CRYPTREC GL-3001-2.0

SSL/TLS 暗号設定ガイドライン

平成 30 年 5 月

独立行政法人 情報処理推進機構
国立研究開発法人 情報通信研究機構

目次

1. はじめに 5
1.1 本書の内容及び位置付け 5
1.2 本書が対象とする読者 5
1.3 ガイドラインの検討体制 6
1.3.1 Version 2.0 における検討体制 6
1.3.2 Version 1.x における検討体制 7
2. 本ガイドラインの理解を助ける技術的な基礎知識 8
2.1 SSL/TLS の概要 8
2.1.1 SSL/TLS の歴史 8
2.1.2 プロトコル概要 10
2.1.3 TLS1.3 の概要 11
2.1.4 TLS プロトコルの最新動向 14
2.2 暗号アルゴリズムの安全性 15
2.2.1 CRYPTREC 暗号リスト 15
2.2.2 異なる暗号アルゴリズムにおける安全性の見方 16

PART I： サーバ構築における設定要求項目について 18
3. 設定基準の概要 19
3.1 実現すべき設定基準の考え方 19
3.2 要求設定の概要 21
3.3 チェックリスト 22
4. プロトコルバージョンの設定 24
4.1 プロトコルバージョンについての要求設定 24
4.2 プロトコルバージョンごとの安全性の違い 25
【コラム①】SSL/TLS から TLS へ －プロトコルとしての本格的な世代交代へ－ 26
5. サーバ証明書の設定 28
5.1 サーバ証明書についての要求設定 28
5.2 サーバ証明書に記載されている情報 31
5.3 サーバ証明書で利用可能な候補となる暗号アルゴリズム 32
5.4 サーバ証明書で考慮すべきこと 33
5.4.1 信頼できないサーバ証明書の利用は止める 33
5.4.2 ルート CA 証明書の安易な手動インストールは避ける 33
5.4.3 サーバ証明書で利用すべき鍵長 34
5.4.4 サーバ証明書を発行・更新する際に新しい鍵情報を生成する重要性 34
【コラム②】DNS の CAA (Certification Authority Authorization) リソースレコード ... 36
6. 暗号スイートの設定 37
6.1 暗号スイートについての要求設定 37
6.2 暗号スイートで利用可能な候補となる暗号アルゴリズム 39
6.3 鍵交換で考慮すべきこと 40

6.3.1 秘密鍵漏えい時の影響範囲を狭める手法の採用（Perfect Forward Secrecy の重要性） 40
6.3.2 鍵交換で利用すべき鍵長 41
6.3.3 DHE/ECDHE での鍵長の設定状況についての注意 41
6.4 暗号スイートについての実装状況 44
6.5 暗号スイートについての詳細な要求設定 44
6.5.1 高セキュリティ型での暗号スイートの詳細要求設定 44
6.5.2 推奨セキュリティ型での暗号スイートの詳細要求設定 45
6.5.3 セキュリティ例外型での暗号スイートの詳細要求設定 48
7. SSL/TLS を安全に使うために考慮すべきこと 49
7.1 サーバ証明書の作成・管理について注意すべきこと 49
7.1.1 サーバ証明書での脆弱な鍵ペアの使用の回避 49
7.1.2 推奨されるサーバ証明書の種類 49
7.1.3 サーバ証明書の有効期限 51
7.1.4 サーバ鍵の適切な管理 51
7.1.5 複数サーバに同一のサーバ証明書を利用する場合の注意 52
7.1.6 ルート CA 証明書 52
7.2 さらに安全性を高めるために 52
7.2.1 HTTP Strict Transport Security（HSTS）の設定有効化 52
7.2.2 リネゴシエーションの脆弱性への対策 53
7.2.3 圧縮機能を利用した実装攻撃への対策 54
7.2.4 OCSP Stapling の設定有効化 55
7.2.5 Public Key Pinning の設定有効化 56
【コラム③】完全 HTTPS 化の落とし穴 57

PART II： ブラウザ＆リモートアクセスの利用について 59
8. ブラウザを利用する際に注意すべきポイント 60
8.1 本ガイドラインが対象とするブラウザ 60
8.1.1 対象とするプラットフォーム 60
8.1.2 対象とするブラウザのバージョン 60
8.2 設定に関する確認項目 61
8.2.1 基本原則 61
8.2.2 設定項目 61
8.3 ブラウザ利用時の注意点 63
8.3.1 SHA-1 を利用するサーバ証明書の警告表示 63
9. その他のトピック 64
9.1 リモートアクセス VPN over SSL （いわゆる SSL-VPN） 64

Appendix： 付録 66
Appendix A：チェックリスト 67
A.1. チェックリストの利用方法 67

A.2.　高セキュリティ型のチェックリスト 68
A.3.　推奨セキュリティ型のチェックリスト 69
A.4.　セキュリティ例外型のチェックリスト 72
Appendix B：サーバ設定編 75
Appendix C：暗号スイートの設定例 75
Appendix D：ルート CA 証明書の取り扱い 76
D.1.　ルート CA 証明書の暗号アルゴリズムおよび鍵長の確認方法 76
D.2.　Active Directory を利用したプライベートルート CA 証明書の自動更新 78

【修正履歴】

修正日	修正内容
2018.5.8　(Ver.2.0)	最新動向を踏まえ、「セキュリティ例外型」を中心とした設定基準の見直しを実施。 最新データへの更新を実施。
2015.8.3　(Ver.1.1)	Appendix B.6 での誤記を修正

1. はじめに

1.1 本書の内容及び位置付け

本ガイドラインは、2018 年 3 月時点における、SSL/TLS 通信での安全性と可用性（相互接続性）のバランスを踏まえた SSL/TLS サーバの設定方法を示すものである。なお、前バージョン発行以降の各種動向を踏まえて設定基準の見直しを実施しているため、前バージョン以前の本ガイドラインを利用している場合には、今バージョンでの設定要件に基づいた見直しを行い、必要に応じて設定変更を実施することを強く推奨する。

本ガイドラインは 9 章で構成されており、章立ては以下のとおりである。

2 章では、本ガイドラインを理解するうえで助けとなる技術的な基礎知識をまとめている。特に高度な内容は含んでおらず、SSL/TLS 及び暗号についての技術的な基礎知識を有している読者は本章を飛ばしてもらって構わない。

3 章では、SSL/TLS サーバに要求される設定基準の概要について説明しており、4 章から 6 章で実現すべき要求設定の考え方を示す。

4 章から 6 章では、3 章で定めた設定基準に基づき、具体的な SSL/TLS サーバの要求設定について示す。本章の内容は、安全性と可用性を踏まえたうえで設定すべき「要求事項」である。

7 章では、チェックリストの対象には含めていないが、SSL/TLS を安全に使うために考慮すべきことをまとめている。本章の内容は、「情報提供」の位置づけとして記載している。

8 章は、クライアントの一つであるブラウザの設定に関する事項を説明しており、ブラウザの利用者に対して啓発するべき事項を取り上げている。本章の内容は、7 章と同様、「情報提供」の位置づけのものである。

9 章は、そのほかのトピックとして、SSL/TLS を用いたリモートアクセス技術（“SSL-VPN”とも言われる）について記載している。本章の内容も「情報提供」の位置づけのものである。

3 章から 6 章が本ガイドラインの最大の特長ともいえ、「暗号技術以外の様々な利用上の判断材料も加味した合理的な根拠」を重視して現実的な利用方法を目指している。具体的には、実現すべき安全性と必要となる相互接続性とのトレードオフを考慮する観点から、安全性と可用性を踏まえたうえで設定すべき「要求設定項目」として 3 つの設定基準（「高セキュリティ型」「推奨セキュリティ型」「セキュリティ例外型」）を提示している。

