の場合は、 FAILURE を返します。

7. ((p – 1) mod q ≠ 0) の場合、 FAILURE を返します。

8. L、 N 、およびfirstseed を使用して、構成素数生成手順を実行します。
p_val、 q_val、 pseed_val、 qseed_val、 pgen_counter_val、およびqgen_counter_valを取得するには、付録A.1.2.1.2 を参照してください。 FAILUREが返された場合、または(q_val ≠ q) または(qseed_val ≠)の場合

qseed)または(qgen_counter_val ≠ qgen_counter)または(p_val ≠ p) または(pseed_val ≠ pseed)または(pgen_counter_val ≠ pgen_counter)の場合は、FAILURE を返します。

9. SUCCESS を返します。

A.2 ジェネレータgの生成

ジェネレーターg は、 pとqの値に依存します。ジェネレータgを決定する 2 つの方法
提供されています;これらの方法のいずれかを使用する必要があります。付録 A.2.1 で説明されている最初の方法は、ジェネレーターgの完全な検証が必要ない場合に使用できます。この方法は、 gを生成する当事者が別の生成器g'と潜在的に悪用可能な関係を持つgを意図的に生成しないと信頼できる場合にのみ使用することをお勧めします。
たとえば、 g = (g')のようなジェネレーター間の指数関係を決定するのは困難に違いありません。
xの既知の値に対するmod p 。 (注: (g') と読んでください) g をxにプライムするものとして。)

付録 A.2.2 では、付録 A.2.1 の生成方法を使用する場合の部分検証の方法を提供します。付録 A.2.3 で説明されている gを生成する 2 番目の方法は、
ジェネレーターgの検証が必要な場合に使用されます。付録 A.2.3 の方法を使用して決定されたジェネレータの検証方法は、付録 A.2.4 で提供されます。

A.2.1 検証できないジェネレータgの生成

この方法は、 pとqの値に基づいて gの値を決定するために使用されます。ジェネレーターgの検証が必要ない場合に使用できます。 gの正しい生成を完全に検証することはできません (付録 A.2.2 を参照)。 gのこの生成方法は、この規格の以前のバージョンでも指定されていたことに注意してください。

このメソッドのジェネレータgを生成するには、次のプロセスまたは同等のプロセスを使用します。

入力：

1.p 、 q 生成された素数。

出力：

1.g _ 要求された g の値。

プロセス：

1. e = (p – 1)/q。

2. h = 1 < h < (p – 1)を満たす任意の整数を設定し、 h がどの値とも異なるようにします。
以前に試しました。 h は乱数発生器または乱数生成器から取得できることに注意してください。
使用するたびに変わるカウンターから。 3. g = he

mod p。

4. (g = 1) の場合、ステップ 2 に進みます。

5. g を返します。

A.2.2 ジェネレータgの有効性の保証

付録 A.2.1 を使用して生成されたジェネレータgの次数は、範囲と次数をチェックすることで部分的に検証でき、それによってg の部分検証を実行します。

ジェネレータgの部分的な検証が必要な場合は、次のプロセスまたは同等のプロセスを使用します。

入力：

1. p、 q、gドメインパラメータ。

出力：

1.ステータス	生成ルーチンから返されるステータス。ステータスはPARTIALLY VALIDまたはINVALIDのいずれかです。

プロセス：

1. 2 ≤ g ≤ (p–1) であることを確認します。 true でない場合は、 INVALID を返します。

2. (gq = 1 mod p) の場合は、 PARTIALLY VALID を返します。

3.無効を返します。

gと別の生成器 g' との潜在的に悪用可能な関係(g を生成したエンティティには知られているが、他のエンティティには知られていない可能性がある) が存在しないことをチェックすることはできません。
この意味で、 gの正しい生成を完全に検証することはできません。

A.2.3 ジェネレーターgの検証可能な正規生成

gの生成は、 p、 q 、 domain_parameter_seedの値(付録 A.1 の生成プロセスの出力) に基づいています。付録 A.1.1.2 の方法を使用してpおよびqを生成した場合、 domain_parameter_seed値が生成ルーチンから返されている必要があります。付録 A.1.2.1 の方法を使用してpおよびqが生成された場合、 firstseed、 pseed、およびqseed の値が生成ルーチンから返されている必要があります。この場合、 domain_parameter_seed = firstseed ||シード|| qseedは次のプロセスで使用されます。

ジェネレーターgを生成するこの方法は検証できます (付録 A.2.4 を参照)。

この生成方法は、 pの特定の値に対するgの複数の値の生成をサポートします。
そしてq.同じpとqに対して異なるg値を使用すると、キーの分離をサポートすることができます。たとえば、デジタル署名にはインデックス= 1で生成され、キーの確立にはインデックス= 2で生成されるgを使用します。

Hash( )をpとqの生成に使用されるハッシュ関数とします(付録 A.1 を参照)。ジェネレーター g を生成するには、次のプロセスまたはそれと同等のプロセスが使用されます。

入力：

1.p 、 q　　素数。

2.domain_parameter_seed pおよびq の生成中に使用されるシード。

3.インデックス　　gの生成に使用されるインデックス。 Index は、符号なし整数を表す長さ 8 のビット文字列です。

出力：

1.ステータス　　生成ルーチンから返されるステータス。ステータスはVALIDまたはINVALIDのいずれかです。

2.g _　　生成されたgの値。

プロセス：

注: count は符号なし 16 ビット整数です。

コメント:インデックスの有効な値が指定されていることを確認してください (上記を参照)。

1. (インデックスが正しくない) 場合は、 INVALID を返します。

2. N = len(q)。

3. e = (p – 1)/q。

4.カウント= 0。

5.カウント=カウント+ 1。

コメント:カウントが0 に戻っていないことを確認してください。

6. (カウント= 0) の場合は、 INVALID を返します。

コメント: domain_parameter_seed
の長さはすでにチェックされています。
「ggen」はビット列 0x6767656E です。

7. U = ドメインパラメータシード || 「ゲン」 ||インデックス ||カウント。

8. W =ハッシュ(U)。

9. g = pを修正します。

10. (g < 2) の場合、ステップ 5 に進みます。　　コメント: ジェネレーターが見つからなかった場合。

11. VALIDと g の値を返します。

A.2.4ジェネレータの正規生成時の検証ルーチン
ルーチンが使用されました

このアルゴリズムは、 p、 q、domain_parameter_seed の値、および適切なインデックスの値に基づいて、付録 A.2.3 のプロセスを使用して生成されたgの値を検証するために使用されます。 pとqの値は付録 A.1 に従って事前に検証されていると想定されます。付録 A.2.3 で指定されているgの生成方法は、この規格の以前のバージョンには含まれていなかったことに注意してください。したがって、この検証方法はそのような場合には適切ではありません。

domain_parameter_seed は、 pとqの生成からの出力です。付録 A.1.1.2 の方法を使用してpおよびqが生成された場合、 domain_parameter_seed が生成ルーチンから返され、検証側が利用できるようにしておく必要があります。 pとqのとき
付録 A.1.2.1 の方法を使用して生成された場合、 firstseed、 pseed、およびqseedの値が生成ルーチンから返され、使用可能になっている必要があります。 firstseed、 pseed、およびqseedを連結して、次のプロセスで使用されるdomain_parameter_seedを形成します。
Hash()をgの生成に使用されるハッシュ関数(つまり、 pおよびqの生成にも使用されるハッシュ関数) とします。

入力インデックスは、ジェネレーターg のインデックス番号です。詳細については、付録 A.2.3 を参照してください。

このメソッドのジェネレーターgを検証するには、次のプロセスまたは同等のプロセスを使用します。

入力：

1.p 、 q	素数。
2.domain_parameter_seed	pと q の生成に使用されるシード。
3.インデックス	付録 A.2.3 で x を生成するために使用されるインデックス。 Index は、符号なし整数を表す長さ 8 のビット文字列です。
4.g	検証されるgの値。

出力：

1.ステータス	生成ルーチンから返されるステータス。ステータスはVALIDまたはINVALIDのいずれかです。

プロセス：

注: count は符号なし 16 ビット整数です。

コメント:インデックスの有効な値が指定されていることを確認してください (上記を参照)。

1. (インデックスが正しくない) 場合は、 INVALID を返します。

2. $2 \leq g \leq (p-1)$ であることを確認します。 true でない場合は、 INVALID を返します。

3. $(g^q \neq 1 \bmod p)$ の場合は、 INVALID を返します。

4. N = len(q)。

5. e = (p – 1)/q。

6.カウント= 0。

7.カウント=カウント+ 1。

コメント:カウントが0 に戻っていないことを確認してください。

8. (カウント= 0) の場合は、 INVALID を返します。

コメント: 「ggen」はビット列 0x6767656E です。

9. U = ドメインパラメータシード || 「ゲン」 ||インデックス ||カウント。

10. W =ハッシュ(U)。

11. computed_g = pを修正します。

12. (computed_g < 2) の場合は、ステップ 7 に進みます。 コメント: ジェネレーターが見つからない場合。

13. (computed_g = g)の場合はVALIDを返し、そうでない場合は INVALID を返します。

付録 B: キーペアの生成

離散対数暗号 (DLC) は、有限体暗号 (FFC) と楕円曲線暗号 (ECC) に分けられます。 2 つの違いは、使用される数学の種類です。 DSA は FFC の一例です。 ECDSA は ECC の一例です。 DLC の他の例としては、FFC 形式と ECC 形式の両方を持つ Diffie-Hellman および MQV 鍵合意アルゴリズムがあります。

整数因数分解暗号 (IFC) の最も一般的な例は RSA です。

この付録では、FFC および ECC キー ペアと秘密番号の生成方法、および IFC キー ペアの生成方法を指定します。すべての生成方法では、承認され、適切にインスタンス化されたランダム ビット ジェネレーター (RBG) を使用する必要があります。 RBG は、生成されるキー ペアおよび秘密番号に関連付けられたセキュリティ強度以上のセキュリティ強度を持たなければなりません。セキュリティの強度とキーのサイズに関するガイダンスについては、SP 800-57 を参照してください。

この付録では、ビット文字列と整数の間で必要な変換については説明しません。この付録のプロセスで必要な場合、変換は付録 C.2 の指定に従って実行されます。

B.1 FFC キーペアの生成

FFC キー ペア(x、 y) は、ドメイン パラメーターのセット(p、 q、 g {、 domain_parameter_seed、 counter})に対して生成されます。 FFC 秘密鍵xと公開鍵y の生成には 2 つの方法が提供されています。これら 2 つの方法のいずれかを使用する必要があります。 DSA キー ペアを生成する前に、セクション 3.1 で指定されているように、ドメイン パラメータ(p、 q、およびg)の有効性が保証されているものとします。

DSA の場合、 LとNの有効な値はセクション 4.2 に記載されています。

B.1.1 追加のランダムビットを使用したキーペアの生成

この方法では、 xに必要なビットよりも 64 多いビットが RBG から要求されるため、ステップ 6 の mod 関数によって生成されるバイアスは無視できます。

次のプロセスまたはそれと同等のプロセスを使用して、FFC キー ペアを生成できます。

入力：

(p、 q、 g)	このプロセスに使用されるドメイン パラメーターのサブセット。 p、 q gは入力時に整数として指定されるか、使用前に整数に変換されます。

出力：

1.ステータス	キーペア生成プロセスから返されたステータス。ステータスはSUCCESSまたはERROR を示します。
2. (x, y)	生成された秘密鍵と公開鍵。生成プロセス中にエラーが発生した場合は、次の仕様のInvalid_xおよびInvalid_yで表されるように、 xおよびyの無効な値が返される必要があります。バツおよびyは整数として返されます。生成された秘密キーxの範囲は [1, q–1]、公開キーの範囲は [1, p–1] です。

プロセス：

1. N = len(q); L = len(p)。

   コメント: (L, N) ペアがセクション 4.2 で指定されていることを確認してください。

2. (L, N) ペアが無効な場合は、 ERRORインジケータ、 Invalid_x、および無効です。

3. requested_security_strength = (L, N) ペアに関連付けられたセキュリティ強度。

   SP 800-57 を参照してください。

4. requested_security_strength以上のセキュリティ強度を持つN+64のreturn_bitsの文字列をRBGから取得します。 ERROR表示が返された場合は、 ERROR表示、 Invalid_x、およびInvalid_yを返します。

5. returns_bits を(負でない) 整数cに変換します(付録 C.2.1 を参照)。

6. x = (c mod (q–1)) + 1。

   コメント: 0 ≤ c mod (q–1) ≤ q–2 であり、1 ≤ x ≤ q–1 を意味します。

7. y = gx mod p。

8. SUCCESS、 x、およびy を返します。

B.1.2 候補のテストによる鍵ペアの生成

この方法では、乱数が取得され、テストされて、正しい範囲のx値が生成されるかどうかが判断されます。 xが範囲外の場合、別の乱数が取得されます (つまり、 xの許容値が得られるまでプロセスが繰り返されます)。

次のプロセスまたはそれと同等のプロセスを使用して、FFC キー ペアを生成できます。

入力：

(p、 q、 g)	このプロセスに使用されるドメイン パラメーターのサブセット。 p、 q gは入力時に整数として指定されるか、使用前に整数に変換されます。

出力：

1.ステータス　キーペア生成プロセスから返されたステータス。ステータスはSUCCESSまたはERROR を示します。

2. (x, y)　生成された秘密鍵と公開鍵。生成プロセス中にエラーが発生した場合は、次の仕様のInvalid_xおよびInvalid_yで表されるように、 xおよびyの無効な値が返される必要があります。バツ
およびyは整数として返されます。生成された秘密キーxの範囲は [1, q−1]、公開キーの範囲は [1, p−1] です。

プロセス：

1. N = len(q); L = len(p)。

コメント: (L, N) ペアがセクション 4.2 で指定されていることを確認してください。

2. (L, N) ペアが無効な場合は、 ERROR表示、 Invalid_x、および
無効です。

3. requested_security_strength = (L, N) ペアに関連付けられたセキュリティ強度。
SP 800-57 を参照してください。

4.セキュリティ強度が1 のRBGからN returns_bitsの文字列を取得します。
requested_security_strength以上。 ERROR表示が返された場合は、 ERROR表示、 Invalid_x、およびInvalid_yを返します。

5. returns_bits を(負でない) 整数cに変換します(付録 C.2.1 を参照)。

6. (c > q−2) の場合は、ステップ 4 に進みます。

7. x = c + 1。

8. y = gx mod p。

9. SUCCESS、 x、およびy を返します。

B.2 FFC メッセージごとの秘密番号の生成

DSA では、署名されるメッセージごとに新しい乱数kを生成する必要があります。 kの生成には 2 つの方法が提供されています。これら 2 つの方法のいずれか、または別の承認された方法
という方法を使用するものとする。

Nの有効な値はセクション 4.2 に記載されています。 inverse(k, q)を、素数qを法とする乗算に関して(負でない) 整数kの逆数を計算する関数とする。逆数を計算する手法は、付録 C.1 に記載されています。

B.2.1 追加のランダムビットを使用したメッセージごとの秘密番号の生成

この方法では、 kに必要なビットよりも 64 多いビットが RBG から要求されるため、ステップ 6 の mod 関数によって生成されるバイアスは容易にはわかりません。

次のプロセスまたはそれと同等のプロセスを使用して、メッセージごとの秘密番号を生成できます。

入力：

(p、 q、 g) セクション 4.3.1 の指定に従って生成される DSA ドメイン パラメータ。

出力：

1.ステータス 秘密番号生成プロセスから返されたステータス。ステータスはSUCCESSまたはERROR を示します。

2. (k, k^{-1}) メッセージごとの秘密番号kとその mod q逆数k^{-1}。 エラーが発生した場合は、生成プロセス中に検出されました。k と k^{-1} の値が無効です
次の仕様のInvalid_kおよびInvalid_k_inverseで表されるように、返される必要があります。 kとkの範囲は [1, q–1] です。

プロセス：

1. N = len(q); L = len(p)。

   コメント: (L, N) ペアがセクション 4.2 で指定されていることを確認してください。

2. (L, N) ペアが無効な場合は、 ERROR表示、 Invalid_k、および
   無効な k_inverse。

3. requested_security_strength = (L, N) ペアに関連付けられたセキュリティ強度。
   SP 800-57 を参照してください。

4. requested_security_strength以上のセキュリティ強度を持つN+64のreturn_bitsの文字列をRBGから取得します。 ERROR表示が返された場合は、 ERROR表示、 Invalid_k、およびInvalid_k_inverse
   を返します。

5. returns_bits を(負でない) 整数cに変換します(付録 C.2.1 を参照)。

6. k = (c mod (q–1)) + 1。

7. (ステータス、 k^{-1}) =逆数(k, q)。

8.ステータス、 k、およびk^{-1}を返す。

B.2.2 テスト候補によるメッセージごとの秘密番号の生成

この方法では、乱数を取得してテストし、正しい範囲のkの値が生成されるかどうかを決定します。 kが範囲外の場合、別の乱数が取得されます (つまり、 kの許容値が得られるまでプロセスが繰り返されます)。

次のプロセスまたはそれと同等のプロセスを使用して、メッセージごとの秘密番号を生成できます。

入力：

(p、 q、 g) セクション 4.3.1 の指定に従って生成される DSA ドメイン パラメータ。

出力：

1.ステータス 秘密番号生成プロセスから返されたステータス。ステータスはSUCCESSまたはERROR を示します。

2. (k, k^{-1}) メッセージごとの秘密番号kとその逆数k^{-1}。 生成プロセス中にエラーが発生した場合は、次の仕様のInvalid_kおよびInvalid_k_inverseで表されるように、 kおよびkの無効な値が返される必要があります。 kとkの範囲は [1, q–1] です。

プロセス：

1. N = len(q); L = len(p)。

コメント: (L, N) ペアがセクション 4.2) で指定されていることを確認してください。

2. (L, N) ペアが無効な場合は、 ERROR表示、 Invalid_k、およびInvalid_k_inverseを返します。

3. requested_security_strength = (L, N) ペアに関連付けられたセキュリティ強度。
SP 800-57 を参照してください。

4.セキュリティ強度が1 のRBGからN returns_bitsの文字列を取得します。
requested_security_strength以上。 ERROR表示が返された場合は、 ERROR表示、 Invalid_k、およびInvalid_k_inverseを返します。