Appendix には、4 章から 6 章までの設定状況を確認するためのチェックリスト等を記載している。チェックリストの目的は、「選択した設定基準に対応した要求設定項目の設定忘れの防止」と「サーバ構築の作業受託先が適切に要求設定項目を設定したことの確認」を行うための手段となるものである。

1.2 本書が対象とする読者

対象読者は、主に SSL/TLS サーバを実際に構築するにあたって具体的な設定を行うサーバ構築者、実際のサーバ管理やサービス提供に責任を持つことになるサーバ管理者、並びに SSL/TLS サ

ーバの構築を発注するシステム担当者としている。

一部の内容については、ブラウザを使う一般利用者への注意喚起も対象とする。

1.3 ガイドラインの検討体制

1.3.1 Version 2.0 における検討体制

本ガイドライン Version 2.0 への改訂にあたっては、2017 年度 CRYPTREC 暗号技術活用委員会において、2015 年以降の動向調査を実施し、その結果を踏まえて本ガイドライン Version 1.x の記述を修正・追記・削除すべきかの検討を行った。

表 1 暗号技術活用委員会の構成（2018 年 3 月時点）

委員長	松本 勉	横浜国立大学　大学院環境情報研究院　教授
委員	上原 哲太郎	立命館大学　情報理工学部　情報システム学科　教授
委員	垣内 由梨香	日本マイクロソフト株式会社　セキュリティレスポンスチーム セキュリティプログラムマネージャー
委員	菊池 浩明	明治大学　総合数理学部　先端メディアサイエンス学科　教授
委員	清藤 武暢	日本銀行金融研究所　情報技術研究センター
委員	須賀 祐治	株式会社インターネットイニシアティブ セキュリティ本部セキュリティ情報統括室　シニアエンジニア
委員	杉尾 信行	株式会社 NTT ドコモ　サービスイノベーション部
委員	手塚 悟	慶應義塾大学　大学院政策・メディア研究科　特任教授
委員	寺村 亮一	NRI セキュアテクノロジーズ株式会社 サイバーセキュリティ事業開発部　主任
委員	松本 泰	セコム株式会社　IS 研究所 コミュニケーションプラットフォームディビジョン マネージャー
委員	三澤 学	三菱電機株式会社　情報技術総合研究所 情報ネットワーク基盤技術部　車載セキュリティグループ 主席研究員
委員	満塩 尚史	内閣官房　IT 総合戦略室　政府 CIO 補佐官
委員	山岸 篤弘	一般財団法人日本情報経済社会推進協会 電子署名・認証センター　主席研究員
委員	山口 利恵	東京大学　大学院情報理工学系研究科 ソーシャル ICT 研究センター　特任准教授
委員	渡邊 創	国立研究開発法人産業技術総合研究所 情報・人間工学領域　研究戦略部 研究企画室　研究企画室長
執筆 とりまとめ	神田 雅透	情報処理推進機構　技術本部　セキュリティセンター

1.3.2 Version 1.x における検討体制

本ガイドライン Version 1.x は、CRYPTREC 暗号技術活用委員会の配下に設置された運用ガイドラインワーキンググループに参加する委員の知見を集約したベストプラクティスとして作成されたものであり、暗号技術活用委員会の承認を得て発行されたものである。

運用ガイドラインワーキンググループは表 2 のメンバーにより構成されている。

表 2　運用ガイドラインワーキンググループの構成（2015 年 3 月時点）

主査	菊池 浩明	明治大学　総合数理学部 先端メディアサイエンス学科　教授
委員	阿部 貴	株式会社シマンテック　SSL 製品本部 SSL プロダクトマーケティング部　マネージャー
委員	漆嶌 賢二	富士ゼロックス株式会社　CS 品質本部 品質保証部 マネージャー
委員	及川 卓也	グーグル株式会社　エンジニアリング シニアエンジニアリングマネージャー
委員	加藤 誠	一般社団法人 Mozilla Japan　技術部 テクニカルアドバイザ
委員	佐藤 直之	株式会社イノベーションプラス Director
委員	島岡 政基	セコム株式会社 IS 研究所 コミュニケーションプラットフォームディビジョン 暗号・認証基盤グループ　主任研究員
委員	須賀 祐治	株式会社インターネットイニシアティブ サービスオペレーション本部　セキュリティ情報統括室 シニアエンジニア
委員	高木 浩光	独立行政法人産業技術総合研究所 セキュアシステム研究部門 主任研究員
委員	村木 由梨香	日本マイクロソフト株式会社　セキュリティレスポンスチーム セキュリティプログラムマネージャー
委員	山口 利恵	東京大学　大学院　情報理工学系研究科 ソーシャル ICT 研究センター　　特任准教授
執筆 とりまとめ	神田 雅透	情報処理推進機構　技術本部　セキュリティセンター

2. 本ガイドラインの理解を助ける技術的な基礎知識

2.1 SSL/TLS の概要

2.1.1 SSL/TLS の歴史

Secure Sockets Layer (SSL)はブラウザベンダであった Netscape 社により開発されたクライアントーサーバモデルにおけるセキュアプロトコルである。SSL には 3 つのバージョンが存在するがバージョン 1.0 は非公開である。SSL2.0 が 1995 年にリリースされたが、その後すぐに脆弱性が発見され、翌 1996 年に SSL3.0 [RFC6101] が公開されている。

標準化団体 Internet Engineering Task Force (IETF)はベンダ間での非互換性の問題を解決するために、Transport Layer Security Protocol Version 1.0 (TLS1.0) [RFC2246] を策定した。TLS1.0 は SSL3.0 をベースにしている。TLS1.0 で定められているプロトコルバージョンからも分かるように TLS1.0 は SSL3.1 とも呼ばれる。

TLS1.1 [RFC4346] は、TLS1.0 における暗号利用モードの一つである CBC[1]モードで利用される初期ベクトルの選択とパディングエラー処理に関連する脆弱性に対処するために仕様策定が行われた。具体的には BEAST[2]攻撃を回避することができる。

さらに TLS1.2 [RFC5246] は特にハッシュ関数 SHA-2 family (SHA-256 や SHA-384)の利用など、より強い暗号アルゴリズムの利用が可能になった。例えばメッセージ認証コード（MAC[3]）や擬似乱数関数にて SHA-2 family が利用可能になっている。また認証暗号が利用可能な暗号スイートのサポートがなされており、具体的には GCM[4]や CCM[5]モードの利用が可能になった。

表 3 に SSL/TLS のバージョンの概要をまとめる。最近では、IETF において、TLS1.3 の規格化の議論が進んでいる。

なお、SSL/TLS に対する攻撃方法の技術的な詳細については、「CRYPTREC 暗号技術ガイドライン（SSL/TLS における近年の攻撃への対応状況）[6]」を参照されたい。

表 3 SSL/TLS のバージョン概要

バージョン	概要
SSL2.0 （1994）	● いくつかの脆弱性が発見されており、なかでも「ダウングレード攻撃（最弱のアルゴリズムを強制的に使わせることができる）」と「バージョンロールバック攻撃（SSL2.0 を強制的に使わせることができる）」は致命的な脆弱性といえる ● SSL2.0 は利用すべきではなく、2005 年頃以降に出荷されているサーバやブラウザでは SSL2.0 は初期状態で利用不可となっている

[1] CBC: Cipher Block Chaining
[2] BEAST: Browser Exploit Against SSL/TLS
[3] MAC: Message Authentication Code
[4] GCM: Galois/Counter Mode
[5] CCM: Counter with CBC-MAC
[6] http://www.cryptrec.go.jp/report/c13_tech_guideline_TLSSSL_web.pdf