5. returns_bits を(負でない) 整数cに変換します(付録 C.2.1 を参照)。

6. (c > q–2) の場合は、ステップ 4 に進みます。

7. k = c + 1。

8. (ステータス, k^{-1}) =逆数(k, q)。

9.ステータス、 k、およびk^{-1}を返します。

B.3 IFC キーペアの生成

B.3.1 IFC キーペアの基準

IFC のキー ペアは、公開キー(n, e) と秘密キー(n, d) で構成されます。nは法であり、2 つの素数pとq の積です。 IFC のセキュリティは、これらの素数とプライベート指数 d の品質と機密性に依存します。素数pとqは次のように生成されます。

次のいずれかの方法:

A. pとqはどちらもランダムに生成された素数 (ランダム素数) です。ここで、 pとq
両方とも次のいずれかになります。

1. 証明可能な素数 (付録 B.3.2 を参照)、または
2. 確率素数 (付録 B.3.3 を参照)。

方法 1 と 2 を使用すると、 1024 ビットまたは 1536 ビットの長さのpとqを生成できます。 512 ビット長のp およびq は、これらの方法を使用して生成されないものとします。代わりに、 512 ビットの長さのpおよびq は、補助素数に基づく条件を使用して生成されます (付録 B.3.4、B.3.5、または B.3.6 を参照) 。

B. pとq は両方とも、次の追加条件を満たすランダムに生成された素数です (条件付き素数)。

- (p–1) は素因数p1を持ちます。
- (p+1) は素因数p2を持ちます。
- (q–1) は素因数q1を持ちます。
- (q+1) は素因数q2を持ちます。

ここで、p1、 p2、 q1 、 q2はpとqの補助素数と呼ばれます。

この方法を使用すると、次のいずれかのケースが適用されます。

1. 素数p1、 p2、 q1、 q2、 p 、 qはすべて証明可能な素数でなければなりません(付録を参照)
B.3.4)、
2. 素数p1、 p2、 q1 、 q2は証明可能な素数とし、素数pとq
確率素数でなければならない(付録 B.3.5 を参照)、または
3 素数p1、 p2、 q1、 q2、 pおよびqはすべて確率素数でなければなりません(付録を参照)
B.3.6)。

補助素数p1、 p2、 q1 、およびq2のそれぞれの最小長は、 nlenに依存します。ここで、 nlenはビット単位の法 nの長さです。 nlen はキー サイズとも呼ばれることに注意してください。補助素数の長さは、表 B.1 の制限に 従い、固定またはランダムに選択できます。最大長は、 nlen (各補助素数ペアの長さの合計)、および素数pとqが確率素数であるか証明可能素数であるかによって決まります (たとえば、補助素数ペアp1とp2 の場合、 len(p1) + len (p2) は、 p1とp2が確率素数または証明可能素数として生成されるかどうかに関係なく、nlenによって決定 される値未満でなければなりません) 3 。

[3] 確率素数pとq の場合: len(p1) + len(p2) < len(p) – log2(len(p)) – 6; len(q1) + len(q2)およびlen(q)についても同様です。証明可能な素数pとq の場合: len(p1) + len(p2) < len(p)/2 – log2(len(p)) – 7; len(q1) + len(q2)についても同様です

表B.1. p1、 p2、 q1 、 q2の最小長と最大長

ンレン	分。補助素数p1、p2、 q1 、 q2の長さ	最大。 len(p1) + len(p2)および len(q1) + len(q2)の長さ	
		p, q確率素数p, q証明可能素数	
1024	> 100ビット	496 ビット未満	< 239 ビット
2048年	> 140ビット	< 1007 ビット	494 ビット未満
3072	> 170ビット	1518 ビット未満	< 750 ビット

nlenの値が異なる場合(つまり、キー サイズが異なる場合)、 pとqの生成に許可される方法は表 B.2 に指定されています。

表B.2.許容される素数生成方法

ンレン	ランダム素数	条件付き素数
1024	いいえ	はい
2048年	はい	はい
3072	はい	はい

さらに、 FIPS 186-4 に準拠するには、すべての IFC キーが次の基準を満たしている必要があります。

1. 公開指数e は、次の制約に従って選択されます。

   (a) 公的検証指数eは素数を生成する前に選択されるものとする
   pとq、およびプライベート署名指数d。

   (b) 指数e は、次のような正の奇数でなければなりません。

   216 < e < 2256。

   eの値は、制約 1(b) を満たす任意の値、つまり、 e は固定値またはランダム値のいずれかであることに注意してください。

2. 素数pとq は、次の制約に従って選択されます。

   (a) (p–1) と(q–1)は、公開指数eに対して互いに素であるものとします。

   (b) プライベート素因数p はランダムに選択され、次の条件を満たすものとします。

およびlen(q)。いずれの場合も、 len(p) = len(q) = nlen/2 です。

(√2)(2(nlen / 2) – 1) ≤ p ≤ (2nlen / 2– 1)。ここで、 nlen は、必要な security_strength に適した長さです。

(c) プライベート素因数q はランダムに選択され、 (2)(2(nlen / 2) – 1を満たすものとします。
√) ≤ q ≤ (2nlen / 2– 1)。ここで、 nlen は、必要な security_strength に適した長さです。 (d) |p – q| > 2(nlen / 2)

– 100。

3. 私的署名指数d は、次の制約に従って選択されます。
pとqの生成:

(a) 指数d は、2nlen/ 2 < d < LCM(p–1, q–1) となるような正の整数値でなければなりません。

(b) d = e $^{-1}$ mod (LCM(p–1, q–1))。

つまり、(a) の不等式が成り立ち、1 ≡ (ed) (mod LCM(p–1, q–1)) となります。

非常にまれなイベントですが、 d ≤ 2nlen / 2 の場合は、 p、 q 、およびdの新しい値が決定されます。必須ではありませんが、別のe値を使用することもできます。

キー ペアの生成中に使用されるハッシュ関数はすべて承認される必要があります(つまり、FIPS 180 で指定されます)。

B.3.2 素数であることが証明されるランダム素数の生成

付録 B.3.1 の制約を満たす承認された方法は、素数であることが証明されているIFC ランダム素数pおよびqの生成に使用されます(ケース A.1 を参照)。そのような方法の 1 つが付録 B.3.2.1 および B.3.2.2 に記載されています。この方法では、最初にランダム シードが必要です (付録 B.3.2.1 を参照)。シードの長さは、係数n に関連付けられたセキュリティ強度の 2 倍に等しくなります。シードを取得した後、プライムを生成できます (付録 B.3.2.2 を参照)。

B.3.2.1 シードを取得する

このメソッドのシードを生成するには、次のプロセスまたは同等のプロセスを使用します。

入力：

ンレン　法nの意図されたビット長。

出力：

状態　返されるステータス。ステータスはSUCCESSまたはFAILURE のいずれかです。

シード　種。ステータス= FAILUREの場合、値 0 がシードとして返されます。

プロセス：

1. nlenが有効でない場合 (セクション 5.1 を参照)、リターン(FAILURE, 0)。

2. SP で指定されているように、 security_strengthをnlenに関連付けられたセキュリティ強度とします。
800-57、パート 1。

3. security_strengthをサポートするRBGから(2 * security_strength)ビットの文字列シードを取得します。

4. 戻ります(成功、シード)。

B.3.2.2 証明可能な素数pおよびqの構築

次のプロセスまたは同等のプロセスを使用して、素数であることが証明されているランダム素数pおよびq (RSA 法 n の因数として使用される)を構築します。

入力：

ンレン	法nの意図されたビット長。
e	公的検証指数。
シード	付録 B.3.2.1 の方法を使用して取得したシード。

出力：

状態	生成プロセスのステータス。ステータスはSUCCESSまたはFAILUREのいずれかです。 FAILUREが返された場合、他のパラメータとしてゼロ値が返されます。

pおよびq nのプライベート素因数。

プロセス：

1. nlenが 2048 でも 3072 でもない場合は、 (FAILURE, 0, 0)を返します。

2. ((e ≤ 216) OR (e ≥ 2256) OR (eが奇数ではない)) の場合、 (FAILURE, 0, 0) を返します。

3. SP で指定されているnlenの値に従ってsecurity_strengthの値を設定します。
800-57、パート 1。

4. (len(seed) ≠ 2 * security_strength) の場合、 (FAILURE, 0, 0)を返します。

5. working_seed =シード。

6. p を生成します。

6.1 L = nlen/2、 N1 = 1、 N2 = 1、 first_seed = working_seedおよびe を使用し、付録 C.10 の証明可能な素数構成法を使用してpおよびpseed を取得します。失敗した場合
が返される場合は、 (FAILURE, 0, 0) を返します。

6.2 working_seed = pseed。

7. q を生成します。

7.1 L = nlen/2、 N1 = 1、 N2 = 1、 first_seed = working_seedおよびe を使用し、付録 C.10 の証明可能な素数構成法を使用してqおよびqseed を取得します。失敗した場合
が返される場合は、 (FAILURE, 0, 0) を返します。

7.2 working_seed = qseed。

8. (|p – q| ≤ 2nlen/2 – 100) の場合、ステップ 7 に進みます。

9. 内部生成されたシードをゼロ化します。

9.1 pseed = 0;

9.2 qseed = 0;

9.3 working_seed = 0。

10. (成功、 p、 q) を返します。

B.3.3 おそらく素数であるランダム素数の生成

付録 B.3.1 の制約を満たす承認された方法は、おそらく素数であるIFC ランダム素数pおよびqの生成に使用されます(ケース A.2 を参照)。

次のプロセスまたは同等のプロセスを使用して、ランダムな確率素数pを構築します。
およびq (RSA 法n の係数として使用されます):

入力：

ンレン	法nの意図されたビット長。
e	公的検証指数。

出力：

状態	生成プロセスのステータス。ステータスはSUCCESSまたはFAILUREのいずれかです。

pおよびq nのプライベート素因数。 FAILUREが返された場合、ゼロ値が返されます。
pとqとして返されます。

プロセス：

1. nlenが 2048 でも 3072 でもない場合は、 (FAILURE, 0, 0) を返します。

2. ((e ≤ 216) OR (e ≥ 2256) OR (eが奇数ではない)) の場合、 (FAILURE, 0, 0) を返します。

3. SP で指定されているnlenの値に従ってsecurity_strengthの値を設定します。
800-57、パート 1。

4. p を生成します。

4.1 i = 0。

4.2 security_strengthをサポートするRBGから(nlen/2)ビットの文字列pを取得します。

4.3 (pが奇数でない場合)、 p = p + 1。

4.4 If ((p < (2)(2(nlen / 2) – 1))、ステップ 4.2 に進みます。

4.5 (GCD(p－1, e) = 1) の場合、

4.5.1適切な関数を使用して、付録 C.3 に指定されているとおりにpの素数性をテストします。
付録 C.3 の表 C-2 または C-3 の値を反復回数として使用します。

4.5.2 pがおそらく素数の場合は、ステップ 5 に進みます。

4.6 i= i + 1。

4.7 (i ≥ 5(nlen/2)) の場合、 (FAILURE, 0, 0)を返します。

それ以外の場合は、ステップ 4.2 に進みます。

5. q を生成します。

5.1 i = 0。

5.2 security_strengthをサポートするRBGから(nlen/2)ビットの文字列qを取得します。

5.3 (qが奇数でない場合)、 q = q + 1。

5.4 (|p – q| ≤ 2nlen/2 – 100) の場合、ステップ 5.2 に進みます。

5.5 If ((q < (2)(2(nlen / 2) – 1))、ステップ 5.2 に進みます。

5.6 (GCD(q－1, e) = 1)の場合

5.6.1適切な関数を使用して、付録 C.3 に指定されているようにqの素数性をテストします。
付録 C.3 の表 C-2 または C-3 の値を反復回数として使用します。

5.6.2 qがおそらく素数の場合、 (SUCCESS, p, q)を返します。

5.7 i = i + 1。

5.8 (i ≥ 5(nlen/2)) の場合、 (FAILURE, 0, 0)を返します。

それ以外の場合は、ステップ 5.2 に進みます。

B.3.4 補助証明可能素数に基づく条件付き証明可能素数の生成

このセクションでは、付録 B.3.1、ケース B.1 で指定された追加条件を使用してIFC 素数pおよびqを生成するための承認された方法を指定します。ここで、 p、 p1、 p2、 q、 q1 、およびq2はすべて証明可能な素数です。この方法では、最初にランダム シードが必要です (付録 B.3.2.1 を参照)。シードの長さは、係数n に関連付けられたセキュリティ強度の 2 倍に等しくなります。最初のシードが取得された後、プライムを生成できます。

表 B.1 に従って、 bitlen1 、 bitlen2、 bitlen3、 bitlen4をそれぞれp1、 p2、 q1 、 q2のビット長とします。証明可能な素数を生成するには、次のプロセスまたは同等のプロセスを使用します。

入力：

ンレン　　法nの意図されたビット長。

e　　公的検証指数。

シード　　付録 B.3.2.1 の方法を使用して取得したシード。

出力：

状態　　生成プロセスのステータス。ステータスはSUCCESSまたはFAILUREのいずれかです。
FAILUREが返された場合は、 pとqの値としてゼロが返されます。

pおよびq nのプライベート素因数。

プロセス：

1. nlenが 1024、2048、3072 のいずれでもない場合は、 (FAILURE, 0, 0) を返します。
2. ((e ≤ 216) OR (e ≥ 2256) OR (eが奇数ではない)) の場合、 (FAILURE, 0, 0) を返します。
3. SP で指定されているnlenの値に従ってsecurity_strengthの値を設定します。
   800-57、パート 1。
4. (len(seed) ≠ 2 * security_strength) の場合、 (FAILURE, 0, 0)を返します。
5. working_seed =シード。
6. p を生成します。
   6.1 L = nlen/2、 N1 = bitlen1、 N2 = bitlen2、 firstseed = working_seedおよびe を使用して、付録 C.10 の証明可能な素数構成法を使用して、 p、 p1、 p2 、およびpseed を取得します。
   FAILUREが返された場合は、 (FAILURE, 0, 0) を返します。
   6.2 working_seed = pseed。
7. q を生成します。
   7.1 L = nlen/2、 N1 = bitlen3、 N2 = bitlen4 、 firstseed = working_seedおよびe を使用し、付録 C.10 の証明可能な素数構成法を使用してq、 q1、 q2 、およびqseed を取得します。 FAILUREが返された場合は、 (FAILURE, 0, 0) を返します。
   7.2 working_seed = qseed。
8. (|p – q| ≤ 2nlen/2 – 100) の場合、ステップ 7 に進みます。
9. 内部生成されたシードをゼロ化します。
   9.1 pseed = 0。

9.2 qseed = 0。

9.3 working_seed = 0。

10. (成功、 p、 q) を返します。

B.3.5 補助証明可能素数に基づく条件付きの可能素数の生成

このセクションでは、付録 B.3.1、ケース B.2 で指定された追加条件を使用してIFC 素数pおよびqを生成するための承認された方法を指定します。ここで、 p1、 p2、 q1 、およびq2はおそらく素数であり、 pとqはおそらく素数です。プライム。この方法では、最初にランダム シードが必要です (付録 B.3.2.1 を参照)。シードの長さは、係数n に関連付けられたセキュリティ強度の 2 倍に等しくなります。最初のシードが取得された後、プライムを生成できます。

表 B.1 に従って、 bitlen1 、 bitlen2、 bitlen3、 bitlen4をそれぞれp1、 p2、 q1 、 q2のビット長とします。 p を構築するには、次のプロセスまたは同等のプロセスを使用します。
そしてq.

入力：

ンレン	法nの意図されたビット長。
e	公的検証指数。
シード	付録 B.3.2.1 の方法を使用して取得したシード。

出力：

状態	生成プロセスのステータス。ステータスはSUCCESSまたはFAILUREのいずれかです。 FAILUREが返された場合は、 pとqの値としてゼロが返されます。

pおよびq nのプライベート素因数。

プロセス：

1. nlenが 1024、2048、3072 のいずれでもない場合は、 (FAILURE, 0, 0) を返します。
2. ((e ≤ 216) OR (e ≥ 2256) OR (eが奇数ではない)) の場合、 (FAILURE, 0, 0) を返します。
3. SP 800-57、パート 1 で指定されているように、nlenの値に従ってsecurity_strengthの値を設定します。
4. (len(seed) ≠ 2 * security_strength) の場合、 (FAILURE, 0, 0)を返します。

コメント: 4 つの素数p1、 p2、 q1 、 q2を生成します。
それはおそらくプライムです。

5. p を生成します。

5.1 bitlen1 を長さとして使用し、 seed をinput_seedとして使用し、ランダム素数を使用します。
p1とprime_seedを取得するための生成ルーチンは、付録 C.6 にあります。失敗した場合
が返される場合は、戻り値(FAILURE, 0, 0) が返されます。

5.2 bitlen2を長さとして、 prime_seed をinput_seed として使用し、付録 C.6 のランダム素数生成ルーチン
を使用してp2とprime_seedの新しい値を取得します。 FAILUREが返された場合は、 (FAILURE,
0, 0)が返されます。

5.3付録 C.9 のルーチンを使用して、 p1、 p2、 nlen、e 、 security_strengthを入力として素数pを生成
し、 Xpも取得します。 FAILUREが返された場合は、 (FAILURE, 0, 0) を返します。

6. q を生成します。

6.1.長さとしてbitlen3を、input_seedとしてprime_seed を使用し、付録 C.6 のランダム素数生成ルーチンを使
用してq1と prime_seed の新しい値を取得します。 FAILUREが返された場合は、 (FAILURE, 0,
0)が返されます。

6.2 bitlen4を長さとして、 prime_seed をinput_seed として使用し、付録 C.6 のランダム素数生成ルーチン
を使用してq2とprime_seedの新しい値を取得します。 FAILUREが返された場合は、 (FAILURE,
0, 0)が返されます。

6.3 q1、 q2、 nlen、e 、 security_strengthを入力として付録 C.9 のルーチンを使用して素数qを生成し、
Xqも取得します。 FAILUREが返された場合は、 (FAILURE, 0, 0) を返します。

7. ((|p – q| ≤ 2nlen/2 – 100)または(|Xp – Xq| ≤ 2nlen/2 – 100)) の場合は、ステップ 6 に進みます。

8. 返されない内部生成されたものをゼロ化します。

8.1 XP = 0。

8.2 Xq = 0。

8.3プライムシード= 0。

8.4 p1 = 0。

8.5 p2 = 0。

8.6 q1 = 0。

8.7 q2 = 0。

9. (成功、 p、 q) を返します。

B.3.6 補助確率素数に基づく条件による確率素数の生成

付録 B.3.1 の制約を満たす承認された方法は、おそらく素数であり、付録 B.3.1 の追加制約を満たすIFC 素数pおよびqの生成に使用されます(ケース B.3 を参照)。この場合、素因数p1、 p2、 q1 、 q2もおそらく素数です。

4 つの乱数Xp1、 Xp2、 Xq1 、 Xq2が生成され、そこから素因数p1、 p2、 q1 、 q2が決定されます。次に、 p1とp2、および追加の乱数Xp を使用して p を決定し、 q1とq2および乱数Xqを使用してqを取得します。 bitlen1、bitlen2、 bitlen3、 bitlen4をそれぞれ、表 B.1 に従って選択されたp1、 p2、 q1 、 q2のビット長とする。

pとq を生成するには、次のプロセスまたは同等のプロセスを使用します。

入力：

ンレン	法nの意図されたビット長。
e	公的検証指数。

出力：

状態	生成プロセスのステータス。ステータスはSUCCESSまたはFAILUREのいずれかです。FAILUREが返された場合は、 pとqの値としてゼロが返されます。
pとq	nのプライベート素因数。

プロセス：

1. nlenが 1024、2048、3072 のいずれでもない場合は、 (FAILURE, 0, 0) を返します。
2. ((e ≤ 216) OR (e ≥ 2256) OR (eが奇数ではない)) の場合、 (FAILURE, 0, 0) を返します。
3. SP 800-57、パート 1 で指定されているように、nlenの値に従ってsecurity_strengthの値を設定します。
4. p を生成します。
   - 4.1 長さbitlen1ビットの奇数整数Xp1と 2 番目の奇数整数Xp2を生成する
     security_strength をサポートする承認済みの乱数生成器を使用して、長さbitlen2ビットの乱数を生成します。
   - 4.2 Xp1から最初の整数まで、連続する奇数整数を順番に検索します。
     推定素数p1が見つかります。候補となる整数は、付録 C.3 に指定されているとおり、素数についてテストされます。 Xp2から始めて、このプロセスを繰り返してp2を見つけます。確率素数p1とp2 は、素数性テストに合格する最初の整数となります。

4.3付録 C.9 のルーチンを使用して、 p1、 p2、 nlen、 eおよびsecurity_Strengthを入力として素数pを生成し、 Xpも取得します。 FAILUREが返された場合は、 (FAILURE, 0, 0) を返します。

5. q を生成します。

5.1 長さbitlen3ビットの奇数整数Xq1と 2 番目の奇数整数Xq2を生成する security_strength をサポートする承認済みの乱数生成器を使用して、長さbitlen4ビットの乱数を生成します。

5.2 Xq1から最初の整数まで、連続する奇数整数を順番に検索します。推定素数q1が見つかります。候補となる整数は、付録 C.3 に指定されているとおり、素数かどうかテストされます。 Xq2から始めて、このプロセスを繰り返してq2を見つけます。確率素数q1とq2 は、素数性テストに合格する最初の整数となります。

5.3付録 C.9 のルーチンを使用し、 q1、 q2、 nlen、eおよびsecurity_Strengthを入力して素数qを生成し、 Xqも取得します。 FAILUREが返された場合は、 (FAILURE, 0, 0) を返します。

6. ((|Xp – Xq| ≤ 2nlen/2 – 100)または (|p – q| ≤ 2nlen/2 – 100)の場合))、ステップ 5 に進みます。

7. 返されない内部生成値をゼロ化します。

7.1 XP = 0。

7.2 Xq = 0。

7.3 XP1 = 0。

7.4 XP2 = 0。

7.5 Xq1 = 0。

7.6 Xq2 = 0。

7.7 p1 = 0。

7.8 p2 = 0。

7.9 q1 = 0。

7.10 q2 = 0。

8. (成功、 p、 q) を返します。

B.4 ECC キーペアの生成

ECC キーのペアdおよびQは、ドメイン パラメーターのセット(q、FR、a、b {、domain_parameter_seed}、 G、n、 h) に対して生成されます。 ECC 秘密鍵dと公開鍵Q の生成には 2 つの方法が提供されています。これら 2 つの方法のいずれかを使用してdとQを生成します。

ECDSA 鍵ペアを生成する前に、セクション 3.1 で指定されているように、ドメイン パラメータ(q、FR、a、b {、domain_parameter_seed}、 G、n、 h)の有効性の保証が得られていなければなりません。

ECDSA の場合、 nの有効なビット長はセクション 6.1.1 に記載されています。楕円曲線の計算と変換ルーチンの定義については、ANS X9.62 を参照してください。

B.4.1 追加のランダムビットを使用したキーペアの生成

この方法では、ステップ 6 の mod 関数によって生成されるバイアスが無視できるように、 dに必要なビットよりも 64 多いビットが RBG から要求されます。

ECC キー ペアを生成するには、次のプロセスまたは同等のプロセスを使用できます。

入力：

1. (q, FR, a, b {,ドメインパラメータシード}, G, n, h)

   このプロセスに使用されるドメイン パラメーター。 nは素数であり、Gは楕円曲線上の点です。

出力：

1.ステータス　キーペア生成プロシージャから返されたステータス。ステータスはSUCCESSまたはERROR を示します。

2. (d, Q) 生成された秘密鍵と公開鍵。生成プロセス中にエラーが発生した場合は、次の仕様のInvalid_dおよびInvalid_Qで表されるように、 dおよびQの無効な値が返される必要があります。 dは整数、 Qは楕円曲線点です。生成された秘密鍵dの範囲は [1, n–1] です。

プロセス：

1. N = len(n)。

   コメント: N がセクション 6.1.1 の表 1 に含まれていることを確認してください。

2. Nが無効な場合は、 ERROR表示、 Invalid_d、およびInvalid_Qを返します。

3. requested_security_strength = Nに関連付けられたセキュリティ強度。 SP 800-57 を参照。パート1。

4. requested_security_strength以上のセキュリティ強度を持つN+64のreturn_bitsの文字列をRBGから取得します。 ERROR表示が返された場合は、 ERROR表示、 Invalid_d、およびInvalid_Qを返します。

5. returns_bits を(負でない) 整数cに変換します(付録 C.2.1 を参照)。

6. d = (c mod (n–1)) + 1。

7. Q = dG。

8. SUCCESS、 d、およびQを返します。

B.4.2 候補のテストによる鍵ペアの生成

この方法では、乱数が取得され、テストされて、正しい範囲のdの値が生成されるかどうかが判断されます。 dが範囲外の場合、別の乱数が取得されます (つまり、 dの許容値が得られるまでプロセスが繰り返されます)。

ECC キー ペアを生成するには、次のプロセスまたは同等のプロセスを使用できます。

入力：

1. (q, FR, a, b {,ドメインパラメータシード}, G, n, h)

このプロセスに使用されるドメイン パラメーター。 nは素数、 Gは楕円曲線上の点です。

出力：

1.ステータス　キーペア生成プロシージャから返されたステータス。ステータスはSUCCESSまたはERROR を示します。

2. (d, Q) 生成された秘密鍵と公開鍵。生成プロセス中にエラーが発生した場合は、次の仕様のInvalid_dおよびInvalid_Qで表されるように、 dおよびQの無効な値が返される必要があります。 dは整数、 Qは楕円曲線点です。生成された秘密鍵dの範囲は [1, n–1] です。

プロセス：

1. N = len(n)。

コメント: N がセクション 6.1.1 の表 1 に含まれていることを確認してください。

2. Nが無効な場合は、 ERROR表示、 Invalid_d、およびInvalid_Qを返します。

3. requested_security_strength = Nに関連付けられたセキュリティ強度。 SP 800-57 を参照。パート1。

4.セキュリティ強度が1 のRBGからN returns_bitsの文字列を取得します。requested_security_strength以上。 ERROR表示が返された場合は、 ERROR表示、 Invalid_d、およびInvalid_Qを返します。

5. returns_bits を(負でない) 整数cに変換します(付録 C.2.1 を参照)。

6. (c > n–2) の場合は、ステップ 4 に進みます。

7. d = c + 1。

8. Q = dG。

9. SUCCESS、 d、およびQを返します。

B.5 メッセージごとの ECC 秘密番号の生成

ECDSA では、署名されるメッセージごとに新しい乱数kを生成する必要があります。 kの生成には 2 つの方法が提供されています。これら 2 つの方法のいずれか、または別の承認された方法
という方法を使用するものとする。

nの有効な値はセクション 6.1.1 に記載されています。楕円曲線の計算と変換ルーチンの定義については、ANS X9.62 を参照してください。

inverse(k, n)を、素数nを法とする乗算に関して(負でない) 整数kの逆数を計算する関数とする。逆数を計算する手法は、付録 C.1 に記載されています。

B.5.1 追加のランダムビットを使用したメッセージごとの秘密番号の生成

この方法では、 kに必要なビットよりも 64 多いビットが RBG から要求されるため、ステップ 6 の mod 関数によって生成されるバイアスは容易にはわかりません。

次のプロセスまたはそれと同等のプロセスを使用して、メッセージごとの秘密番号を生成できます。

入力：

1. (q, FR, a, b {,ドメインパラメータシード}, G, n, h)

   このプロセスに使用されるドメイン パラメーター。 nは素数、 Gは楕円曲線上の点です。

出力：

1.ステータス　キーペア生成プロシージャから返されたステータス。ステータスはSUCCESSまたはERROR を示します。

2. (k, k^{-1}) 生成された秘密数kとその逆数k $^{-1}$。エラーが発生した場合は、
   生成プロセス中に発生した場合は、 Invalid_kおよびInvalid_k_inverseで表されるように、 kおよびkの無効な値が返される必要があります。
   以下の仕様で。 kとk は[1, n–1] の範囲の整数です。

プロセス：

1. N = len(n)。

   コメント: N がセクション 6.1.1 の表 1 に含まれていることを確認してください。

2. Nが無効な場合は、 ERROR表示、 Invalid_k、およびInvalid_k_inverseを返します。

3. requested_security_strength = Nに関連付けられたセキュリティ強度。 SP 800-57 を参照。
パート1。

4. requested_security_strength以上のセキュリティ強度を持つN+64のreturn_bitsの文字列をRBGか
ら取得します。 ERROR表示が返された場合は、 ERROR表示、 Invalid_k、およびInvalid_k_inverse
を返します。

5. returns_bits を負でない整数cに変換します(付録 C.2.1 を参照)。

6. k = (c mod (n–1)) + 1。

7. (ステータス, k^{-1})=逆数(k, n)。

8.ステータス、 k、およびk^{-1}を返す。

B.5.2 テスト候補によるメッセージごとの秘密番号の生成

この方法では、乱数を取得してテストし、正しい範囲のkの値が生成されるかどうかを決定します。 kが範囲外の場合、別の乱数が取得されます (つまり、 kの許容値が得られるまでプロセスが繰り返されます)。

次のプロセスまたはそれと同等のプロセスを使用して、メッセージごとの秘密番号を生成できます。

入力：

1. (q, FR, a, b {,ドメインパラメータシード}, G, n, h)

このプロセスに使用されるドメイン パラメーター。 nは素数、 Gは楕円曲線上の点です。

出力：

1.ステータス　キーペア生成プロシージャから返されたステータス。ステータスはSUCCESSまたはERROR を
示します。

2. (k, k^{-1}) 生成された秘密数kとその逆数k^{-1}。エラーが発生した場合は、
生成プロセス中に検出されました。k と k^{-1} の値が無効です
次の仕様のInvalid_kおよびInvalid_k_inverseで表されるように、返される必要があります。 kとk^{-1} は[1, n–1] の範囲の整数です。

プロセス：

1. N = len(n)。

コメント: N がセクション 6.1.1 の表 1 に含まれていることを確認してください。

2. Nが表 1 に含まれていない場合は、エラー表示、 Invalid_k、およびInvalid_k_inverseを返します。

3. requested_security_strength = Nに関連付けられたセキュリティ強度。 SP 800-57 を参照。
パート1。

4.セキュリティ強度が1 のRBGからN returns_bitsの文字列を取得します。
requested_security_strength以上。 ERROR表示が返された場合は、 ERROR表示、
Invalid_k、およびInvalid_k_inverseを返します。

5. returns_bits を(負でない) 整数cに変換します(付録 C.2.1 を参照)。

6. (c > n–2) の場合は、ステップ 4 に進みます。

7. k = c + 1。

8. (ステータス、 k^{-1}) =逆数(k, n)。

9.ステータス、 k、およびkを返します。

付録 C: 他の数量の生成

この付録には、この規格の実装に必要な補足プロセスのルーチンが含まれています。付録C.1 は、メッセージごとの秘密kの逆行列 (セクション 4.5、付録 B.2.1、B.2.2、B.5.1、および B.5.2 を参照) と署名部分sの逆行列を生成するために必要です。署名検証中に使用されます (セクション 4.7 を参照)。付録 C.2 のルーチンは、この標準の実装で必要な場合にビット文字列と整数の間で変換する必要があります。付録 C.3 には、DSA ドメイン パラメータと RSA キー ペアの生成中に使用される確率的素数テストが含まれています。付録 C.4 および C.5 には、付録 C.3.3 のルーカスの確率的素数性テスト中に完全二乗をチェックし、ヤコビ記号を計算するために必要なアルゴリズムが含まれています。付録 C.6 には、素数を構築するための Shawe-Taylor アルゴリズムが含まれています。付録 C.7 では、付録 C.6 のランダム素数生成ルーチンで必要とされる、試行除算を実行するプロセスを提供します。付録 C.8 のふるい手順は、付録 C.7 の試行分割ルーチンで必要となります。付録 C.7 の試行分割プロセスと付録 C.8 のふるい手順は、ANS X9.80、素数生成、素数検査、および素数証明書から抽出されたものです。 RSA キー ペアの生成中に付録 C.9 が必要です。付録 C.10 は、RSA の証明可能なプライムを構築する方法を提供します (付録 B.3.2.2 および B.3.4 を参照)。

C.1 逆数値の計算

このアルゴリズムまたは同等の結果を生成するアルゴリズムは、乗法逆z–1 mod a を計算するために使用されます。ここで、0 < z < a、0 < z–1 < a、およびaは素数です。この標準では、 zはkまたはs のいずれかであり、 aはqまたはn のいずれかです。

入力：

1.z _	mod aを反転する値(つまり、 kまたは s)。
2 .	ドメインパラメータと（プライム)モジュラス（つまり、 qまたはn のいずれか）。

出力：

1.ステータス	この関数から返されるステータス。ステータスはSUCCESSまたはERRORです。
1 2.z _	z mod aの逆乗 (存在する場合)。

プロセス：

1. aとzが正の整数であり、 z < a であることを確認します。そうでない場合は、エラーを返します表示。

2. i = a、 j = z、 y2 = 0、およびy1 = 1 を設定します。

3.商 = i/j 。

4.剰余 = i –(j *商)。

5. y = y2 –(y1 *商)。

6. i = j、 j = 剰余、 y2 = y1、およびy1 = y を設定します。

7. (j > 0) の場合は、ステップ 3 に進みます。

8. (i ≠ 1) の場合は、エラー表示を返します。

9. SUCCESSとy2 mod a を返します。

C.2 ビット文字列と整数間の変換

C.2.1 ビット文字列から整数への変換

n個の長さのビット列 { x1, …, xn } は、規則に従って整数に変換されます。

$$\{ x1, \cdots, xn \} \rightarrow (x1 \quad 2^{n-1}) + (x2 \quad 2^{n-2}) + \cdots + (n1 \quad 2) + xn \text{ 。}$$

シーケンスの最初のビットは対応する整数の最上位ビットに対応し、最後のビットは最下位ビットに対応することに注意してください。

入力：

1. b1、 b2、 … 、 bn変換されるビット列。

出力：

1.C _ 要求されたビット文字列の整数表現。

プロセス：

1. (b1, b2,… , bn)をbの左端から右端までのビットとします。

2. $C = \sum_{i=1}^{n} (\text{に}) 2b\_$ 。

3. Cを返します。

この規格では、整数Cの 2 進数の長さは、 C < 2nを満たす最小の整数nとして定義されます。

。

C.2.2 整数からビット文字列への変換

0 ≤ x < 2nの範囲の整数x は、次に示すようにバイナリ展開を使用して、n 個の長さのビット シーケンスに変換できます。

$$x = (x1 \quad 2n-1) + (x2 \quad 2n-2) + \cdots + (xn-1 \quad 2) + xn \rightarrow \{x1, \cdots, xn\}$$

シーケンスの最初のビットは対応する整数の最上位ビットに対応し、最後のビットは最下位ビットに対応することに注意してください。

入力：

1.C _ 変換される非負の整数。

出力：

1. b1、 b2、 …、 bn整数Cのビット文字列表現。

プロセス：

1. (b1, b2, …, bn)がビット列を表すものとします。ここで、 bi = 0 または 1、 b1は最上位ビット、 bnは最下位ビットです。
2. C < 2nを満たす任意の整数nについて 、ビットbi は以下を満たします。

$$C = \sum_{k=1}^{n} 2^{(に)} b_k$$

3. b1、 b2、 …、 bn を返します。

この規格では、整数Cの 2 進数の長さは、 C < 2nを満たす最小の整数nとして定義されます。

。

C.3 確率的素数検定

素数の生成と検証中に、確率的素数性テストが必要になる場合があります。承認された堅牢な確率的素数性テストを選択して使用するものとします。

利用可能な確率アルゴリズムがいくつかあります。付録 C.3.1 および C.3.2 で説明されている Miller-Rabin の確率的素数検定は、MO Rabin による手順のバージョンであり、部分的に Gary L. Miller のアイデアに基づいています。これらのバージョンの 1 つは、以下で説明する Miller-Rabin テストとして使用されます。詳細については、[4]を参照してください。これらのテストでは、 RBG を承認済みのランダム ビット ジェネレーターとします。

利用可能なルーカスの確率的素数性テストがいくつかあります。 [5] で提供されるバージョンは付録 C.3.3 で指定されます。

この標準では、素数性をテストするための 2 つの代替方法が許可されています。1 つはミラー•ラビン テストのみを複数回反復する方法、または反復されたミラー•ラビン テストとその後に 1 回のルーカス テストを使用する方法です。反復の値(付録 C.3.1 および C.3.2 で使用されるもの) は、使用されるアルゴリズム、セキュリティ強度、使用されるエラー確率、素数候補の長さ (ビット単位)、および