バージョン	概要
SSL3.0 （RFC6101） （1995）	● SSL2.0 での脆弱性に対処したバージョン ● 2014 年 10 月に POODLE[7]攻撃が発表されたことにより特定の環境下での CBC モードの利用は危険であることが認識されている。POODLE 攻撃は、SSL3.0 におけるパディングチェックの仕方の脆弱性に起因しているため、この攻撃に対する回避策は現在のところ存在していない ● POODLE 攻撃の発表を受け、2018 年 3 月時点での主流の最新版ブラウザで SSL3.0 は初期状態で利用不可となっている
TLS1.0 （RFC2246） （1999）	● 2018 年 3 月時点での SSL Pulse の調査結果[8]によれば、約 12 万の主要なサイトについて TLS1.0 が利用できるのは約 88% ● ブロック暗号を CBC モードで利用した時の脆弱性を利用した攻撃（BEAST 攻撃など）が広く知られているが、容易な攻撃回避策が存在し、すでにセキュリティパッチも提供されている。また、SSL3.0 で問題となった POODLE 攻撃は、プロトコルの仕様上、TLS1.0 には適用できない ● 暗号スイートとして、より安全なブロック暗号の AES と Camellia、並びに公開鍵暗号・署名に楕円曲線暗号が利用できるようになった ● 秘密鍵の生成などに擬似乱数関数を採用 ● MAC の計算方法を HMAC に変更
TLS1.1 （RFC4346） （2006）	● ブロック暗号を CBC モードで利用した時の脆弱性を利用した攻撃（BEAST 攻撃等）への対策を予め仕様に組み入れるなど、TLS1.0 の安全性強化を図っている ● 実装に関しては、規格化された年が TLS1.2 とあまり変わらなかったため、TLS1.1 と TLS1.2 は同時に実装されるケースも多く、2018 年 3 月時点での SSL Pulse の調査結果[8]によれば約 12 万の主要なサイトについて TLS1.1 が利用できるのは約 85%
TLS1.2 （RFC5246） （2008）	● 暗号スイートとして、より安全なハッシュ関数 SHA-256 と SHA-384、CBC モードより安全な認証付き秘匿モード（GCM、CCM）が利用できるようになった。特に、認証付き秘匿モードでは、利用するブロック暗号が同じであっても、CBC モードの脆弱性を利用した攻撃（BEAST 攻撃等）がそもそも適用できない ● 必須の暗号スイートを TLS_RSA_WITH_AES_128_CBC_SHA に変更 ● IDEA と DES を使う暗号スイートを削除 ● 擬似乱数関数の構成を MD5/SHA-1 ベースから SHA-256 ベースに変更 ● 2018 年 3 月時点での SSL Pulse の調査結果[8]によれば約 12 万の主要なサイトについて TLS1.2 が利用できるのは約 91%

[7] POODLE: Padding Oracle On Downgraded Legacy Encryption
[8] https://www.ssllabs.com/ssl-pulse/

バージョン	概要
TLS1.3 （draft28）	● 共通鍵暗号は認証暗号（AEAD: Authenticated Encryption with Associated Data）のみ採用した結果、AES-GCM、AES-CCM、ChaCha20-Poly1305 のみが規定された。このうち、AES-GCM が必須になった ● 鍵交換は、DHE、ECDHE、PSK のみが規定され、ECDHE が必須になった ● 署名は、RSA-PSS、RSASSA-PKCS1-v1_5、ECDSA が必須になった ● ハッシュ関数は SHA-256 以上が規定された。このうち、SHA-256 が必須になった ● 楕円曲線は secp256r1 (NIST P-256)が必須になった ● ハンドシェイク性能の向上のため、1-RTT、0-RTT（Round Trip Time）になるようにシーケンスが簡素化された ● ハンドシェイクのデータを暗号化して保護した ● TLS1.2 互換に配慮し、ClientHello、ServerHello、ChangeCipherSpec が規定された ● まだ draft であるが、サーバやブラウザで実装が始まっている

2.1.2 プロトコル概要

SSL/TLS はセッション層に位置するセキュアプロトコルで、通信の暗号化、データ完全性の確保、サーバ（場合によりクライアント）の認証を行うことができる。セッション層に位置することで､アプリケーション層ごとにセキュリティ確保のための仕組みを実装する必要がなく､HTTP、SMTP、POP など様々なプロトコルの下位に位置して、上記の機能を提供することができる。

SSL/TLS では、暗号通信を始めるに先立って、ハンドシェイクが実行される（図 1 参照)。

ハンドシェイクでは、①ブラウザ（クライアント）とサーバが暗号通信するために利用する暗号アルゴリズムとプロトコルバージョンを決定し､②サーバ証明書によってサーバの認証を行い、③そのセッションで利用するセッション鍵を共有する、までの一連の動作を行う。

その際、SSL/TLS では相互接続性の確保を優先してきたため、一般には複数の暗号アルゴリズムとプロトコルバージョンが実装されている。結果として、暗号通信における安全性強度は、ハンドシェイクの①の処理でどの暗号アルゴリズムとプロトコルバージョンを選択したかに大きく依存する。

サイトの身分証明ともいえるサーバ証明書は、Trusted Third Party である認証局（CA[9]）によって発行されるのが一般的であり、特に Web Trust for CA などの一定の基準を満たしている代表的な認証局の証明書はルート CA として予めブラウザに登録されている。(4)の検証では、ブラウザに登録された認証局の証明書を信頼の起点として、当該サーバ証明書の正当性を確認する。

[9] Certificate Authority

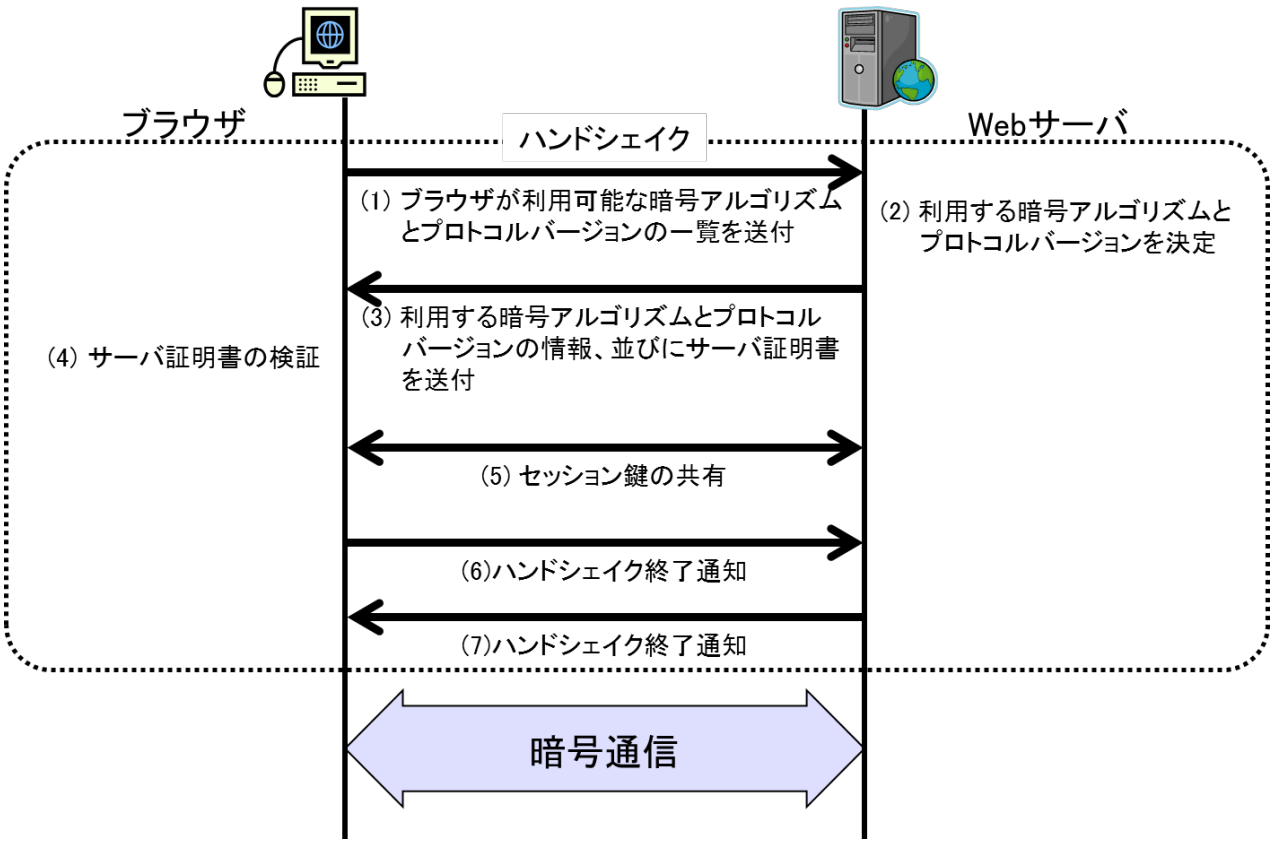


図 1　SSL/TLS プロトコル概要

2.1.3　TLS1.3 の概要

TLS1.3 は、TLS1.2 策定以降に見つかった新たな脆弱性や攻撃手法への対策を施すと共に、QUIC 等のプロトコルに対応するための性能向上を狙いとして、プロトコルと暗号アルゴリズムの抜本的な再設計が行われた。最新仕様は draft28 であり、2018 年 3 月に RFC Editor Queue に進み、RFC 発効前の最終作業が続いている（2018 年 4 月現在）。

TLS1.2（以前）との差異の観点から見た主な特徴を以下に示す。

(1) 脆弱なアルゴリズムとして、Triple DES、DSA、RC4、MD5、SHA-1、SHA-224、静的な RSA が削除された。また、認証暗号（AEAD）でない AES の CBC モードが削除された。
(2) NIST FIPS/SP で規定されていないアルゴリズムとして、共通鍵暗号の ChaCha20 と署名の EdDSA が追加された。
(3) 鍵交換は、DHE、ECDHE、PSK のみが規定され、ECDHE が必須になった。
(4) 楕円曲線として secp256r1 が必須になった。
(5) ハッシュ関数は SHA-256 が必須になった。
(6) 脆弱なハンドシェイク機能として、リネゴシエーション、圧縮、セッション回復が削除された。