実行するテストの種類。表 C.1、C.2、および C.3 に最小反復回数を示します。
実行されるミラーラビンテストの内容。

付録 F に記載されているように、表 C.1、C.2、および C.3 のpおよびqのミラーラビン検定の数の値を導き出した誤り確率の定義が十分に保守的でない場合、規定の多数の Miller-Rabin テストの後に 1 つの Lucas テストを続けることができます。 2 つのテストの組み合わせ (つまり、ランダムに選択された塩基を使用した指定されたラウンド数の Miller-Rabin テストとそれに続く 1 ラウンドの Lucas テスト) に合格する既知の非プライム値はないため、2 つのテストの組み合わせは次の結果を提供する可能性があります。 Miller-Rabin テストのみを使用する場合に比べて、素数性がさらに保証されます。 DSA の場合、2 つのテストを組み合わせた方がパフォーマンスが向上する可能性があります。ただし、 RSA 素数を生成するときにp1、 p2、 q1 、およびq2値の素数性をテストする場合、Lucas テストは必要ありません。詳細については、付録 F を参照してください。

表C.1. DSA の Miller-Rabin 反復の最小数

パラメーター	MR検査のみ	MR テストとそれに続く One Lucas テスト
p :1024ビット q :160ビット エラー確率 = 2 − 80	pとqの場合: 40	pの場合 :3 qの場合: 19
p :2048ビット q :224ビット エラー確率 = 2 − 112	pとqの場合: 56	pの場合 :3 qの場合: 24
p :2048ビット q :256ビット エラー確率 = 2 − 112	pとqの場合: 56	pの場合 :3 qの場合: 27
p :3072ビット q :256ビット エラー確率 = 2 − 128	pとqの場合: 64	pの場合 :2 qの場合: 27

表C.2. RSA デジタル署名で使用する素数を生成する場合の MR テストの最小ラウンド数

パラメーター	MR検査のみ
1 p 、 2 p 、 q1および2 q > 100 ビット pおよびq: 512ビット エラー確率 = 2 − 80	1pにつき、 2p 、 q1、2q ： 28 _ _ _ pとqの場合: 5
1 p 、 2 p 、 q1および2 q > 140 ビット pとq: 1024ビット エラー確率 = 2 − 112	のために 1 p 、 p2 、 q1および2 q : 38 pとqの場合: 5
1 p 、 p2 、 q1および2 q > 170 ビット pおよびq: 1536ビット エラー確率 = 2 −128	1 p 、 p2 、 q1 、および2 qの場合: 41 pとqの場合: 4

表C.3.誤り確率2 〜 100を使用して RSA デジタル署名で使用する素数を生成する場合の MR テストの最小ラウンド数

パラメーター	MR検査のみ
p1 、 p2 、 q1およびq2 > 100 ビット pとq:512	p1 、 p2 、 q1およびq2の場合: 38 pとqの場合: 7
p1 、 p2 、 q1およびq2 > 140 ビット pとq: 1024ビット	1p 、 2p 、 q1の場合_ そして2q ： 32 pとqの場合: 4
p1 、 p2 、 q1およびq2 > 170 ビット pおよびq: 1536ビット	p1 、 p2 、 q1およびq2の場合: 27 pとqの場合: 3

C.3.1 ミラー•ラビンの確率的素数検定

RBG を承認済みのランダム ビット ジェネレーターとします。

入力：

1.w _ 素数性をテストする奇数の整数。これは、 pまたはq、あるいは補助素数p1、 p2、 q1 、またはq2のいずれかになります。

2.反復　実行されるテストの反復回数。値は表 C.1、C.2、または C.3 と一致していなければなりません。

出力：

1.ステータス　検証手順から返されたステータス。ステータスは、 PROBABLY PRIMEまたはCOMPOSITEのいずれかです。

プロセス：

1. aを、2a がw－1を除算するような最大の整数であるとします。

2. m = (w－1) / 2a。

3. wlen = len (w)。

4. i = 1の場合、反復は次のようになります。

4.1 RBG からwlenビットの文字列bを取得します。

コメント: 1 < b < w－1 であることを確認してください。

4.2 ((b ≤ 1) または(b ≥ w－1)) の場合、ステップ 4.1 に進みます。

4.3z _ = bm mod w。

4.4 ((z = 1) または(z = w － 1)) の場合、ステップ 4.7 に進みます。

4.5 j = 1の場合、 a － 1 を実行します。

4.5.1z _ = 2z _ モッドw。

4.5.2 (z = w－1) の場合、ステップ 4.7 に進みます。

4.5.3 (z = 1) の場合、ステップ 4.6 に進みます。

4.6コンポジットを返します。

4.7 続行します。　コメント:ステップ 4 の do ループのiをインクリメントします。

5.おそらくプライムを返します。

C.3.2 強化された Miller-Rabin 確率的素数検定

このメソッドは、エラーが発生したときに、RSA モジュライの生成または検証時に役立つ可能性のある追加情報を提供します。
RBG を承認済みのランダム ビット ジェネレーターとします。

入力：

1.w _　素数性をテストする奇数の整数。これは、 pまたはq、あるいは補助素数p1、 p2、 q1 、またはq2のいずれかになります。

2.反復　実行されるテストの反復回数。値は表 C.1、C.2、または C.3 と一致していなければなりません。

出力：

1.ステータス　検証プロシージャから返されるステータス。ステータスは、PROBABLY PRIME、Provably COMPOSITE WITH FACTOR (factor とともに返される)、および PROVABLY COMPOSITE AND NOT A POWER OF A PRIME のいずれかです。

プロセス：

1. aを 2a でw–1を割るような最大の整数とします。

2. m = (w-1) / 2a。

3. wlen = len (w)。

4. i = 1の場合、反復は次のようになります。

4.1 RBG からwlenビットの文字列bを取得します。

コメント: 1 < b < w–1 であることを確認してください。

4.2 ((b ≤ 1) または(b ≥ w–1)) の場合、ステップ 4.1 に進みます。

4.3 g = GCD(b, w)。

4.4 (g > 1) の場合は、 PROVABLY COMPOSITE WITH FACTORを返し、
gの値。

4.5z _ = bm mod w。

4.6 ((z = 1) または(z = w – 1)) の場合、ステップ 4.15 に進みます。

4.7 j = 1の場合、 a – 1 を実行します。

4.7.1 x = z. コメント: x ≠ 1 およびx ≠ w–1。

4.7.2z _ = x ^{2}モッドw。

4.7.3 (z = w–1) の場合、ステップ 4.15 に進みます。

4.7.4 (z = 1) の場合、ステップ 4.12 に進みます。

4.8 x = z. コメント: x = b(w–1)/2 mod wおよびx ≠ w–1。

4.9z _ 2 = × モッドw。

4.10 (z = 1) の場合、ステップ 4.12 に進みます。

4.11 x = z. コメント: x = b(w–1) mod wおよびx ≠ 1。

4.12 g = GCD(x–1, w)。

4.13 (g > 1) の場合は、 PROVABLY COMPOSITE WITH FACTORを返し、
gの値。

4.14明らかに複合的なものを返し、素数のパワーではありません。

4.15 続行します。

コメント:ステップ 4 の do ループのiをインクリメントします。

5.おそらくプライムを返します。

C.3.3 (一般) ルーカスの確率的素数検定

ルーカス テストとしては、次のプロセスまたはそれと同等のプロセスが使用されます。

入力：

C素数性をテストする候補の奇数整数。

出力：

status statusは、 PROBABLY PRIMEまたはCOMPOSITEのいずれかです。

プロセス：

1. C が完全な正方形かどうかをテストします (付録 C.4 を参照)。そうであれば、 (COMPOSITE) を返します。

2.ヤコビ関数が適用されるシーケンス {5, –7, 9, –11, 13, –15, 17, …} の最初のDを見つけます。

記号( ) $\frac{D}{C}$ = –1。ヤコビ係数を計算するための承認された方法については、付録 C.5 を参照してください。

シンボル。もし($\frac{D}{C}$ シーケンス内の任意のDに対して = 0 、 (COMPOSITE) を返します。

3. K = C+1。

4. Kr = 1 として、 Kr Kr – 1 … K0 をKの 2 進展開とします。

5. Ur = 1、 Vr = 1 に設定します。

6. i = r–1 ～ 0 の場合、次のようにします。

6.1 Utemp = Ui+1 Vi+1 mod C.

6.2 Vtemp = $\frac{Vi+1 Di+1 Ui+1}{2}$ モードC。

6.3 (Ki = 1) の場合、

コメント: Ki = 1 の場合は、ステップ 6.3.1 と 6.3.2 を実行します。それ以外の場合は、ステップ 6.3.3 と 6.3.4 を実行します。

6.3.1 Ui = $\frac{Utemp + Vtemp}{2}$ モッドC.

6.3.2 Vi = mod C. $\frac{Vtemp + DUtemp}{2}$

それ以外

6.3.3 Ui =使用温度

6.3.4 Vi = Vtemp。

7. (U0 = 0) の場合は、 (PROBABLY PRIME) を返します。それ以外の場合は、 (COMPOSITE) を返します。

ステップ 6.2、6.3.1、および 6.3.2 には、 A/2 mod C という形式の式が含まれています。ここで、 Aは整数、 Cは奇数の整数です。 A/2 が整数でない (つまり、 Aが奇数である)場合、 A/2 mod Cは(A+C)/2 mod Cとして計算できます。あるいは、 A/2 mod C = A•(C+1))/2 mod C、 Aが奇数であるか偶数であるかに関係なく、任意の整数Aに対して計算されます。

C.4 完全な正方形かどうかの確認

次のアルゴリズムを使用して、 n ビットの正の整数 C が完全な正方形であるかどうかを判断できます。

入力：

Cチェックされる整数。

出力：

status status は、 PERFECT SQUAREまたはNOT A PERFECT SQUAREのいずれかです。

プロセス：

1. 2n > C ≥ 2(n－1)となるように nを設定します。

2. m = n/2 。

3. i = 0。

4. 2m > X0 ≥ 2(m－1)となるようにX0を選択します。

5. 繰り返します

5.1 i = i + 1。

5.2 Xi = ((Xi–1) ^{2} + C)/(2Xi – 1)。

(西)まで ^{2} < 2m + C.

6. C = Xi 2の場合 、それから

ステータス=パーフェクトスクエア。

それ以外

ステータス=完璧な正方形ではありません。

7.ステータスを返します。

ノート：

1. X0 > (1/2) Sqrt(C)から始めると、 X0 − Sqrt(C)がX0未満であることが保証されます。
この不等式はステップ 5 でも維持されます。つまり、すべての i に対して Xi − Sqrt(C) < Xiです。

2. i ≥ 1 の場合、0 ≤ Xi − Sqrt(C) = (Xi–1 − Sqrt(C))2 / (2 Xi–1) < X0/2iです。

特に、0 ≤ Xm − Sqrt(C) < 1 です。Sqrt(C) が整数の場合、次のようになります。
Xmのフロアに等しくなります。

3. 一般に、 iの値が mよりはるかに小さい場合、不等式Xi − Sqrt(C) < 1 が発生します。これを検出するには、すべてのi ≥ 1 に対して2(m−1) ≤ Sqrt(C) < Xiであるという事実を使用できます。

$$\begin{aligned} Xi - Sqrt(C) &= ((Xi)^2 - C)/(Xi + Sqrt(C)) \\ &\le ((Xi)^2 - C)/(2\,Sqrt(C)) \\ &\le ((Xi)^2 - C)/(2m) \end{aligned}$$

したがって、条件$(Xi)^2 < 2m + C$ は、 Xi − Sqrt(C) < 1を意味します。

C.5 ヤコビ記号アルゴリズム

このルーチンは、ヤコビ記号 - を計算します。 $\left(\frac{a}{n}\right)$ 。

ヤコビ( ):

入力：

a 任意の整数。この規格の場合、初期値は {5、-7、9、-11、
13、-15、17、 …}、付録 C.3.3 によって決定されます。

n 任意の整数。この規格の場合、初期値は付録 C.3.3 によって決定されるテスト対象の候補です。

出力：

result計算されたヤコビ記号。

プロセス：

1. a = a mod n。 コメント: a は0 ≤ a < nの範囲になります。

2. a = 1、またはn = 1 の場合、(1) を返します。

3. a = 0 の場合、(0) を返します。

4. a = 2e a1となるようにeとa1を定義します。ここで、 a1は奇数です。

5. eが偶数の場合、 s = 1。

それ以外の場合、((n ≡ 1 (mod 8)) または(n ≡ 7 (mod 8))) の場合、 s = 1。

それ以外の場合、((n ≡ 3 (mod 8)) または(n ≡ 5 (mod 8)) の場合、 s = –1。

6. ((n ≡ 3 (mod 4)) かつ(a1 ≡ 3 (mod 4))) の場合、 s = –s となります。

7. n1 = n mod a1。

8. (s * ヤコビ(n1, a1)) を返します。 コメント: このルーチンを再帰的に呼び出します。

例: a = 5 およびn = 3439601197のヤコビ記号を計算します。

1. nは 1 ではなく、 aも 1 ではないため、手順 2 に進みます。

2. aは 0 ではないため、ステップ3に進みます。 3.

5 = 20 * 5、つまりe = 0、およびa1 = 5。

4. eは偶数なので、 s = 1。

5. a1は 3 mod 4 と合同ではないため、 s を変更しないでください。

6. n1 = 2 = n mod 5.

7. (1 * Jacobi(2, 5)) を計算して返します。これは Jacobi を再帰的に呼び出します。 a = 2 およびn = 5のヤコビ記号を計算します。

7.1 nは 1 ではなく、 aも 1 ではないため、ステップ 7.2 に進みます。

7.2 aは 0 ではないため、ステップ 7.3 に進みます。 7.3

2 = 21 * 1 なので、 e = 1、 a1 = 1 となります。

7.4 eは奇数で、 n ≡ 5 (mod 8) なので、 s = –1 と設定します。

7.5 nは 3 mod 4 ではなく、 a1は 3 mod 4 ではないため、手順 7.6 に進みます。

7.6 n1 = 0 = n mod 1。

7.7 戻り値 (–1 * Jacobi(0, 1) = –1)。これは Jacobi を再帰的に呼び出します。ヤコビ係数を計算する
a = 0 およびn = 1の記号:

7.7.1 n = 1 なので 1 を返します。

したがって、ヤコビ (0,1) = 1、つまりヤコビ (2,5) = –1*(1) = –1、およびヤコビ (5, 3439601197) = 1* (–1) = –1 となります。

C.6 Shawe-Taylor Random_Prime ルーチン

このルーチンは再帰的であり、ハッシュ関数を使用して証明可能な素数を構築するために使用できます。

選択されたハッシュ関数をHash()とし、ハッシュ関数出力ブロックのビット長をoutlenとします。この構成的方法の素数を生成するには、次のプロセスまたはそれと同等のプロセスを使用します。

ST_ランダム_プライム ( ):

入力：

1.長さ	生成される素数の長さ。
2.入力シード	要求された素数の生成に使用されるシード。

出力：

1.ステータス	生成ルーチンから返されるステータス。ステータスはSUCCESSまたはFAILUREのいずれかです。 FAILUREが返された場合は、他の出力値として 0 が返されます。
2.プライム	要求された素数。
3プライムシード	生成時に決定される種子。
4. prime_gen_counter (オプション)	プライム。

プロセス：

1. (長さ< 2) の場合は、 (FAILURE, 0, 0 {, 0}) を返します。

2. (長さ≥ 33) の場合は、ステップ 14 に進みます。

3.プライムシード=入力シード。

4. prime_gen_counter = 0。

コメント:長さビットの擬似乱数整数cを生成します。

5. c =ハッシュ(プライムシード) ⊕ハッシュ(プライムシード+ 1)。

6. c = 2length – 1 + (c mod 2length – 1 ）。

7. c = (2 c / 2) + 1。

コメント:素数をc以上の最小の奇数の整数に設定します。

8. prime_gen_counter = prime_gen_counter + 1。

9.プライムシード=プライムシード+ 2。

10. c に対して決定的な素数性テストを実行します。たとえば、 cは小さいので、その素数は次のようになります。
トライアル分割でテスト可能です。付録 C.7 を参照してください。

11. (cが素数) の場合、

11.1素数= c.

11.2 戻り値(SUCCESS、 prime、 prime_seed {、 prime_gen_counter})。

12. (prime_gen_counter > (4 length)) の場合、 (FAILURE, 0, 0 {, 0})を返します。

13. 手順 5 に進みます。

14. (status, c0, prime_seed, prime_gen_counter) = (ST_Random_Prime ((length / 2)
+ 1)、 input_seed)。

15. FAILUREが返された場合は、 (FAILURE, 0, 0 {, 0})を返します。

16.反復= 長さ/アウトレングス – 1.

17. old_counter = prime_gen_counter。

コメント:区間[2length – 1]で擬似乱数整数xを生成します。
、 2長さ]。

18. x = 0。

19. For i = 0 から反復はdo

x = x + (ハッシュ(prime_seed + i) 2i × outlen)。

20.プライムシード=プライムシード+反復回数 + 1。

21. x = 2length – 1 + (x mod 2length – 1)。

コメント:区間[2length – 1]で素数候補cを生成します。
、 2長さ]。

22. t = x / (2c0) 。

23. (2tc0 + 1 > 2length) の場合、 t = 2length – 1 / (2c0) 。

24. c = 2tc0 + 1。

25.prime_gen_counter = prime_gen_counter + 1。

コメント: 候補素数cの素数性をテストします。まず、 2
からc – 2 までの整数aを選択します。

26. a = 0。

27. For i = 0から反復は行う

a = a + (ハッシュ(prime_seed + i) 2 私*アウトレン)。

28.プライムシード=プライムシード+反復回数+ 1。

29. a = 2 + (a mod (c – 3))。

30. z = a ^{2t} モッドc。

31. ((1 = GCD(z – 1, c)) および (1 = z mod c)^{0c}の場合、

31.1素数= c.