(7) HMAC ベースの導出関数（HKDF-Expand(・), HKDF-Extract(・)）を使った鍵導出に変更された。

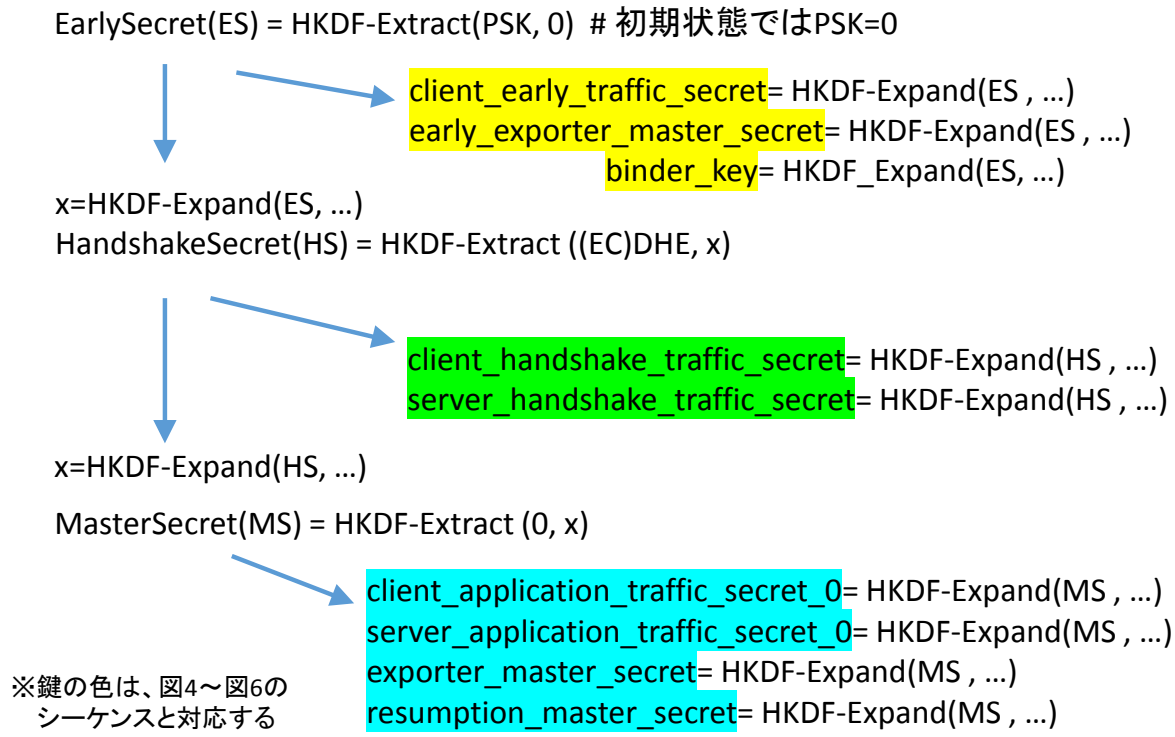


図 2　鍵の導出方法

(8) ServerHello 以降のハンドシェイクパラメータを暗号化して保護する。

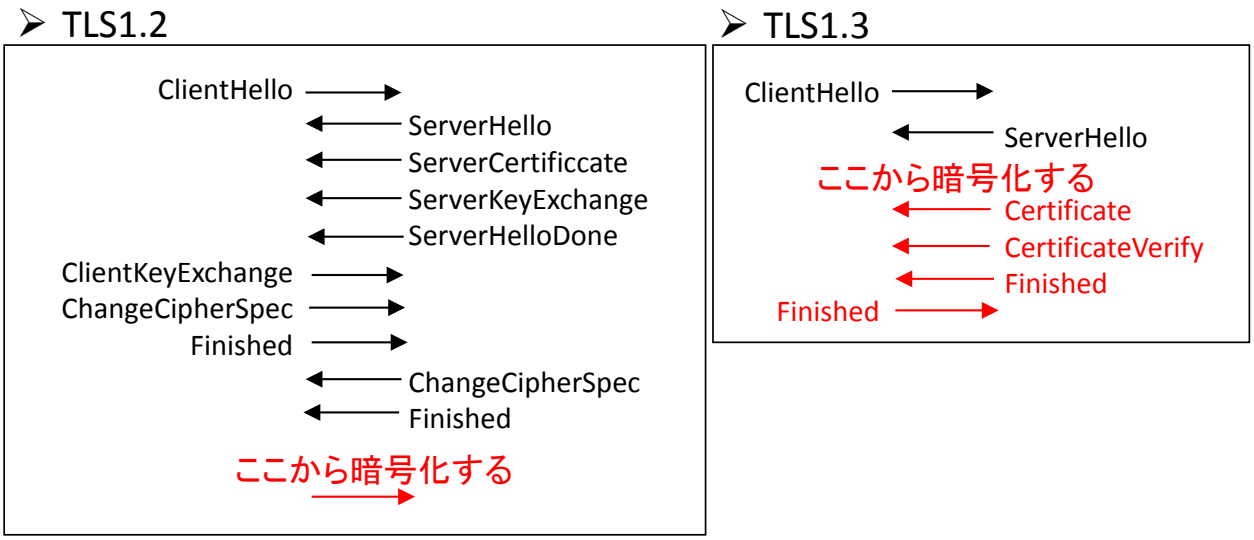


図 3　TLS1.2 と TLS1.3 との暗号化開始個所の比較

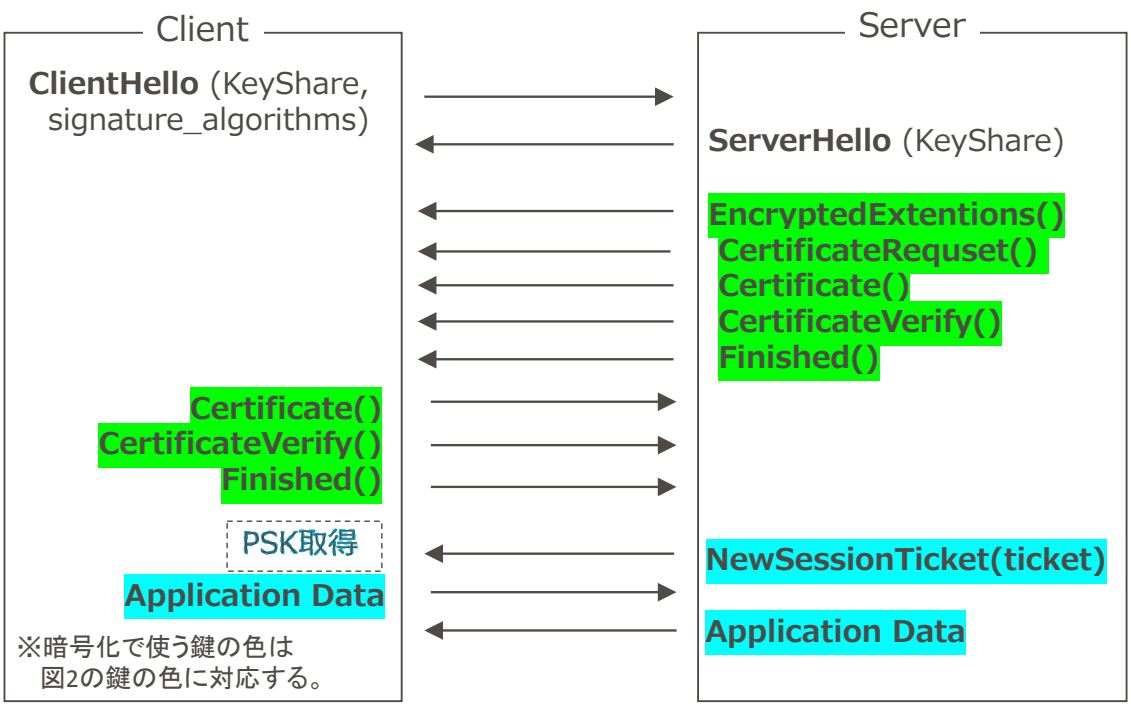


図 4　TLS1.3 のシーケンス図

(9) 性能向上のため、1-RTT でハンドシェイクが完了するようにメッセージおよび拡張が見直された。

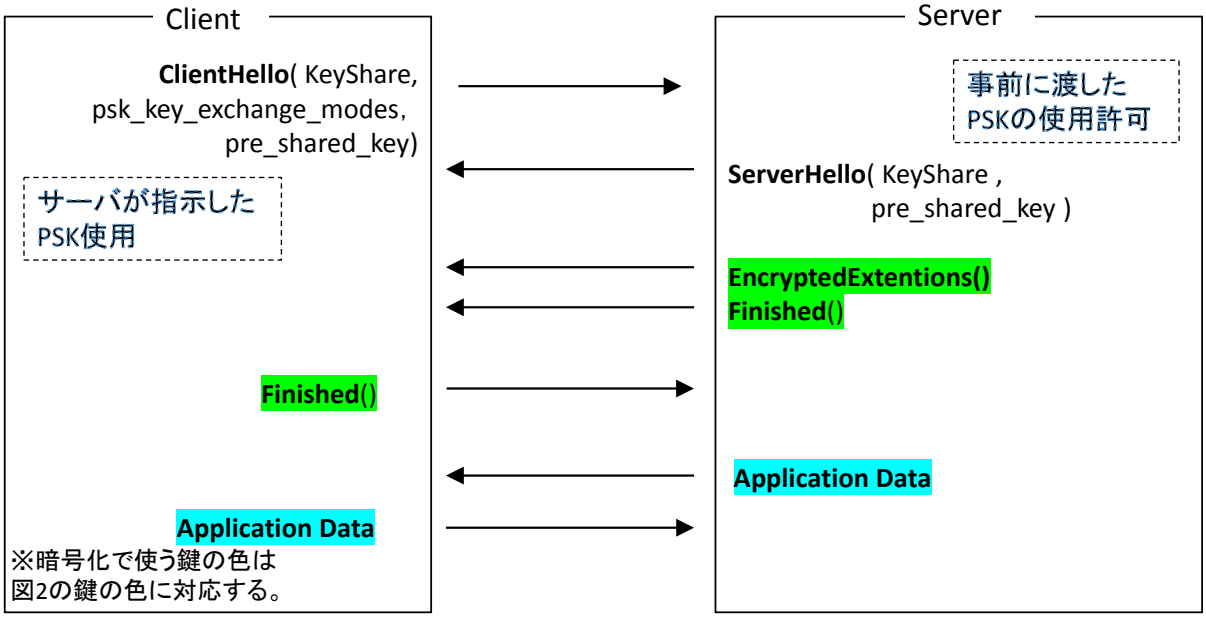


図 5　1-RTT のシーケンス図

(10) QUIC 等への適用を考慮し、0-RTT でアプリデータを送信する機能が追加された。

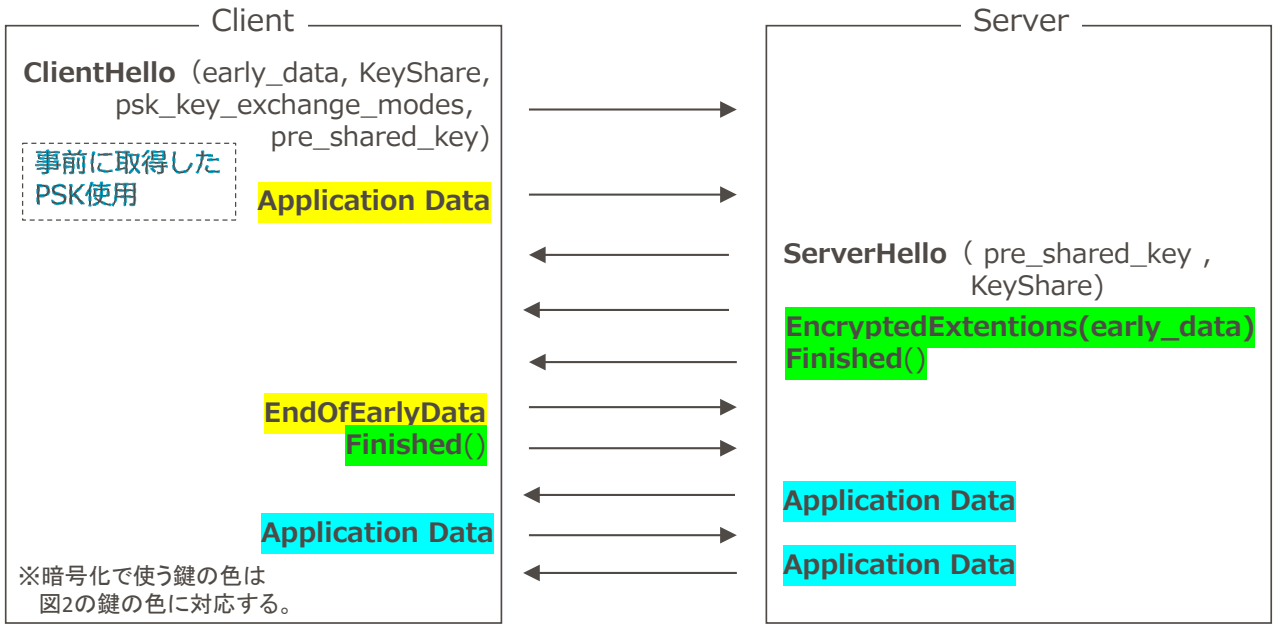


図 6　0-RTT のシーケンス図

(11) ClientHello、ServerHello、ChangeCipherSpec の TLS1.2 互換性を保つことにより、中間ノードによる接続性を向上した。

2.1.4　TLS プロトコルの最新動向

TLS1.3 がまもなく RFC として発行されるのを受け（ガイドライン発行時の状況によっては文章修正の可能性有り）、2017 年 11 月に NIST は TLS に関する新たなガイドラインのドラフト版[10]を発表した。このドラフト版では、2020 年 1 月 1 日までに、①連邦政府で利用する全てのサーバ及びクライアント（ブラウザ）で TLS1.2 をサポートすることを要求するとともに、②TLS1.3 をサポートし移行する計画を作るよう勧告する、内容になっている。

また、2015 年 4 月以降に発行された SSL/TLS に関する RFC 32 件のうち、「プロトコルバージョン」「サーバ証明書」「暗号スイート（暗号アルゴリズム）」の 3 つの観点から、利用可否や利用期間などの記述が含まれるものは、以下のとおりである。例えば、既存の TLS1.2 までのプロトコルに対して、SSL3.0 の無効化や RC4 の無効化など、プロトコルの脆弱性の排除に関するものが規格化されている。

[10] NIST SP 800-52 Rev. 2 (draft), Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations

表 4　2015 年 4 月以降に「プロトコルバージョン」「サーバ証明書」
「暗号スイート（暗号アルゴリズム)」に関連して発行された RFC

RFC	Title	プロトコル	サーバ証明書	暗号スイート	内容
7465	Prohibiting RC4 Cipher Suites	×	×	○	RC4 禁止
7507	TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks	×	×	○	新暗号スイートの定義
7525	Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)	○	×	×	SSL2.0, SSL3.0 禁止 TLS1.0, TLS1.1 非推奨
7568	Deprecating Secure Sockets Layer Version 3.0	○	×	×	SSL3.0 禁止
7905	ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)	×	×	○	ChaCha20-Poly1305 の暗号スイート追加

2.2 暗号アルゴリズムの安全性

2.2.1 CRYPTREC 暗号リスト

総務省と経済産業省は、暗号技術に関する有識者で構成される CRYPTREC 活動を通して、電子政府で利用される暗号技術の評価を行っており、2013 年 3 月に「電子政府における調達のために参照すべき暗号のリスト(CRYPTREC 暗号リスト)」 を策定した[11]。CRYPTREC 暗号リストは、「電子政府推奨暗号リスト」、「推奨候補暗号リスト」及び「運用監視暗号リスト」で構成される。

「政府機関の情報セキュリティ対策のための統一基準（平成 28 年度版)」（平成 28 年 8 月 31 日、サイバーセキュリティ戦略本部）では以下のように記載されており、 政府機関における情報システムの調達及び利用において、CRYPTREC 暗号リストのうち「電子政府推奨暗号リスト」が原則的に利用される。

政府機関の情報セキュリティ対策のための統一基準（抄）

6.1.5 暗号・電子署名－遵守事項(1)(b)

情報システムセキュリティ責任者は、暗号技術検討会及び関連委員会（CRYPTREC）により安全性及び実装性能が確認された「電子政府推奨暗号リスト」を参照した上で、情報システムで使用する暗号及び電子署名のアルゴリズム並びにそれを利用した安全なプロトコル及びその運用方法について、以下の事項を含めて定めること。

（ア） 行政事務従事者が暗号化及び電子署名に対して使用するアルゴリズム及びそれを利用した安全なプロトコルについて、「電子政府推奨暗号リスト」に記載され

[11] http://www.cryptrec.go.jp/list.html

た暗号化及び電子署名のアルゴリズムが使用可能な場合には、それを使用させること。

（イ） 情報システムの新規構築又は更新に伴い、暗号化又は電子署名を導入する場合には、やむを得ない場合を除き、「電子政府推奨暗号リスト」に記載されたアルゴリズム及びそれを利用した安全なプロトコルを採用すること。

（以下、略）

2.2.2 異なる暗号アルゴリズムにおける安全性の見方

異なる技術分類の暗号アルゴリズムを組合せて利用する際、ある技術分類の暗号アルゴリズムの安全性が極めて高いものであっても、別の技術分類の暗号アルゴリズムの安全性が低ければ、結果として、低い安全性の暗号アルゴリズムに引きずられる形で全体の安全性が決まる。逆に言えば、異なる技術分類の暗号アルゴリズムであっても、同程度の安全性とみなされている暗号アルゴリズムを組合せれば、全体としても同程度の安全性が実現できることになる。

異なる技術分類の暗号アルゴリズムについて同程度の安全性を持つかどうかを判断する目安として、“ビット安全性（等価安全性ということもある)”という指標がある。具体的には、評価対象とする暗号アルゴリズムに対してもっとも効率的な攻撃手法を用いたときに、どの程度の計算量があれば解読できるか（解読計算量[12]）で表現され、鍵長[13]とは別に求められる。表記上、解読計算量が 2^x である場合に“x ビット安全性”という。例えば、共通鍵暗号においては、全数探索する際の鍵空間の大きさで 2^x（x は共通鍵のビット長)、ハッシュ関数の例としては、一方向性で 2^x、衝突困難性で $2^{(x/2)}$（x は出力ビット長）が解読計算量の（最大）理論値である。

“ビット安全性”による評価では、技術分類に関わらず、どの暗号アルゴリズムであっても、解読計算量が大きければ安全性が高く、逆に小さければ安全性が低い。また、解読計算量が実現可能と考えられる計算量を大幅に上回っていれば、少なくとも現在知られているような攻撃手法ではその暗号アルゴリズムを破ることは現実的に不可能であると予測される。

そこで、暗号アルゴリズムの選択においては、“x ビット安全性”の“x ビット”に着目して、長期的な利用期間の目安とする使い方ができる。例えば、NIST SP800-57 Part 1 revision 4[14]では、表 5 のように規定している。

なお、表記の便宜上、本ガイドラインでは以下の表記を用いる。

- AES-xxx：鍵長が xxx ビットの AES のこと
- Camellia-xxx：鍵長が xxx ビットの Camellia のこと
- RSA-xxx：鍵長が xxx ビットの RSA のこと
- DH-xxx：鍵長が xxx ビットの DH のこと
- ECDH-xxx：鍵長が xxx ビット（例えば NIST 曲線パラメータ P-xxx を利用）の ECDH のこと

[12] 直感的には、基本となる暗号化処理の繰り返し回数のことである。例えば、解読計算量 2^{20} といえば、暗号化処理 2^{20} 回相当の演算を繰り返し行えば解読できることを意味する

[13] ハッシュ関数の場合はダイジェスト長に相当する

[14] NIST SP800-57, Recommendation for Key Management – Part 1: General (Revision 4)

- ECDSA-xxx : 鍵長が xxx ビット（例えば NIST 曲線パラメータ P-xxx を利用）の ECDSA のこと
- HMAC-SHA-xxx: メッセージ認証子を作る HMAC において利用するハッシュ関数 SHA-xxx のこと。SSL/TLS では、暗号スイートで決めるハッシュ関数は HMAC として利用される。
- SHA-xxx：デジタル署名を作成する際に利用するハッシュ関数 SHA-xxx のこと。