31.2 戻り値(SUCCESS, prime, prime_seed {, prime_gen_counter})。

32. (prime_gen_counter ≥ ((4 length) + old_counter)) の場合、 (FAILURE, 0, 0)を返します。
{, 0})。

33. t = t + 1。

34. 手順 23 に進みます。

C.7 治験部門

整数が素数であることは、その平方根以下の素因数がないことを示すことで証明されます。この手順は、10 桁を超える整数をテストする場合には推奨されません。

cが素数であることを証明するには:

1. c未満の素数の表を準備します。これは、次の$\sqrt{}$ふるい手順を適用することで実行できます。
付録 C.8。
2. cを表内のすべての素数で割ります。 c が素数の 1 つで割り切れる場合、 cを宣言します。
複合して終了します。都合がよければ、 c を合成数で割ることもできます。たとえば、素数の表を準備するよりも、3 または 5 で割り切れる整数を除くすべての整数で割った方が便利な場合があります。

3. それ以外の場合は、 cが素数であると宣言して終了します。

C.8 ふるい分け手順

ふるいの手順は次のように説明されます。整数Y0、 Y0 + 1、…、 Y0 + J のシーケンスが与えられると、ふるいは、選択された制限までの素数で割り切れるシーケンス内の整数を識別します。

このプロセスにおける数学記号の定義 (h、 L、 M、 p など) はこのプロセスの内部でのみ定義されており、この規格の他の場所での使用と混同しないように注意してください。

まず、すべての素数pjの因子基底を2 から選択された限界 L まで選択します。Lの値は任意であり、コンピューターの制限によって決定される場合があります。 Lの適切な典型的な値
103 から105 までの範囲になります 。

1.因子ベース内のすべてのpjについてSj =Y0 mod pjを計算します。

2. 長さJ + 1の配列をゼロに初期化します。

3. Y0 – Sj + pjから始めて、すべてのpjを 番目 配列の要素を 1 に設定します。これを
配列の全長とすべてのj について。

4. 終了すると、値 1 を持つ配列内のすべての位置が小さな素数で割り切れるため、合成になります。

配列は、メモリが小さい場合はコンパクトにするためにビット配列にすることができ、メモリがすぐに利用できる場合には速度を上げるためにバイト配列にすることができます。ふるい間隔全体を一度にふるい分ける必要はありません。アレイは、次の部分に進む前に各部分をふるいにかけ、適切な小さな部分に分割できます。終了すると、値 0 を持つすべての場所が主要テストの候補になります。

この手順の作業量は約M log log L です。ここで、 Mはふるいの間隔の長さです。これは、素数性テストの複合候補を削除するための非常に効率的な手順です。 L = 105の場合
、ふるいはすべての複合材料の約 96% を除去します。

場合によっては、連続した整数のセットをふるいにかけるのではなく、テスト対象の整数のセットが等差数列Y0、 Y0 + h、 Y0 + 2h、…、 Y0 + Jh (hが大きい場合)にある整数で構成されます。因数基底のどの素数でも割り切れません。

1. 因子基数を選択し、長さJ + 1の配列を0 に初期化します。

2.因子ベース内のすべてのpjについてSj =Y0 mod pjを計算します。

3. Tj = h mod pjおよびr = mod pj を計算しますTj – 1

4. Y0 + r から始めて、すべてのpjを 番目 配列の要素を 1 に設定します。これを全体に対して実行します。
配列の長さとすべてのj。配列内の位置Y0 + rは、実際には数値Y0 + rh を表すことに注意してください。

5. 終了すると、値 1 を持つ配列内のすべての位置が、いくつかの小さい値で割り切れます。
プライムなので複合です。

注: 素数「2」は、ふるい配列内の最も多くの場所に触れるため、ふるいにかけるのに最も長い時間 (M/2) かかります。簡単な最適化は、ふるい配列の初期化と素数「2」のふるい分けを組み合わせることです。初期化中に素数「3」をふるいにかけることも可能です。これらの最適化により、総ふるい時間の約 1/3 を節約できます。

C.9 補助素数に基づいて推定素因数を計算する

このルーチンは、2 つの補助素数と中国剰余定理 (CRT) を使用して、確率素数 (pまたはq の候補) を構築します。

入力：

r1とr2 log2(r1r2) ≤ (nlen/2) – log2(nlen/2) –
6を満たす 2 つの奇数の素数。

ンレン	nの目的の長さ、RSA 係数。
e	公的検証指数。
セキュリティ強度	乱数生成に必要な最低限のセキュリティ強度。

出力：

状態	生成プロシージャから返されるステータス。ステータスはSUCCESSまたはFAILUREのいずれかです。 FAILUREが返された場合は、他の出力値として 0 が返されます。

private_prime_factor nの素因数。

バツ	private_prime_factorの生成中に使用される乱数。

プロセス：

1. (GCD(2r1, r2) ≠ 1) の場合、 (FAILURE, 0, 0) を返します。
2. R = ((r2 $^{-1}$ mod 2r1) * r2) – (((2r1) $^{-1}$ mod r2) * 2r1)。

   コメント: R ≡ 1 (mod 2r1)となるように CRT を適用します。
   そしてR ≡ –1 (mod r2)。
3. 承認された乱数発生器を使用して、 X ≤ ≤の乱数Xを生成します。
   (2)(2のようなsecurity_Strengthをサポートします。$\sqrt{}$ nlen / –12) (2 nlen / 2 – 1)。
4. Y = X + ((R – X) mod 2r1r2)。

   コメント: Y は最初の奇数 ≥ X であり、 r1が成立します。
   は Y–1 の素因数であり、 r2は Y+1の素因数です。

   コメント: シーケンスから候補を構築し、素数テストを実行することにより、要求された素数を決定します。
5. i = 0。
6. (Y ≥ 2nlen/2) の場合は、ステップ 3 に進みます。
7. (GCD(Y–1, e) = 1) の場合、
   7.1 付録 C.3 に規定されているように、 Yの素数性をチェックします。おそらくプライムなら
   が返されない場合は、手順 8 に進みます。
   7.2 private_prime_factor = Y。
   7.3 リターン(SUCCESS、 private_prime_factor、 X)。
8. i = i + 1。

9. (i ≥ 5(nlen/2)) の場合は、 (FAILURE, 0, 0)を返します。

10. Y = Y + (2r1r2)。

11. 手順 6 に進みます。

C.10 同時に構築された補助証明可能素数に基づいて証明可能素数を構築する (おそらく条件付き)

次のプロセス (または同等のプロセス) を使用して、 L ビットの証明可能な素数p (RSA モジュラスの素因数の 1 つの候補)を生成します。この仕様でのpの使用は一般的に使用されることに注意してください。 RSA 素因数pとq は両方とも、この方法を使用して生成できます。

いわゆる「強い素数」が必要な場合、このプロセスは、それぞれp-1 とp+1を除算する素数p1とp2 (指定されたビット長N1とN2)を生成できます。結果として得られる素数p は、 p1とp2に要求されたビット長が適切なサイズであれば、強力な素数に伝統的に要求される条件を満たします。

p1とp2に選択されたビット長に関係なく、量p－1 は素数p0を持ちます。
そのビット長はp の半分よりわずかに大きいです。さらに、量p0 －1 は、ビット長がp0 の半分より
わずかに大きい素約数を持ちます。

このアルゴリズムでは、 N1 + N2 ≤ L － L/2 － 4 であることが必要です。アルゴリズムが確実に機能するように、N1とN2の値は、 N1 + N2 ≤ (L/2) – log2(L) – 7になるように選択する必要があります。最大5L生成可能pの明確な候補者。

使用する選択されたハッシュ関数をHashとし、ハッシュ関数出力ブロックのビット長をoutlenとします。

Provable_Prime_Construction():

入力：

1.L _	pの要求されたビット長に等しい正の整数。 L= nlen/2の許容値は、付録 B.3.1、基準 2(b) および (c) に指定されているように計算され、 nlen は表 B.1 に指定されている値を想定していることに注意してください。
2.N1 _	p1の要求されたビット長に等しい正の整数。 N1 ≧の場合 2 の場合、 p1はN1ビットの奇数素数です。それ以外の場合、 p1 = 1。N1 ≥ 2 の許容値は表 B.1 に示されています。
3.N2 _	p2に要求されたビット長に等しい正の整数。 N2 ≧の場合 2 の場合、 p2はN2ビットの奇数素数です。それ以外の場合、 p2 = 1。 N2 ≥ 2の許容値を表 B.1 に示します。
4.ファーストシード	使用される最初のシードに等しいビット文字列。

5. e 公的検証指数。

出力：

1.ステータス 生成プロシージャから返されるステータス。ステータスはSUCCESSまたはFAILUREのいずれかです。 FAILUREが返された場合は、他の出力値として0 が返されます。

2.p 、 p1、 p2 必要な素数p と、p1 が p-1 を除算し、 p2がp + 1を除算する特性を持つp1と p2 。

3.シード 生成時に決定される種子。

プロセス：

1. L、 N1、およびN2が受け入れられない場合は、 (FAILURE, 0, 0, 0, 0) を返します。

コメント: p1とp2、および素数を生成します。
p0.

2. N1 = 1 の場合、

2.1 p1 = 1。

2.2 p2seed =最初のシード。

3. N1 ≥ 2 の場合、

3.1 N1を長さとして、 firstseed をinput_seedとして使用し、付録 C.6 のランダム素数生成ルーチンを使用してp1とp2seed を取得します。

3.2 FAILUREが返された場合は、 (FAILURE, 0, 0, 0, 0)を返します。

4. N2 = 1 の場合、

4.1 p2 = 1。

4.2 p0seed = p2seed。

5. N2 ≥ 2 の場合、

5.1 N2を長さとして、 p2seed をinput_seed として使用し、付録 C.6 のランダム素数生成ルーチンを使用してp2とp0seed を取得します。

5.2 FAILUREが返された場合は、 (FAILURE, 0, 0, 0, 0)を返します。

6.長さとして L / 2 + 1 を使用し、input_seedとしてp0seed を使用し、ランダム素数を使用します。p0とpseedを取得するための生成ルーチンは、付録 C.6 にあります。 FAILUREが返された場合は、(FAILURE, 0, 0, 0, 0)を返します。

コメント:区間 [(2)(2L－1), 2L －1]で(強い) 素数pを生成します。√

7.反復= L / outlen −1。

8. pgen_counter = 0。

コメント:区間 [(2)(2L−1)−1, 2L −1]で擬似乱数xを生成します。√

9. x = 0。

10. For i = 0 から反復は行う

レン。 x = x + (ハッシュ(pseed + i)) 2 私*アウト

11. pseed = pseed +反復回数+ 1. 12. x =

(2)(2L−1) + (x mod (2L − (2)(2L−1)√))。

コメント: 素数 p の候補を生成します。

13. (GCD(p0p1, p2) ≠ 1) の場合は、 (FAILURE, 0, 0, 0, 0) を返します。

14. 0 = (y p0 p1–1) mod p2となるように、区間 [1, p2]でyを計算します。 15. t = ((2 y p0 p1) + x)/(2 p0 p1 p2) 。

16. ((2(t p2 − y) p0 p1 + 1) > 2L) の場合、 t =

((2 y p0 p1) + (2)(2L−1)) / (2 p0 p1 p2) 。

コメント: p は0 = (p–1)
mod (2p0 p1)および 0 = (p+1)
mod p2 を満たします。

17. p = 2(t p2 − y) p0 p1 + 1。

18. pgen_counter = pgen_counter + 1。

19. (GCD(p–1, e) = 1) の場合、

コメント: [2, p– 2]の範囲内の整数aを選択します。

19.1 a = 0

19.2 For i = 0から反復実行まで

レン。 a = a + (ハッシュ(pseed + i)) 2 私*アウト

19.3 pseed = pseed +反復回数+ 1。

19.4 a = 2 + (a mod (p–3))。

コメント: pの素数性をテストします:

19.5z _ = a 2(t p2 − y) p1 mod p。

GCD(z–1, p)) および (1 = (z p2, pseed)) を返します。 p0 mod p)) を返し、 (SUCCESS, p, p1, 19.6 If ((1 =

20. (pgen_counter ≥ 5L) の場合は、 (FAILURE, 0, 0, 0, 0) を返します。

21. t = t + 1。

22. 手順 16 に進みます。

付録 D: 連邦政府向けに推奨される楕円曲線
政府利用

この楕円曲線のコレクションは連邦政府での使用が推奨されており、秘密キーの長さと基礎となるフィールドの選択肢が含まれています。これらの曲線は、SHA-1 と、ANS X9.62 および IEEE Standard 1363-2000 標準で指定された方法を使用して生成されました。この付録では、使用されたプロセスについて説明します。これらの曲線は、この規格の以前のバージョンに含まれていたものと同じであることに注意してください。

D.1 NIST 推奨の楕円曲線

D.1.1 選択肢

D.1.1.1 鍵の長さの選択

楕円曲線暗号の主なパラメータは、楕円曲線Eと、基点と呼ばれるE上の指定された点Gです。基点の次数はn であり、大きな素数です。曲線上の点の数は、 nで割り切れない整数h (余因子)のhnです。

効率上の理由から、補因子はできるだけ小さいことが望ましいです。

以下に示すすべての曲線には余因子 1、2、または 4 があります。その結果、曲線の秘密鍵と公開鍵はほぼ同じ長さになります。

D.1.1.2 基礎となるフィールドの選択

キーの長さごとに 2 種類のフィールドが提供されます。

- 素体は、要素の素数pを含むフィールドGF(p)です。の
  このフィールドの要素は p を法とする整数であり、フィールドの演算は p を法とする整数の算術として実装されます。
- バイナリ フィールドはフィールドGF(2m) で、これにはいくつかのmに対して2m個の要素が含まれます(
  分野の程度)。このフィールドの要素は長さm のビット列であり、フィールド演算はビットの演算によって実装されます。

nのビット長の 5 つの範囲のセキュリティ強度は、 SP 800-57 で提供されます。フィールドGF(p) の場合、セキュリティ強度はp のバイナリ展開の長さに依存します。フィールドGF(2m) の場合、セキュリティ強度はm の値に依存します。表 E-1 は、この付録で提供される曲線の基礎となるさまざまなフィールドのビット長を示しています。列 1 には、 n のビット長の範囲がリストされています(セクション 6.1.1 の表 1 も参照)。列 2 は、素体上の曲線に使用されるpの値を示します。ここで、 len(p)は、整数p の 2 進展開の長さです。列 3 は、バイナリ フィールド上の曲線のmの値を示します。

表 D-1: 推奨される曲線の基礎となるフィールドのビット長

nのビット長	プライムフィールド	バイナリフィールド
161 – 223	len(p) = 192	m = 163
224 – 255	len(p) = 224	m = 233
256 – 383	len(p) = 256	m = 283
384 – 511	len(p) = 384	m = 409
≥ 512	len(p) = 521	m = 571

D.1.1.3 バイナリフィールドの基底の選択

バイナリ フィールドの算術演算を記述するには、まずビット文字列がどのように解釈されるかを指定する必要があります。これは、フィールドの基準の選択と呼ばれます。基底には、多項式基底と正規基底の 2 つの一般的なタイプがあります。

- 多項式基底は、体多項式と呼ばれる 2 を法とする既約多項式によって指定されます。ビット文字列(am–1 … a2 a1 a0)は多項式を表すために使用されます。

$$1am - 1t^{m-} + \cdots + a2\,t^{2} + a1\,t + a0$$

GF(2)を超えます。フィールド演算は、p(t) を法とする多項式演算として実装されます。ここで、 p(t)はフィールド多項式です。

- 正規基底は要素によって指定されます… am–1) が要素を表す θ 特定の種類の。ビット列(a0 a1 a2 ために取られます

$$a0\,\theta + a1\,\theta^{2} + a2\,\theta^{2^{2}} + \cdots + 午前-1\,\theta^{2^{m-1}}。$$

正規基底体の演算は、タイプ T の低複雑性正規基底と呼ばれる特別なクラスを除いて、一般に記述が簡単ではなく、実装も効率的ではありません。与えられた体の次数 m に対して、 Tの選択により基底と体の演算が指定されます (付録 D.3 を参照)。

選択できる多項式基底と通常の基底が多数あります。基底表現を選択するには、次の手順が一般的に使用されます。

•多項式基底:既約三項式 tの場合 + tk + 1 がGF (2)上に存在する場合、フィールドは多項式p(t)は、最低次数の中間をもつ既約三項式となるように選択されます。既約三項式が存在しない場合は、五項式 tが選項t k 。択されます。選択された特定の五項式には次の特性があります。第 2 項は最低 +t b + t + t c +1は次数m を持ちます。第三項 t b すべての既約の中で最も低い次数を持ちます ;そして第 4 項t c 最低次数と第 3 項tを持ちま 、す。 。

次数mと第 2 項tの五項式 m次のすべての既約五項式の中で、第 2 項t

•正規基底: Tが最小のタイプ T の低複雑性正規基底を選択します。

各バイナリ フィールドに対して、上記の基底表現のパラメータが与えられます。

D.1.1.4 曲線の選択

2 種類の曲線が与えられます。

•擬似ランダム曲線は、シードされた暗号化ハッシュ関数の出力から係数が生成される曲線です。ドメイン パラメーターのシード値が係数とともに指定されている場合、係数がその方法で生成されたものであることを簡単に検証できます。

•特殊曲線とは、楕円曲線演算の効率を最適化するために係数と基礎となるフィールドが選択されている曲線です。

各曲線サイズ範囲に対して、次の曲線が与えられます。

→ GF(p)上の擬似ランダム曲線。

→ GF(2m) 上の擬似ランダム曲線。

→コブリッツ曲線または異常二値曲線と呼ばれる、 GF(2m)上の特殊な曲線。

擬似ランダム曲線は、ANS X9.62 の規定に従って SHA-1 を使用して生成されました。

D.1.1.5 基点の選択

次数nの任意の点が基点として機能できます。各曲線にはサンプル基点G = (Gx , Gy) が指定されます。ユーザーは、ネットワークを暗号的に分離するために独自のベース ポイントを生成したい場合があります。 ANS X9.62 または IEEE 標準 1363-2000 を参照してください。

D.1.2 素体上の曲線

各素数p の擬似ランダム曲線

$$E : y^{2} \equiv x^{3} - 3x + b \pmod{p}$$

素数次数nがリストされています[4] 。 (したがって、これらの曲線の余因子は常にh = 1 です。) 次のパラメーターが与えられます。

- 素数法p
- 次数n
- SHA-1 ベースのアルゴリズムへの160 ビット入力シードSEED (つまり、ドメイン パラメーター) シード)
- SHA-1 ベースのアルゴリズムの出力c