表 5　NIST SP800-57 でのビット安全性の取り扱い方針（WG で加工）

<table>
<tr><td rowspan="2">ビット安全性</td><td rowspan="2">SSL/TLS で利用する代表的な暗号アルゴリズム</td><td rowspan="2">利用上の条件</td><td colspan="2">長期的な利用期間</td></tr>
<tr><td>2030 年まで</td><td>2031 年以降</td></tr>
<tr><td rowspan="2">80 ビット</td><td rowspan="2">RSA-1024
DH-1024
ECDH-160
ECDSA-160
SHA-1</td><td>新規に処理をする場合</td><td>利用不可</td><td>利用不可</td></tr>
<tr><td>過去に処理したものを利用する場合</td><td colspan="2">過去のシステムとの互換性維持の利用だけを容認</td></tr>
<tr><td rowspan="2">112 ビット</td><td rowspan="2">3-key Triple DES
RSA-2048
DH-2048
ECDH-224
ECDSA-224</td><td>新規に処理をする場合</td><td>利用可</td><td>利用不可</td></tr>
<tr><td>過去に処理したものを利用する場合</td><td>利用可</td><td>過去のシステムとの互換性維持の利用だけを容認</td></tr>
<tr><td>128 ビット</td><td>AES-128
Camellia-128
ECDH-256
ECDSA-256
SHA-256</td><td>特になし</td><td>利用可</td><td>利用可</td></tr>
<tr><td>128 ビットから 192 ビットの間</td><td>RSA-4096
DH-4096
HMAC-SHA-1</td><td>特になし</td><td>利用可</td><td>利用可</td></tr>
<tr><td>192 ビット</td><td>ECDH-384
ECDSA-384
SHA-384</td><td>特になし</td><td>利用可</td><td>利用可</td></tr>
<tr><td>256 ビット</td><td>AES-256
Camellia-256
ECDH-521
ECDSA-521
HMAC-SHA256</td><td>特になし</td><td>利用可</td><td>利用可</td></tr>
<tr><td>256 ビット以上</td><td>HMAC-SHA384</td><td>特になし</td><td>利用可</td><td>利用可</td></tr>
</table>

PART I：

サーバ構築における設定要求項目について

3. 設定基準の概要

本章では、SSL/TLS サーバの構築時に、主に暗号通信に関わる設定に関しての要求事項を決めるために考慮すべきポイントについて取りまとめる。

3.1 実現すべき設定基準の考え方

SSL/TLS は、1994 年に SSL2.0 が実装されて以来、その利便性から多くの製品に実装され、利用されている。一方、プロトコルの脆弱性に対応するため、何度かプロトコルとしてのバージョンアップが行われている影響で、製品の違いによってサポートしているプロトコルバージョンや暗号スイート等が異なり、相互接続性上の問題が生じる可能性がある。

本ガイドラインでは、安全性の確保と相互接続の必要性のトレードオフにより、「高セキュリティ型」「推奨セキュリティ型」「セキュリティ例外型」の 3 段階の設定基準に分けて各々の要求設定を定める。それぞれの設定基準における、安全性と相互接続性についての関係は表 6 のとおりである。

実際にどの設定基準を採用するかは、安全性の確保と相互接続の必要性の両面を鑑みて、サーバ管理やサービス提供に責任を持つ管理者が最終的に決定すべきことではあるが、本ガイドラインでは、安全性もしくは相互接続性についての特段の要求がなければ「推奨セキュリティ型」の採用を強く勧める。本ガイドラインの発行時点では、「推奨セキュリティ型」がもっとも安全性と可用性（相互接続性）のバランスが取れている要求設定であると考えている。

「セキュリティ例外型」は、システム等の制約上、脆弱なプロトコルバージョンである SSL3.0 の利用を全面禁止することが現実的ではなく、安全性上のリスクを受容してでも SSL3.0 を継続利用せざるを得ないと判断される場合にのみ採用すべきである。なお、セキュリティ例外型であっても、SSL3.0 の無期限の継続利用を認めているわけではなく、近いうちに SSL3.0 を利用不可に設定するように変更される可能性がある。

また、SSL3.0 を利用する関係から、利用可能な暗号スイートの設定においても、脆弱な暗号アルゴリズムである RC4 の利用を認めている。ただし、本来的には RC4 は SSL3.0 に限定して利用すべきであるが、TLS1.0 以上のプロトコルバージョンで RC4 の利用を不可にする設定を行うことが難しいため、TLS1.0 以上であっても RC4 が使われる可能性が排除できないことにも注意されたい。

したがって、セキュリティ例外型を採用する際は、推奨セキュリティ型への移行完了までの短期の暫定運用として、移行計画や利用終了期限を定めたりするなど、今後への具体的な対処方針の策定をすべきである。

表 6　安全性と相互接続性についての比較

設定基準	概要	安全性	相互接続性の確保
高セキュリティ型	扱う情報が漏えいした際、組織の運営や資産、個人の資産やプライバシー等に致命的または壊滅的な悪影響を及ぼすと予想される情報を、高い安全性を確保して SSL/TLS で通信するような場合に採用する設定基準 **※高い安全性を必要とする理由があるケースを対象としており、高度な使い方をする場合の設定基準である。** ＜利用例＞ 政府内利用（G2G 型）のなかでも、高い安全性が要求される通信を行う場合	本ガイドラインの公開時点（2018 年 5 月）において、標準的な水準を大きく上回る高い安全性水準を達成	本ガイドラインで対象とするブラウザ（8.1.2 節）が搭載されている PC、スマートフォンなどでは問題なく相互接続性を確保できる 本ガイドラインが対象としない、バージョンが古い OS やブラウザの場合や発売開始からある程度の年月が経過している一部の古い機器（フィーチャーフォンやゲーム機など）については接続できない可能性がある
推奨セキュリティ型	扱う情報が漏えいした際、組織の運営や資産、個人の資産やプライバシー等に何らかの悪影響を及ぼすと予想される情報を、安全性確保と利便性実現をバランスさせて SSL/TLS での通信を行うための標準的な設定基準 **※ほぼすべての一般的な利用形態で使うことを想定している** ＜利用例＞ • 政府内利用（G2G 型）や社内システムへのリモートアクセスなど、特定された通信相手との安全な通信が要求される場合 • 電子申請など、企業・国民と役所等との電子行政サービスを提供する場合 • 金融サービスや電子商取引サービス、多様な個人情報の入力を必須とするサービス等を提供する場合 • 既存システムとの相互接続を考慮することなく、新規に社内システムを構築する場合	本ガイドラインの公開時点（2018 年 5 月）における標準的な安全性水準を実現	ほとんどのすべての機器について相互接続性を確保できる ※すでにサポートが切れているなどかなり古い機器などで接続できない場合があるが、この種の機器は本来接続させるべきではない

設定基準	概要	安全性	相互接続性の確保
セキュリティ 例外型	脆弱なプロトコルバージョンや暗号が使われるリスクを受容したうえで、安全性よりも相互接続性に対する要求をやむなく優先させてSSL/TLS での通信を行う場合であって、推奨セキュリティ型への移行完了までの短期の暫定運用としての設定基準 ＜利用例＞ • 利用するサーバやクライアントの実装上の制約、もしくは既存システムとの相互接続上の制約により、推奨セキュリティ型（以上）の設定が事実上できない場合	本ガイドラインの公開時点（2018 年 5 月）において、最低限度の安全性水準を満たしているとは言えない状況になっている。速やかな推奨セキュリティ型への移行を強く求める	ほとんどのすべての機器について相互接続性を確保できる

3.2 要求設定の概要

SSL/TLS における暗号通信に関わる設定には以下のものがある。

- プロトコルバージョンの設定（4 章）
- サーバ証明書の設定（5 章）
- 暗号スイートの設定（6 章）
- SSL/TLS を安全に使うために考慮すべきこと（7 章）

本ガイドラインでは、上記の設定項目のうち、プロトコルバージョン、サーバ証明書、暗号スイートの 3 つの項目について、3.1 節に記載した設定基準に対応した詳細な要求設定を該当章に各々まとめている。