[4] xの係数としてa ≡ -3が選択されたのは、効率性の理由からです。 IEEE 標準 1363-2000 を参照してください。

- 係数b (b2 c ≡ –27 (mod p) を満たす)
- •基点x座標Gx
- •基点y座標Gy

整数pとnは 10 進数形式で与えられます。ビット文字列とフィールド要素は 16 進数で指定されます。

D.1.2.1 曲線 P-192

p = 6277101735386680763835789423207666416083908700390324961279

n = 6277101735386680763835789423176059013767194773182842284081

シード= 3045ae6f c8422f64 ed579528 d38120ea e12196d5

c = 3099d2bb bfcb2538 542dcd5f b078b6ef 5f3d6fe2 c745de65

b = 64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1

G_x = 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012

G_y = 07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811

D.1.2.2 曲線 P-224

p = 26959946667150639794667015087019630673557916260026308143510066298881

n = 26959946667150639794667015087019625940457807714424391721682722368061

シード= bd713447 99d5c7fc dc45b59f a3b9ab8f 6a948bc5

c = 5b056c7e 11dd68f4 0469ee7f 3c7a7d74 f7d12111 6506d031218291fb

b = b4050a85 0c04b3ab f5413256 5044b0b7 d7bfd8ba 270b39432355ffb4

G_x = b70e0cbd 6bb4bf7f 321390b9 4a03c1d3 56c21122 343280d6115c1d21

G_y = bd376388 b5f723fb 4c22dfe6 cd4375a0 5a074764 44d5819985007e34

D.1.2.3 曲線 P-256

p = 115792089210356248762697446949407573530086143415290314195533631308867097853951

n = 115792089210356248762697446949407573529996955224135760342422259061068512044369

シード = c49d3608 86e70493 6a6678e1 139d26b7 819f7e90

c = 7efba166 2985be94 03cb055c 75d4f7e0 ce8d84a9 c5114abc af317768 0104fa0d

b = 5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e 27d2604b

$G_{バツ}$ = 6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81 2deb33a0 f4a13945 d898c296

G = 4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece y cbb64068 37bf51f5

D.1.2.4 曲線 P-384

p = 394020061963944792122790401001436138050797392704654466679482934042457217714968703290472660882589380018616069731123l9

n = 39402006196394479212279040100143613805079739270465446667946905279627659399113263569398956308152294913554433653942643

シード = a335926a a319a27a 1d00896a 6773a482 7acdac73

c = 79d1e655 f868f02f ff48dcde e14151dd b80643c1 406d0ca1 0dfe6fc5 2009540a 495e8042 ea5f744f 6e184667 cc722483

b = b3312fa7 e23ee7e4 988e056b e3f82d19 181d9c6e fe814112 0314088f 5013875a c656398d 8a2ed19d 2a85c8ed d3ec2aef

$G_{バツ}$ = aa87ca22 be8b0537 8eb1c71e f320ad74 6e1d3b62 8ba79b98 59f741e0 82542a38 5502f25d bf55296c 3a545e38 72760ab7

G_y = 3617de4a 96262c6f 5d9e98bf 9292dc29 f8f41dbd 289a147c e9da3113 b5f0b8c0 0a60b1ce 1d7e819d 7a431d7c 90ea0e5f

D.1.2.5 曲線 P-521

p = 6864797660130609714981900799081393217269435300143305409394463459185543183397656052122559640661454554977296311391480858037121987999716643812574028291115057151

n = 6864797660130609714981900799081393217269435300143305409394463459185543183397655394245057746333217197532963996371363321113864768612440380340372808892707005449

シード = d09e8800 291cb853 96cc6717 393284aa a0da64ba

c = 0b4 8bfa5f42 0a349495 39d2bdfc 264eeeeb 077688e4 4fbf0ad8 f6d0edb3 7bd6b533 28100051 8e19f1b9 ffbe0fe9 ed8a3c22 00b8f875 e523868c 70c1e5bf 55bad637

b = 051 953eb961 8e1c9a1f 929a21a0 b68540ee a2da725b 99b315f3 b8b48991 8ef109e1 56193951 ec7e937b 1652c0bd 3bb1bf07 3573df88 3d2c34f1 ef451fd4 6b503f00

$G_{バッ}$ = c6 858e06b7 0404e9cd 9e3ecb66 2395b442 9c648139 053fb521 f828af60 6b4d3dba a14b5e77 efe75928 fe1dc127 a2ffa8de 3348b3c1 856a429b f97e7e31 c2e5bd66

G_{y} = 118 39296a78 9a3bc004 5c8a5fb4 2c7d1bd9 98f54449 579b4468 17afbd17 273e662c 97ee7299 5ef42640 c550b901 3fad0761 353c7086 a272c240 88be9476 9fd16650

D.1.3 バイナリフィールド上の曲線

各フィールド次数 m に対して、コブリッツ曲線とともに擬似ランダム曲線が与えられます。擬似ランダム曲線の形式は次のとおりです。

$$E: はい^{2} + xy = x^{3} + ×^{2} + b、$$

そして、コブリッツ曲線は次のような形になります。

$$え：はい^{2} + xy = x^{3} + 斧^{2} + 1、$$

ここで、a = 0 または 1。

各擬似ランダム曲線の余因子は h = 2 です。各コブリッツ曲線の余因子は、a = 1 の場合は h = 2、a = 0 の場合は h = 4 です。

擬似ランダム曲線の係数と両方の種類の曲線の基点の座標は、付録 D.1.1.3 で説明されている多項式と正規の基底表現の両方で与えられます。

各 m に対して、次のパラメータが与えられます。

フィールド表現:

- 通常の基底型 T
- 体の多項式 (3 項または 5 項)

コブリッツ曲線:

- 係数 a
- 基点次数 n
- 基点x座標G
- 基点y座標G

擬似ランダム曲線:

- 基点次数 n

擬似ランダム曲線 (多項式基底表現):

- 係数 b
- 基点x座標G
- 基点y座標G

擬似ランダム曲線 (Normal Basis 表現):

- SHA-1 ベースのアルゴリズムへの 160 ビット入力シード SEED (つまり、ドメイン パラメーター)
  シード）
- 係数 b (つまり、SHA-1 ベースのアルゴリズムの出力)
- 基点x座標G
- 基点y座標G

整数 (T、m、n など) は 10 進数形式で与えられます。ビット文字列とフィールド要素は 16 進数で指定されます。

D.1.3.1 次数 163 バイナリ フィールド

T = 4

$p(t) = t^{163} + t^{7} + t^{6} + t^{3} + 1$

D.1.3.1.1 曲線 K-163

a = 1

n = 5846006549323611672814741753598448348329118574063

多項式の基礎:

G x = 2 fe13c053 7bbc11ac aa07d793 de4e6d5e 5c94eee8

G y = 2 89070fb0 5d38ff58 321f2e80 0536d538 ccdaa3d9

通常の基準:

G x = 0 5679b353 caa46825 fea2d371 3ba450da 0c2a4541

G y = 2 35b7c671 00506899 06bac3d9 dec76a83 5591edb2

D.1.3.1.2 曲線 B-163

n = 5846006549323611672814742442876390689256843201587

多項式の基礎:

b = 2 0a601907 b8c953ca 1481eb10 512f7874 4a3205fd

G x = 3 f0eba162 86a2d57e a0991168 d4994637 e8343e36

G y = 0 d51fbc6c 71a0094f a2cdd545 b11c5c0c 797324f1

通常の基準:

シード = 85e25bfe 5c86226c db12016f 7553f9d0 e693a268

b = 6 645f3cac f1638e13 9c6cd13e f61734fb c9e3d9fb

G x = 0 311103c1 7167564a ce77ccb0 9c681f88 6ba54ee8

G y = 3 33ac13c6 447f2e67 613bf700 9daf98c8 7bb50c7f

D.1.3.2 次数 233 バイナリ フィールド

T = 2

$p(t) = t^{233} + t^{74} + 1$

D.1.3.2.1 曲線 K-233

a = 0

n = 3450873173395281893717377931138512760570940988862252126328087024741343

多項式の基礎:

G x = 172 32ba853a 7e731af1 29f22ff4 149563a4 19c26bf5 0a4c9d6e efad6126

G y = 1db 537dece8 19b7f70f 555a67c4 27a8cd9b f18aeb9b 56e0c110 56fae6a3

通常の基準:

G x = 0fd e76d9dcd 26e643ac 26f1aa90 1aa12978 4b71fc07 22b2d056 14d650b3

G y = 064 3e317633 155c9e04 47ba8020 a3c43177 450ee036 d6335014 34cac978

D.1.3.2.2 曲線 B-233

n = 6901746346790563787434755862277025555839812737345013555379383634485463

多項式の基礎:

b = 066 647ede6c 332c7f8c 0923bb58 213b333b 20e9ce42 81fe115f 7d8f90ad

G x = 0fa c9dfcbc 8313bb21 39f1bb75 5fef65bc 391f8b36 f8f8eb73 71fd558b

G y = 100 6a08a419 03350678 e58528be bf8a0bef f867a7ca 36716f7e 01f81052

通常の基準:

シード = 74d59ff0 7f6b413d 0ea14b34 4b20a2db 049b50c3

b = 1a0 03e0962d 4f9a8e40 7c904a95 38163adb 82521260
0c7752ad 52233279

G = 18b 863524b3 cdfefb94 f2784e0b 116faac5 4404bc91 62a363ba b84a14c5

G = 049 25df77bd 8b8ff1a5 ff519417 822bfedf 2bbd7526
y 44292c98 c7af6e02

D.1.3.3 次数 283 バイナリ フィールド

T = 6

$p(t) = t^{283} + t^{12+t} + t^{7\,5+t} + 1$

D.1.3.3.1 曲線 K-283

a = 0

n = 3885337784451458141838923813647037813284811733793061324

29587499752981582970442260387 3

多項式の基礎:

G = 503213f 78ca4488 3f1a3b81 62f188e5 53cd265f 23c1567a

16876913 b0c2ac24 58492836

G = 1ccda38 0f1c9e31 8d90f95d 07e5426f e87e45c0 e8184698 y

e4596236 4e341161 77dd2259

通常の基準:

G = 3ab9593 f8db09fc 188f1d7c 4ac9fcc3 e57fcd3b db15024b

212c7022 9de5fcd9 2eb0ea60

G = 2118c47 55e7345c d8f603ef 93b98b10 6fe8854f feb9a3b3 y

04634cc8 3a0e759f 0c2686b1

D.1.3.3.2 曲線 B-283

n = 77706755689029162836778476272940756265696259243769048891091965267700442777873786928711

多項式の基礎:

b = 27b680a c8b8596d a5a4af8a 19a0303f ca97fd76 45309fa2 a581485a f6263e31 3b79a2f5

G_x = 5f93925 8db7dd90 e1934f8c 70b0dfec 2eed25b8 557eac9c 80e2e198 f8cdbecd 86b12053

G_y = 3676854 fe24141c b98fe6d4 b20d02b4 516ff702 350eddb0 826779c8 13f0df45 be8112f4

通常の基準:

シード = 77e2b073 70eb0f83 2a6dd5b6 2dfc88cd 06bb84be

b = 157261b 894739fb 5a13503f 55f0b3f1 0c560116 66331022 01138cc1 80c0206b dafbc951

G_x = 749468e 464ee468 634b21f7 f61cb700 701817e6 bc36a236 4cb8906e 940948ea a463c35d

G_y = 62968bd 3b489ac5 c9b859da 68475c31 5bafcdc4 ccd0dc90 5b70f624 46f49c05 2f49c08c

D.1.3.4 次数 409 バイナリ フィールド

T = 4

$p(t) = t^{409} + t^{87} + 1$

D.1.3.4.1 曲線 K-409

a = 0

n = 330527984395124299475957654016385519914202341482140609641324395022880711289249191050673258457777458014096366590617731358671

多項式の基礎:

G = 060f05f 658f49c1 ad3ab189 0f718421 0efd0987 e307c84c
27accfb8 f9f67cc2 c460189e b5aaaa62 ee222eb1 b35540cf
e9023746

Gy = 1e36905 0b7c4e42 acba1dac bf04299c 3460782f 918ea427
e6325165 e9ea10e3 da5f6c42 e9c55215 aa9ca27a 5863ec48
d8e0286b

通常の基準:

G = 1b559c7 cba2422e 3affe133 43e808b5 5e012d72 6ca0b7e6
a63aeafb c1e3a98e 10ca0fcf 98350c3b 7f89a975 4a8e1dc0
713cec4a

G = 16d8c42 052f07e7 713e7490 eff318ba 1abd6fef 8a5433c8 y
94b24f5c 817aeb79 852496fb ee803a47 bc8a2038 78ebf1c4
99afd7d6

D.1.3.4.2 曲線 B-409

n = 661055968790248598951915308032771039828404682964281219 2
8464879830415777482737480520814372376217911096597986728 8
366567526771

多項式の基礎:

b = 021a5c2 c8ee9feb 5c4b9a75 3b7b476b 7fd6422e f1f3dd67
4761fa99 d6ac27c8 a9a197b2 72822f6c d57a55aa 4f50ae31
7b13545f

G = 15d4860 d088ddb3 496b0c60 64756260 441cde4a f1771d4d
b01ffe5b 34e59703 dc255a86 8a118051 5603aeab 60794e54
bb7996a7

G y = 061b1cf ab6be5f3 2bbfa783 24ed106a 7636b9c5 a7bd198d
0158aa4f 5488d08f 38514f1f df4b4f40 d2181b36 81c364ba
0273c706

通常の基準:

シード = 4099b5a4 57f9d69f 79213d09 4c4bcd4d 4262210b

b = 124d065 1c3d3772 f7f5a1fe 6e715559 e2129bdf a04d52f7
b6ac7c53 2cf0ed06 f610072d 88ad2fdc c50c6fde 72843670
f8b3742a

G = 0ceacbc 9f475767 d8e69f3b 5dfab398 13685262 bcacf22b
84c7b6dd 981899e7 318c96f0 761f77c6 02c016ce d7c548de
830d708f

G y = 199d64b a8f089c6 db0e0b61 e80bb959 34afd0ca f2e8be76
d1c5e9af fc7476df 49142691 ad303902 88aa09bc c59c1573
aa3c009a

D.1.3.5 次数 571 バイナリ フィールド

T = 10

$p(t) = t^{571} + t^{10} + t^{5} + t^{2} + 1$

D.1.3.5.1 曲線 K-571

a = 0

n = 1932268761508629172347675945465993672149463664853217499
328617625725759571144780212268133978522706711834706712800
825351461273674974066617311929682421617092503555733685276673

多項式の基礎:

G = 26eb7a8 59923fbc 82189631 f8103fe4 ac9ca297 0012d5d4
60248048 01841ca4 43709584 93b205e6 47da304d b4ceb08c

bbd1ba39 494776fb 988b4717 4dca88c7 e2945283 a01c8972

G = 349dc80 7f4fbf37 4f4aeade 3bca9531 4dd58cec 9f307a54 y

ffc61efc 006d8a2c 9d4979c0 ac44aea7 4fbebbb9 f772aedc

b620b01a 7ba7af1b 320430c8 591984f6 01cd4c14 3ef1c7a3

通常の基準:

G = 04bb2db a418d0db 107adae0 03427e5d 7cc139ac b465e593

4f0bea2a b2f3622b c29b3d5b 9aa7a1fd fd5d8be6 6057c100

8e71e484 bcd98f22 bf847642 37673674 29ef2ec5 bc3ebcf7

G = 44cbb57 de20788d 2c952d7b 56cf39bd 3e89b189 84bd124e

751ceff4 369dd8da c6a59e6e 745df44d 8220ce22 aa2c852c

fcbbef49 ebaa98bd 2483e331 80e04286 feaa2530 50caff60

D.1.3.5.2 曲線 B-571

n = 3864537523017258344695351890931987344298927329706434998

657235251451519142289560424536143999389415773083133881 12

192694448624687246281681307023452828830333241139319110 52

85703

多項式の基礎:

b = 2f40e7e 2221f295 de297117 b7f3d62f 5c6a97ff cb8ceff1

cd6ba8ce 4a9a18ad 84ffabbd 8efa5933 2be7ad67 56a66e29

4afd185a 78ff12aa 520e4de7 39baca0c 7ffeff7f 2955727a

G = 303001d 34b85629 6c16c0d4 0d3cd775 0a93d1d2 955fa80a

a5f40fc8 db7b2abd bde53950 f4c0d293 cdd711a3 5b67fb14

99ae6003 8614f139 4abfa3b4 c850d927 e1e7769c 8eec2d19

G = 37bf273 42da639b 6dccfffe b73d69d7 8c6c27a6 009cbbca y

1980f853 3921e8a6 84423e43 bab08a57 6291af8f 461bb2a8

b3531d2f 0485c19b 16e2f151 6e23dd3c 1a4827af 1b8ac15b

通常の基準:

シード= 2aa058f7 3a0e33ab 486b0f61 0410c53a 7f132310

b = 3762d0d 47116006 179da356 88eeaccf 591a5cde a7500011
8d9608c5 9132d434 26101a1d fb377411 5f586623 f75f0000 1ce61198 3c1275fa 31f5bc9f
4be1a0f4 67f01ca8 85c74777

G_{x} = 0735e03 5def5925 cc33173e b2a8ce77 67522b46 6d278b65 0a291612 7dfea9d2 d361089f
0a7a0247 a184e1c7 0d417866 e0fe0feb 0ff8f2f3 f9176418 f9 7d117e 624e2015 df1662a8

G_{y} = 04a3642 0572616c df7e606f ccadaecf c3b76dab 0eb1248d d03fbdfc 9cd3242c 4726be57
9855e812 de7ec5c5 00b4576a 24628048 b6a72d88 0062eed0 dd34b1 09 6d3acbb6
b01a4a97

D.2 モジュラー演算の実装

上記の例の素数係数は特殊なタイプ (一般化メルセンヌ数と呼ばれる)であり、剰余乗算を一般よりも効率的に実行できます。
このセクションでは、例に示されている素数係数ごとに、この高速演算を実装するためのルールを示します。

2 つの整数を掛ける (mod m) ための通常の方法は、整数の積をとり、それを減らすことです (mod m)。したがって、次の問題があります: mより小さい整数Aが与えられた場合 ^{2}、計算する

B = A mod m。

一般に、整数の除算の余りとしてBを取得する必要があります。ただし、 mが一般化メルセンヌ数の場合、 B は少数の項の和または差 (mod m) として表すことができます。この式を計算するには、整数の合計または差を評価し、

結果は法mを削減しました。後者の縮小は、m のコピーをいくつか加算または減算することによって実現できます。

5 つの曲線例のそれぞれの素数係数p は、一般化されたメルセンヌ数です。

D.2.1 曲線 P-192

この曲線の係数はp = 2です。 ^{192} ^{64} – 2 – 1. pより小さいすべての整数A ^{2}は次のように書くことができます

$$A = A_5 \, 2^{320} + A_4 \, 2^{256} + A_3 \, 2^{192} + A_2 \, 2^{128} + A_1 \, 2^{64} + A_0,$$

ここで、各Ai は64 ビットの整数です。 64 ビット ワードの連結として、これは次のように表すことができます。

A = (A5 || A4 || A3 || A2 || A0)。

Bの式は次のとおりです。

B = T + S1 + S2 + S3 mod p、

ここで、192 ビット項は次のように与えられます。

T = (A2 || A1 || A0)

S1 = (A3 || A3)

S2 = (A4 || A4 || 0)

S3 = (A5 || A5 || A5)。

D.2.2 曲線 P-224

この曲線の係数は 2 p = ー + pより小さいすべての整数A 224 96 1. 2 次のように書くことができます:

AA 13 2 416 あ 12 2 384 A 2 11 =2 352 + あ 10 2 + 320 + あ 9 288 あ 8 2 256 A 2 7 + 224 + あ 6 + 2 192

あ 5 2 2 160 + あ 4 + 128 + あ 3 + 2 96 あ 2 2 64 あ 1 2 32 あ 0 、

ここで、各Ai は32 ビットの整数です。 32 ビット ワードの連結として、これは次のように表すことができます。

A = (A13 || A12 || ⋯|| A0)。

B の式は次のとおりです。

B = T + S 1 + S 2 – D1 – D2 mod p、

ここで、224 ビットの項は次の式で与えられます。

T = (A6 || A5 || A4 || A3 || A2 || A1 || A0)

S1 = (A10 || A9 || A8 || A7 || 0 || 0 || 0)

S2 = (0 || A13 || A12 || A11 || 0 || 0 || 0)

D1 = (A13 || A12 || A11 || A10 || A9 || A8 || A7)

D2 = (0 || 0 || 0 || 0 || A13 || A12 || A11)。

D.2.3 曲線 P-256

この曲線の係数は、 p = 2256 – 2224 + 2192 + 296 – 1 です。pより小さいすべての整数A 2 できる

次のように書かれます:

AA + 15 2 480 A 2 14 = 448 2 + あ 13 416 + あ 12 + 2 384 あ 11 2 352 あ 10 2 2 320 + あ 9 288 + あ 8 2 256 +

あ 7 2 224 + あ 6 + 2 192 + あ 5 2 + 160 + あ 4 2 128 あ 3 2 96 あ 2 2 64 あ 1 2 32 あ 0 、

ここで、それぞれのA は 32 ビット整数です。 32 ビット ワードの連結として、これは次のように表すことができます。

A = (A15 || A14 || || A0)。

Bの式は次のとおりです。

$$B = T + 2S1 + 2S2 + S3 + S4 - D1 - D2 - D3 - D4 \bmod p、$$

ここで、256 ビット項は次の式で与えられます。

T = (A7 || A6 || A5 || A4 || A3 || A2 || A1 || A0)

S1 = (A15 || A14 || A13 || A12 || A11 || 0 || 0 || 0)

S2 = (0 || A15 || A14 || A13 || A12 || 0 || 0 || 0)

S3 = (A15 || A14 || 0 || 0 || 0 || A10 || A9 || A8)

S4 = (A8 || A13 || A15 || A14 || A13 || A11 || A10 || A9)

D1 = (A10 || A8 || 0 || 0 || 0 || A13 || A12 || A11)

D2 = (A11 || A9 || 0 || 0 || A15 || A14 || A13 || A12)

D3 = (A12 || 0 || A10 || A9 || A8 || A15 || A14 || A13)

D4 = (A13 || 0 || A11 || A10 || A9 || 0 || A15 || A14)

D.2.4 曲線 P-384

この曲線の係数はp = 2 で、次のように記述 $^{384} - 2^{128} - 2^{96} + 2^{32} - 1$. pより小さいすべての整数A ^{2} できる できます。

$$AA + {}_{23} + 2^{736} \; 22 \; 2^{704} \; あ_{21} \; 2^{672} \; あ_{20} \; 2^{640} \; A\,2_{19} = 2^{608} + あ_{18}{}^{576} + あ_{17} + 2\,2^{544} \; あ_{16}{}^{512}$$

$$あ_{15} + 2^{480} \; A\,2_{14}{}^{448} \; 2 + あ_{13}{}^{416} + あ_{12} + 2^{384} \; あ_{11} \; 2^{352} \; あ_{10} \; 2\,2^{320} + あ_{9}{}^{288} + あ_{8} + 2^{256}$$

$$あ_{7} \; 2^{224} + あ_{6} + 2^{192} + あ_{5} \; 2^{160} + あ_{4} + 2^{128} \; あ_{3} \; 2^{96} \; あ_{2} \; 2^{64} \; あ_{1} \; 2^{32} \; あ_{0}、$$

ここで、それぞれのA 、は 32 ビット整数です。 32 ビット ワードの連結として、これは次のように表すことができます。

$$A = (A23 \,||\, A22 \,||\, \quad\quad \,||\, A0)。$$

Bの式は次のとおりです。

$$B = T + 2S1 + S2 + S3 + S4 + S5 + S6 - D1 - D2 - D3 \bmod p、$$

ここで、384 ビット項は次の式で与えられます。

T = (A11 || A10 || A9 || A8 || A7 || A6 || A5 || A4 || A3 || A2 || A1 || A0)

S1 = (0 || 0 || 0 || 0 || 0 || A23 || A22 || A21 || 0 || 0 || 0 || 0)

S2 = (A23 || A22 || A21 || A20 || A19 || A18 || A17 || A16 || A15 || A14 || A13 || A12)

S3 = (A20 || A19 || A18 || A17 || A16 || A15 || A14 || A13 || A12 || A23|| A22|| A21)

S4 = (A19 || A18 || A17 || A16 || A15 || A14 || A13 || A12 || A20 || 0 || A23 || 0)

S5 = (0 || 0 || 0 || 0 || A23 || A22 || A21 || A20 || 0 || 0 || 0 || 0)

S6 = (0 || 0 || 0 || 0 || 0 || 0 || A23 || A22 || A21 || 0 || 0 || A20)

D1 = (A22 || A21 || A20 || A19 || A18 || A17 || A16 || A15 || A14 || A13 || A12 || A23)

D2 = (0 || 0 || 0 || 0 || 0 || 0 || 0 || A23 || A22 || A21 || A20 || 0)

D3 = (0 || 0 || 0 || 0 || 0 || 0 || 0 || A23 || A23 || 0 || 0 || 0)。

D.2.5 曲線 P-521

この曲線の係数はp = 2です。 ^{521} – 1. pより小さいすべての整数A ^{2} 書くことができます

$$A = A1 \quad 2521 + A0、$$

ここで、それぞれのA 。 は 521 ビットの整数です。これは 521 ビット ワードの連結として次のように表すことができます。

$$A = (A1 \,||\, A0)。$$

Bの式は次のとおりです。

$$B = (A0 + A1) \bmod p.$$

D.3 通常の塩基

GF(2m)の要素は、あるTについて、GF(2m)のタイプT正規基底5 Bで表現されます。各要素は、ビット文字列として一意の表現を持ちます。

$$(a0\ a1 \cdots am–1)。$$

算術演算は次のように実行されます。

加算: 2 つの要素の加算は、2 を法とするビット単位の加算によって実装されます。したがって、たとえば、次のようになります。

$$(1100111) + (1010010) = (0110101)。$$

二乗法: if

$$\alpha = (a0\ a1 \cdots am–1)$$

それから

$$\alpha^2 = (am–1\ a0\ a1 \cdots am–2)。$$

乗算:乗算を実行するには、入力に対して関数F(u,v)が構築されます。 _ _

[5] これがこの標準で考慮される唯一のケースであるため、このセクションではmが奇数、 Tが偶数であると仮定します。

u = (u0 u1 … um–1) およびv = (v0 v1 … vm–1)

次のように。

1.pを設定する ← Tm +1。

2. u をp を法とする次数Tを持つ整数とします。

3. シーケンスF (1)、 F (2)、 … 、F (p–1)を次のように計算します。

3.1設定 ← 1.

3.2 jが0 からT–1 までの場合、次のことを行います。

3.2.1 nを設定する ← w。

3.2.2 i = 0 ～ m–1 の場合、次のようにします。

3.2.2.1 F(n)の設定 ← 私。

3.2.2.2 nを設定する ← 2n mod p.

3.2.3 設定w ← うわーmod p.

4. 式を出力します。

$$F\,uv(,):\quad =\ \sum_{k=1}^{p-2} \text{紫外線}\ F\,k(F\,pk)$$

この計算は、ベースごとに 1 回だけ実行する必要があります。

Bに対して関数F を与えると、積は

(c0 c1 … cm–1) = (a0 a1 … am–1) * (b0 b1 … bm–1)

は次のように計算されます。

1. (u0 u1 … um–1) ← (a0 a1 . . . am–1) を設定します。

2. (v0 v1 … vm–1) ← (b0 b1 . . . bm–1) を設定します。

3. k = 0 ～ m – 1 の場合、次のようにします。

3.1 コンピューティング

ck = F(u, v)。

3.2 u ← LeftShift (u)およびv ← LeftShift (v) を設定します。ここで、 LeftShift は循環を示します。
左シフト操作。

4. 出力c = (c0 c1 … cm–1)。

例: GF(27)のタイプ 4 正規基底の場合、p = 29 およびu = 12 または 17。したがって、 Fの値は次のようになります。
は次のように与えられます。

F (1) = 0 F (8) = 3 F (15) = 6 F (22) = 5

F (2)=1 F (9)=3 F (16)=4 F (23)=6

F (3) = 5 F (10) = 2 F (17) = 0 F (24) = 1

F (4) = 2 F (11) = 4 F (18) = 4 F (25) = 2

F (5) = 1 F (12) = 0 F (19) = 2 F (26) = 5

F (6) = 6 F (13) = 4 F (20) = 3 F (27) = 1

F (7) = 5 F (14) = 6 F (21) = 3 F (28) = 0

したがって、

$$F(u, v) = u0\,v1 + u1\,(v0 + v2 + v5 + v6) + u2\,(v1 + v3 + v4 + v5) + u3\,(v2 + v5) +$$
$$u4\,(v2 + v6) + u5\,(v1 + v2 + v3 + v6) + u6\,(v1 + v4 + v5 + v6)。$$

したがって、もし

$$a = (1\,0\,1\,0\,1\,1\,1) \text{ および} b = (1\,1\,0\,0\,0\,0\,1)、$$

それから

$$c0 = F((1\,0\,1\,0\,1\,1\,1), (1\,1\,0\,0\,0\,0\,1)) = 1,$$
$$c1 = F((0\,1\,0\,1\,1\,1\,1), (1\,0\,0\,0\,0\,1\,1)) = 0、$$

$$c6 = F((1\,1\,0\,1\,0\,1\,1), (1\,1\,1\,0\,0\,0\,0)) = 1,$$

c = ab = (1 0 1 1 0 0 1)となります。

D.4 コブリッツ曲線でのスカラー乗算

このセクションでは、GF(2m)上のコブリッツ曲線Ea上のスカラー倍数nPを計算する特に効率的な方法について説明します。

操作 τ は次のように定義されます。

$$\tau(x, y) = (x^2、y^2)。$$

通常の基底表現が使用される場合、演算はxとyを表すビット列上で右循環シフト を実行することで実装されます
トします。

mとaを指定して、次のパラメータを定義します。

- Cは 5 より大きい整数です。

• μ = (–1)1–a 。

• i = 0 およびi = 1 の場合、シーケンスsi(m)を次のように定義します。

$$si(0) = 0、\ si(1) = 1 - i、\ \bullet$$

$$si(m) = \mu\ si(m-1) - 2\,si(m-2) + (-1)i$$

• シーケンスV(m)を定義します。

$$V(0) = 2、\ V(1) = \mu$$

$$V(m) = \mu\ \bullet v(m-1) - 2V(m-2)。$$

曲線の例では、量si(m)とV(m)は次のとおりです。

曲線 K-163:

s0(163) = 25793864391107316504195 37

s1(163) = –75536006447622637546 1594

V(163) = –48454666325394107768043 17

曲線 K-233:

s0(233) = –27859711741434429761757834964435883

s1(233) = –44192136247082304936052160908934886

V(233) = –13738154601110823539498729965136 6779

曲線 K-283:

s0(283) = –66598153210904904110879553600159146928 0025

s1(283) = 11558600549091367751922810725916099139459 68

V(283) = 77772448708728309992877919709628239775699 17

曲線 K-409:

s0(409) = –18307510456002382137810317198756461378590542487556869338419259

s1(409) = –88930485261383040971966532418442126796265661009966064448167 90

V(409)= 10457288737315625927447685387048320737638796957687575791173829

曲線 K-571:

s0(571) = –37373194468764636924293858924761155671472939645961310241234064 20\
23524191672998326 1305

s1(571) = –31918577064464160995838145959489596741319689121485646586105651 17\
58982848515832612248752

V(571)= –14838092698169141389961914029705149036454257418049393623291233951\
342085168289731114598 43

次のアルゴリズムは、GF(2m)上のコブリッツ曲線Ea上のスカラー倍数nPを計算します。
楕円加算および減算の平均回数は最大で ～ 1 + (m/3)、最大で ～
m/3 (確率は少なくとも 1 – 25–C) 。

1. i = 0 ～ 1 の場合、次のようにします。

1.1n _ 』← n / 2a–C + (m–9) / 2 .

1.2 g ← si(m) ・ n ← 』。

1.3時間 』g / 2m 』

1.4 j ← V(m)•h 』。

1.5 l ← 丸め((g + j/ 2C 』 』) / 2(m+5) / 2)。

1.6 λ 私←私 』 。

1.7 fi ← ラウンド(i)。 λ

1.8 η 私は← λ 私–フィ ..

1.9ひ←0。

2. η ←2 η _ +0 ημ1.

3. もし (η ≥ 1)、

それから

もし (η – 3 μη 1 < -1)

次にh1を設定します← μ

それ以外の場合はh0 ← 1を設定します。

それ以外

もし (η 0 +4 – μη 1 ≧ 2)

次にh1を設定します← μ。

4. もしも (η < -1)

それから

if (0η – 3 ημ1 ≥ 1)

次にh1 ← –を設定します μ

それ以外の場合は、 h0 ← -1を設定します。

それ以外

もし (η 0 +4 – μη 1 < –2)

次にh1 ← –を設定します μ。

5. q0 ← f0 + h0。

6. q1 ← f1 + h1。

7. r0 ← n – (s0 + μ s1) q0 – 2s1 q1。

8. r1 ← s1 q0 – s0 q1。

9. Q ← Oを設定します。

10.P0 ← P. _

11. 一方((r0 ≠ 0) または(r1 ≠ 0))

11.1 (r0 が奇数) の場合、

11.1.1 u ← 2 – (r0 – 2 r1 mod 4) を設定します。

11.1.2 r0 ← r0 – u を設定します。

11.1.3 (u = 1) の場合、 Q ← Q + P0 を設定します。

11.1.4 (u = –1) の場合、 Q ← Q – P0 を設定します。

11.2 P0 ← P0 を設定します。

11.3 セット(r0 , r1) ← (r1 + μ r0 /2、 – r0 /2) 。

終わりながら

12. 出力Q。

D.5 擬似ランダム曲線の生成 (プライムケース)

pのビット長をlとし、次のように定義します。

$$v = (l - 1)/160$$

$$w = l - 160v - 1.$$

1.ドメイン パラメータ シードとして任意の 160 ビット文字列sを選択します。

2. h = SHA-1(s) を計算します。

3. h0を hの右端のwビットを取ることによって得られるビット列とする。

4. zを整数とし、そのバイナリ展開は 160 ビット文字列s によって与えられます。

5. i が1 からvの場合、次のようにします。

5.1 160 ビット文字列si を整数(z + i) mod (2 160) のバイナリ展開として定義します。

5.2 hi = SHA-1(si) を計算します。

6.次のようにh0 、 h1 、 …、 hvを連結して得られるビット文字列をhとします。

h = h0 || h1 || …||なるほど。

7. cを整数とし、そのバイナリ展開はビット文字列h によって与えられます。

8. ((c = 0 または4c + 27 ≡ 0 (mod p))) の場合、ステップ 1 に進みます。

9. c b2 ≡ aとなるような整数a、 b ∈GF(p)を選択しま

す。 ^{3} (mod p)。

(最も単純な選択は、 a = cおよびb = c です。ただし、パフォーマンス上の理由から、別の選択が必要な場合もあります。)

10. yによって与えられるGF(p)上の楕円曲線Eを確認します。 ^{2} 合 3 + ax + bには適切な順序があります。 = xの場

そうでない場合は、ステップ 1 に進みます。

D.6 曲線の擬似ランダム性の検証 (プライムケース)

160 ビットのドメイン パラメーター シード値 s が与えられた場合、次のように係数b が暗号化ハッシュ関数 SHA-1 を介してsから取得されたことを検証します。

pのビット長をlとし、次のように定義します。

v = (l – 1) /160

w = l – 160v – 1.

1. h = SHA-1(s) を計算します。

2. h0を hの右端のwビットを取ることによって得られるビット列とします。

3. zを整数とし、そのバイナリ展開は 160 ビット文字列s によって与えられます。

4. i = 1 からvの場合は次のようにします

4.1 160 ビット文字列si を整数の 2 進展開として定義する

(z + i) mod (2160)。

4.2 hi = SHA-1(si) を計算します。

5. hを、次のようにh0 、 h1 、 …、 hvを連結して得られるビット文字列とします。

h = h0 || h1 || …||なるほど。

6. c を整数とし、そのバイナリ展開はビット文字列 h によって与えられます。

7. b2 c ≡ –27 (mod p) であることを確認します。

D.7 擬似ランダム曲線の生成 (バイナリの場合)

させて：

$$v = (m - 1)/B$$

$$w = m - Bv.$$

1. ドメイン パラメータ シードとして任意の 160 ビット文字列 s を選択します。

2. h = SHA-1(s) を計算します。

3. h0 を、h の右端の w ビットを取ることによって得られるビット列とします。

4. z を整数とし、そのバイナリ展開は 160 ビット文字列 s によって与えられます。

5. i が 1 から v の場合、次のようにします。

5.1 160 ビット文字列 si を整数 (z + i) mod (2160) のバイナリ展開として定義します。

5.2 hi = SHA-1(si) を計算します。

6. 次のように h0 、 h1 、…、 hv を連結して得られるビット文字列を h とします。

$$h = h0 \,||\, h1 \,||\, \cdots ||\text{なるほど。}$$

7. b を GF(2m) の要素とし、その 2 進展開はビット文字列 h によって与えられます。

8. GF(2m) の要素 a を選択します。

9. GF(2m) 上の楕円曲線 E が $y^2 + xy = x^3 2 + b$ によって与えられることを確認します。 に は適切な + 斧があります

注文。そうでない場合は、ステップ 1 に進みます。

D.8 曲線の擬似乱数性の検証 (バイナリの場合)

160 ビットのドメイン パラメーター シード値 s が与えられた場合、次のように係数 b が暗号化ハッシュ関数 SHA-1 を介して s から取得されたことを検証します。

定義する

$$v = (m - 1)/160$$

$$w = m - 160v$$

1. h = SHA-1(s) を計算します。

2. h0 を h の右端の w ビットを取ることによって得られるビット列とします。

3. z を整数とし、そのバイナリ展開は 160 ビット文字列 s によって与えられます。

4. i = 1 から v の場合は次のようにします

4.1 160 ビット文字列 si を整数 (z + i) mod (2160) のバイナリ展開として定義します。

4.2 hi = SHA-1(si) を計算します。

5. h を、次のように h0 、 h1 、…、 hv を連結して得られるビット文字列とします。

h = h0 || h1 || …||なるほど。

6. c を、ビット文字列 h で表される GF(2m) の要素とします。

7. c = b であることを確認します。

D.9 多項式基底から正規基底への変換

それが体 GF(2m)の要素であると仮定します。 p を、与えられた多項式基底に関して表すビット文字列とするα。与えられた正規基底を表すビット列である n を計算することが望まれます。これは行列計算によって行われます α

p Γ = n、

ここで、Γ は GF(2) のエントリを持つ m 行 m 列の行列です。基数のみに依存する行列 Γ は、最後から 2 番目の行が与えられると簡単に計算できます。各変換の最後から 2 番目の行は以下のようになります。

~~163 度:~~

3 e173bfaf 3a86434d 883a2918 a489ddbd 69fe84e1

~~度数 233:~~

0be 19b89595 28bbc490 038f4bc4 da8bdfc1 ca36bb05 853fd0ed
0ae200ce

~~学位283:~~

3347f17 521fdabc 62ec1551 acf156fb 0bceb855 f174d4c1 7807511c
9f745382 add53bc3

~~学位409:~~

0eb00f2 ea95fd6c 64024e7f 0b68b81f 5ff8a467 acc2b4c3 b9372843
6265c7ff a06d896c ae3a7e31 e295ec30 3eb9f769 de78bef5

~~学位571:~~

7940ffa ef996513 4d59dcbf e5bf239b e4fe4b41 05959c5d 4d942ffd 46ea35f3 e3cdb0e1 04a2aa01
cef30a3a 49478011 196bfb43 c55091b6 1174d7c0 8d0cd d61 3bf6748a bad972a4

Γ の最後から 2 番目の行 r が与えられると、行列の残りの部分は次のように計算されます。正規基底に関する表現pが r である GF(2m) の要素を とします。 Γ の行は上から下に、要素を表すビット列になります。

$$\beta^{m-1}、\beta^{m-2}、\cdots、\beta^{2}、\beta、1$$

通常の基礎に関して。 (要素 1 はすべて 1 のビット列で表されることに注意してください。)

あるいは、行列は付録 D.10 で説明されている行列の逆行列です。

これらの計算の詳細については、IEEE 標準 1363-2000 の付録 A.7 を参照してください。
標準。

D.10 正規基底から多項式基底への変換

字列とします。所 α は体 GF(2m) の要素です。 n を、与えられた正規基底を基準とする仮定を表すビット文 α と
与の多項式基底を表すビット列であるpを計算することが望ましい。これは行列計算によって行われます α と

$$n\Gamma = p、$$

ここで、Γ は GF(2) のエントリを持つ m 行 m 列の行列です。基底のみに依存する行列 Γ は、最上位行が与えられると簡単に計算できます。各変換の一番上の行を以下に示します。

163 度:

7 15169c10 9c612e39 0d347c74 8342bcd3 b02a0bef

度数 233:

149 9e398ac5 d79e3685 59b35ca4 9bb7305d a6c0390b cf9e2300
253203c9

学位283:

31e0ed7 91c3282d c5624a72 0818049d 053e8c7a b8663792 bc1d792e ba9867fc 7b317a99

学位409:

0dfa06b e206aa97 b7a41fff b9b0c55f 8f048062 fbe8381b 4248adf9
2912ccc8 e3f91a24 e1cfb395 0532b988 971c2304 2e85708d

学位571:

452186b bf5840a0 bcf8c9f0 2a54efa0 4e813b43 c3d41496 06c4d27b
487bf107 393c8907 f79d9778 beb35ee8 7467d328 8274caeb da6ce05a
eb4ca5cf 3c3044bd 4372232f 2c1a27c4

Γ の最上位行 r が与えられると、行列の残りの部分は次のように計算されます。多項式基底に関す β ～の要素になる
る表現を r とする GF(2 m) とします。 Γ の行は上から下に、要素を表すビット列になります。

$$\beta\beta^{2}、\beta^{22}、\cdots、\beta^{2m-1}$$

多項式基底に関して。

あるいは、行列は付録 D.9 で説明されている行列の逆行列です。

これらの計算の詳細については、IEEE Std 1363-2000 の付録 A.7 を参照してください。
標準。

付録 E: DSA における v = r の証明

(参考)

この付録の目的は、署名検証において M= M、r= r、s= s の場合、v = r であることを示すことです。 Hash を承認されたハッシュ関数とします。次の結果が必要です。

補題: p と q を q が (p – 1) を割る素数とし、h を正の整数から mod p を引いた値とする。 (gq mod p) = 1、および if (m mod p よりも大きいとし、g = (h $^{(p-1)/q}$ q) = (n mod q)、その後 (g ^{m} mod p) = (g ^{n} mod p)。

証拠：

gq mod p = (h(p–1) / q mod p) q mod p

= h(p – 1) mod p

= 1

フェルマーの小定理による。ここで、(m mod q) = (n mod q)、つまり、ある整数 k に対して m = (n + kq) とします。それから

gm mod p = gn = + kq mod p

(gn gkq) mod p

= ((gn mod p) (gq mod p) ^{k}) mod p

= gn mod p、

(gq mod p) = 1 であるため。

主な結果の証明:

定理: 署名検証で M= r、s= M の場合、v = r 。

証拠：

w = (s') $^{-1}$ mod q = s–1 mod q

u1 = ((ハッシュ(M '))w) mod q = ((Hash(M))w) mod q

u2 = ((r')w) mod q = (rw) mod q。

ここで y = (g mod p)、補題により、

v = ((gu1 y ^{u2}) mod p) mod q

(M)w = ((g ^{ハッシュ} y rw) mod p) mod q

= ((g $^{ハッシュ(M)w}$ gxrw) mod p) mod q

= ((g(ハッシュ(M) + xr)w) mod p) mod q。

また：

$$s = (k^{-1} (\text{ハッシュ}(M) + xr)) \bmod q.$$

したがって、次のようになります。

$$w = (k (\text{ハッシュ}(M) + xr)^{-1}) \bmod q$$

$$(\text{ハッシュ}(M) + xr)w \bmod q = k \bmod q_{\circ}$$

したがって、補題により次のようになります。

$$v = (g^{k} \bmod p) \bmod q = r$$

付録 F: 必要なテストのラウンド数の計算

Miller-Rabin 確率的素数検定の使用

(参考)

F.1 Miller-Rabin 素数検定の必要なラウンド数

論文 [1] のアイデアは、t ラウンドの Miller-Rabin (MR) テストを通過する奇数の k ビット整数が実際に合成である確率である pk,t を推定するために適用されました。確率 pk は、t ラウンドの MR テスト (ランダムに生成された基数を使用) に合格すると期待できるバイナリ長 k の奇数の合成数の は

数と、その値と奇数の素数の数の合計との比として理解されます。バイナリの長さ k。これは、テストのために選択される候補が、奇数の k ビット整数のセット全体から一様にランダムに選択されると仮定するのと同じです。 Pomerance らに従って、pk,t は、t ラウンドの MR テスト (ランダムに生成された基底を使用) に合格するバイナリ長 k の奇数合成数の予想数と、その総数の比率によって (過剰に) 推定できます。バイナリ長 k の奇数素数。

k ビット素数を生成する責任を負った当事者の観点から見ると、目的は、p が次のような t の値を決定することです。

kt 許容できる小さな目標以下である

値p 引数 。

[1] を使用すると、pk の上限を計算でき、t の上限は k と ptarget (誤って合成数を生成する最 k と t の関数として。これから、

大許容確率) の関数として計算できます。以下は t を計算するためのアルゴリズムです。

1. t = 1, 2 … 2 k − 1 … –log2(ptarget)/2

1.1 M = 3、4の場合— 1の場合 (1)

1.1.1 p を計算する kt (2)のように。

1.1.2 pk,t ≤ ptarget の場合

1.1.2.1 受け入れます。

1.1.2.2 停止します。

(1) において、k は候補素数のビット長であり、(2) は次のとおりです。

$$p_{kt} = 2.00743 \ln(2)\, 2^{2k - kk\text{ マウント} + \frac{8(\pi^2 - 6)}{3} 2^{k-2} \sum_{\cdots=3}^{M} 2^{mmt} \cdots 1) \sum_{j=2}^{\cdots} \frac{1}{2^{j + \frac{(k-1)}{j}}}} \quad (2)$$

この式を t に使用して、DSA および RSA 候補素数をテストするために次の方法が使用されます。

F.2 DSA プライムの生成

DSA の場合、ドメイン パラメーターに使用される素数 p および q を生成する際には、最大限の注意を払う必要があります。同じ素数 p と q が多くの当事者によって使用されます。これは、これらの番号が持つ可能性のある弱点が複数のユーザーに影響を与える可能性があることを意味します。これは、素数があまり頻繁に生成されないことも意味します。通常、システム全体は、長期間にわたって同じドメイン パラメータのセットを使用します。したがって、この場合は追加の注意が必要です。

これを念頭に置くと、単純に候補素数を t ラウンドの MR テストにかけるのは楽観的すぎるかもしれません。t の最小許容値は付録 F.1 の (1) および (2) に従って決定されます。これは、たとえば、素数性テストの候補を選択するプロセス中に [1] で行われた仮定が満たされたことを疑う理由がある場合に当てはまります。MR テストで生き残った候補に対して追加の (異なる) 素数テストを実行することで、プロセスに対する信頼性をさらに高めることができます。別のオプションとして、追加の MR テストを実行することももちろん可能です。これらの考慮事項により、次の代替案が導き出されます。(A) 付録 F.1 の (1) および (2) に従って決定された MR テストのラウンド数を使用し、それに続いて (ANS X9 で推奨されているように) 1 回のルーカス テストを行う。 .31)、または (B) t を決定する際に (はるかに) より保守的なアプローチを使用し (たとえば、以下に説明するように)、候補素数を追加ラウンドの MR テストに供します。

戦略 (B) の 1 つのアプローチは、大多数のシステム ユーザーの視点を採用することです。システム ユーザーは (想定される) 素数の生成には関与していませんが、セキュリティのためにその素数に依存しなければなりません。そのような関係者は、MR 検査の候補が、付録 F.1 の (1) および (2) に従って t を決定する際に想定された一様分布から大きく逸脱する方法で選択されたことを懸念する可能性があります。選択プロセスに何らかの異常な偏りがある可能性がある場合、合成数がテストに耐えられる確率を最小限に抑えることが重要です。任意の k ビットの奇数合成数 (選択方法に関係なく) について、ランダムに選択された塩基を使用した MR テストの t ラウンドに合格する確率は 4 未満であることがわかります (ただし、これは特に厳しい限界ではありません)。 ） 。 $4^{-t} \leq ptarget$ となるように t を選択することは、$t \geq -\log_2(ptarget)/2$ を選択することと同じです。合成数が MR テストを生き残る確率が ptarget 以下であることを保証するために、ラウンド数を $t = \lceil -\log_2(ptarget)/2 \rceil$ に設定できます。たとえ候補者を選択する方法が非常に偏っていて、テストに合成数しか提供されなかったとしても、次のことは合理的です。

合成数が t ラウンドの MR テスト プロセスを通過するまでに、少なくとも 1/ ptarget 回の試行 (これは 4t を超えます) がかかることが予想されます。

警告: 上記の説明で示されているように、MR テストの推奨ラウンド数に関連して「エラーの確率」という表現を使用する場合は注意が必要です。合成数がミラーラビン検定の t 回存続する確率は、次の確率と同じではありません。

する これは、ミラー•ラビン検定の t 回の生存数が pk,t 、複合である確率です。通常、素数の生成を担当
当事者にとっては後者の確率が最も重要なはずですが、他の誰かが生成した数値の素数性を検証する責任を負う
当事者にとっては前者の確率の方が重要である可能性があります。ただし、十分に大きい k (たとえば、
k ≥ 51) の場合、付録 F.1 の式 (2) を取得するために行われたのと同じ候補の選択に関する仮定の下で、pk,t ≤
4–t であることが示されます ([1] を参照してください。) このような場合、ptarget が、生成プロセスでは、合
成数と、その素数性を検証する試行後に合成数が存続する確率が生成されます。

付録 C.3 の表 C.1 は、上記の戦略 (A) または (B) のいずれかを使用して DSA の素数 p および q を生成する場合
の t の最小値を示しています。 「MR 検査のみ」というタイトルの列に示されている t 値を取得するには、保守
的な戦略 (B) に従いました。これらの t 値は、p と q の素数性を検証するのに十分です。 「1 つのルーカス テスト
が続く場合の MR テスト」というタイトルの列に示されている t 値は、計算 (1) および (2) を使用した次の戦略
(A) の結果です。

付録 F.1 に記載されています。

F.3 RSA署名の素数の生成

RSA 署名アルゴリズムの素数を生成する場合、MR テスト手順でのエラーの確率を減らすことが依然として非常に
重要です。ただし、(可能性のある) 素数はユーザーのキー ペアの生成に使用されるため、テスト プロセスで合
成数が生き残った場合、エラーの影響は DSA ドメイン パラメーターを生成する場合ほど劇的ではない可
能性があります。ユーザーのドメインではなく、1 人のユーザーのトランザクションのみが影響を受けます。さ
らに、あるユーザーに対して生成された p 値または q 値が複合値である場合、そのユーザーによって生成された署名
が検証できないことがほぼ確実であるため、問題は長期間発見されないことはありません。

したがって、RSA 素数 p および q を生成するときは、ラウンド数を使用するだけで十分です。
実行する MR 検査の最小数として、付録 F.1 の (1) および (2) から導出されます。
ただし、pk の定義の場合、p をテストするときに t が十分に保守的であるとはみなされません。
および q、t ラウンドの Miller-Rabin テストの後に 1 回の Lucas テストを行うことが推奨されます。
テスト。

RSA 署名アルゴリズムでの使用に推奨される p および q の長さは、512、1024、および 1536 ビットです。 n = pq
であるため、n に対応する長さは 1024、2048、3072 ビットであることを思い出してください。
それぞれ。現在 SP 800-57、パート 1 で指定されているように、これらの長さはそれぞれ 80、112、および 128 ビ
ットのセキュリティ強度に対応します。したがって、ミラーラビン テストのラウンド数をターゲットのエラー確率
値 2 〜 80、2 〜 112、および 2 〜 128 に一致させることは理にかなっています。この確率は過去によく使用され
ており、多くのアプリケーションで許容される可能性があるため、すべての素数の長さに 2 〜 100 の確率が含まれ
ます。

条件付きで RSA 素数 p および q を生成する場合、値 t を使用するだけで十分です。
生成時に実行する MR テストの最小数として (1) と (2) から導出されます。

補助素数 p1、p2、q1、q2。これらの数値に対して追加のルーカス テストを使用する必要はありません。非常にまれに、数値 p1、p2、q1、または q2 のいずれかが合成である場合でも、対応する RSA 素数 (p または q) が必須条件を満たす可能性が依然として高くなります。

と q2 のサイズは重要ではあるが 1 p 、 p2 、 q1 のリソースを持っている敵に対して、Lenstra の楕円曲線因数分解 [2] は確実に適用されるように選択されました (これに対しては、大きな p と q を選択する以外に防御策はありません)。は、Pollard P-1 法 [2]、Williams P+1 法 [3]、またはさまざまなサイクリング法 [2] よりも効果的な因数分解アルゴリズムです。圧倒的なリソースを持つ敵にとって、最良の万能因数分解アルゴリズムは一般数体ふるい [2] であると想定されます。

付録 C.3 の表 C.2 および C.3 は、RSA 署名鍵ペアの構築に使用される素数を生成する際の MR テストの最小ラウンド数を指定します。

付録 G: 参考資料

[1] I. Damgard、P. Landrock、および C. Pomerance、C.「
Strong Probable Prime Test」、Mathematics of Computation、v. 61、No、203、pp. 177-194、1993。

[2] AJ メネゼス、PC オールショット、SA ヴァンストン。応用暗号のハンドブック。 CRC
プレス、1996年。

[3] ウィリアムズHC。 「p+1 因数分解法」。数学。コンプ。 39、225-234、1982年。

[4] DE Knuth、The Art of Computer Programming、Vol. 2、第 3 版、アディソン•ウェスリー、1998 年、
アルゴリズム P、395 ページ。

[5] R. Baillie および SS Wagstaff Jr.、計算の数学、V. 35 (1980)、1391 ページ –
1417。