表 7 に要求設定の概要を記す。

表 7　要求設定の概要

<table>
<tr><th colspan="2">要件</th><th>高セキュリティ型</th><th>推奨セキュリティ型</th><th>セキュリティ例外型</th></tr>
<tr><td colspan="2">想定対象</td><td>G2G 等</td><td>一般</td><td>推奨セキュリティ型以上の設定が現実的ではない等の特殊事情があるケースに限定</td></tr>
<tr><td colspan="2">暗号スイートの（暗号化）セキュリティレベル</td><td>①256 bit
②128 bit</td><td>①128 bit
②256 bit</td><td>① 128 bit
② 256 bit
③ RC4, Triple DES</td></tr>
<tr><td rowspan="6">暗号アルゴリズム</td><td rowspan="2">鍵交換</td><td>鍵長 2048 ビット以上の DHE または 鍵長 256 ビット以上の ECDHE</td><td colspan="2">鍵長 1024 ビット以上の DHE または鍵長 256 ビット以上の ECDHE</td></tr>
<tr><td></td><td colspan="2">鍵長 2048 ビット以上の RSA
鍵長 256 ビット以上の ECDH</td></tr>
<tr><td rowspan="2">暗号化</td><td colspan="3">鍵長 128 ビット及び 256 ビットの AES または Camellia</td></tr>
<tr><td></td><td></td><td>RC4
Triple DES</td></tr>
<tr><td>モード</td><td>GCM</td><td colspan="2">GCM, CBC</td></tr>
<tr><td>ハッシュ関数</td><td>SHA-384, SHA-256</td><td>SHA-384, SHA-256, SHA-1*</td><td>SHA-384, SHA-256, SHA-1</td></tr>
<tr><td colspan="2">プロトコルバージョン</td><td>TLS1.2 のみ</td><td>TLS1.2 ～ TLS1.0</td><td>TLS1.2～1.0, SSL3.0</td></tr>
<tr><td colspan="2">証明書鍵長</td><td colspan="3">鍵長 2048 ビット以上の RSA または 鍵長 256 ビット以上の ECDSA</td></tr>
<tr><td colspan="2">証明書でのハッシュ関数</td><td colspan="2">SHA-256</td><td>SHA-256, SHA-1</td></tr>
</table>

* 署名生成及び証明書での利用を除く

3.3 チェックリスト

図 7 に高セキュリティ型のチェックリストのイメージを示す。
チェックリストの目的は、

- 選択した設定基準に対応した要求設定項目をもれなく実施したことを確認し、設定忘れを防止すること

- サーバ構築の作業受託先が適切に要求設定項目を設定したことを、発注者が文書として確認する手段を与えること

である。したがって、選択した設定基準に応じたチェックリストを用い、すべてのチェック項目について、該当章に記載の要求設定に合致していることを確認して「済」にチェックが入ることが求められる。

Appendix A には、各々の設定基準に対応したチェックリストを載せる。

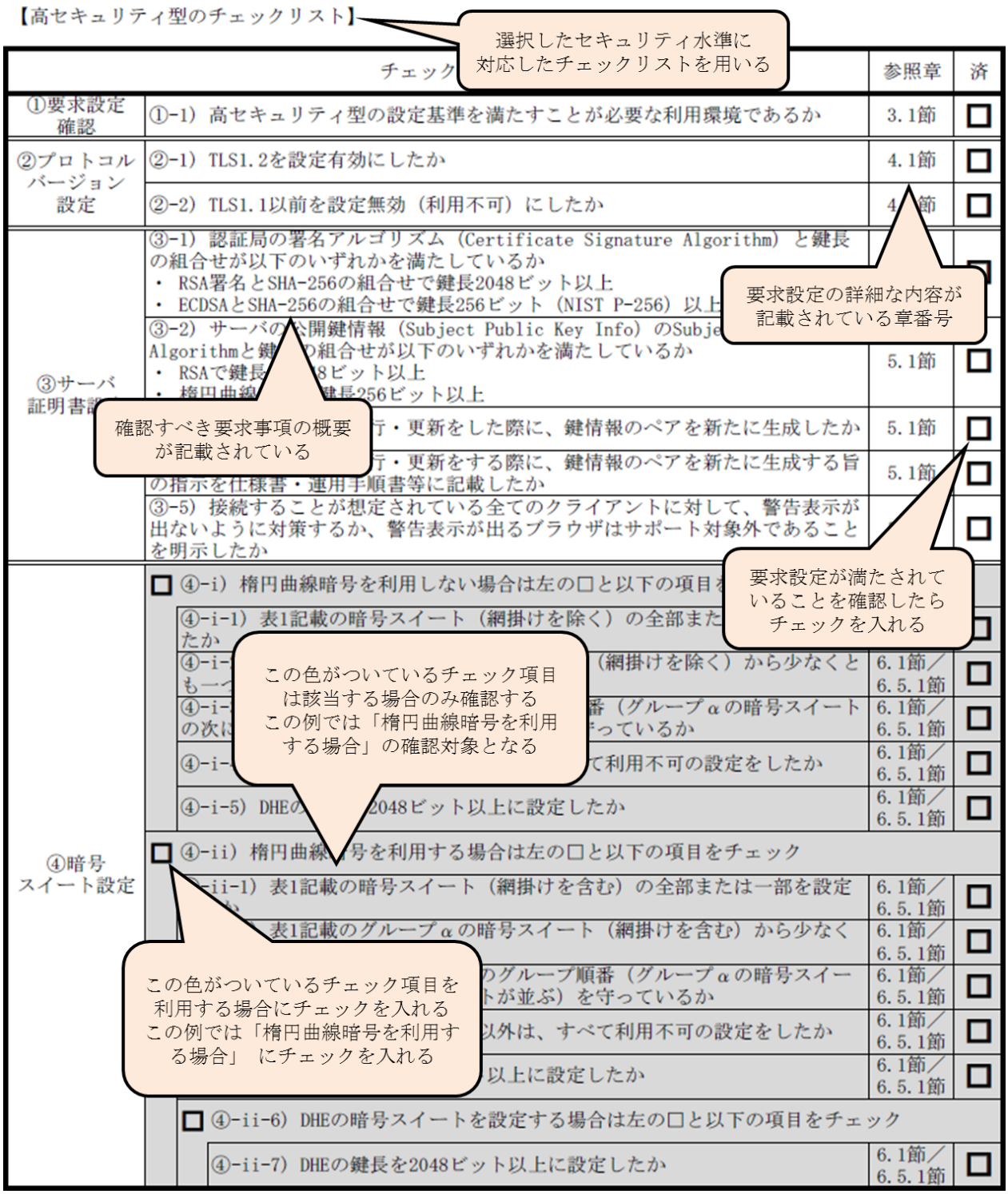


図 7　チェックリスト（高セキュリティ型の例）

4. プロトコルバージョンの設定

SSL/TLS は、1994 年に SSL2.0 が実装され始めた後、2014 年 3 月現在の最新版となる TLS1.2 まで 5 つのプロトコルバージョンが実装されている。4.1 節にプロトコルバージョンについての要求設定をまとめる。4.2 節にプロトコルバージョンごとの安全性の違いを記す。

4.1 プロトコルバージョンについての要求設定

基本的に、プロトコルのバージョンが後になるほど、以前の攻撃に対する対策が盛り込まれるため、より安全性が高くなる。しかし、相互接続性も確保する観点から、多くの場合、複数のプロトコルバージョンが利用できるように実装されているので、プロトコルバージョンの選択順位を正しく設定しておかなければ、予想外のプロトコルバージョンで SSL/TLS 通信を始めることになりかねない。

そこで、SSL2.0 から TLS1.2 までの安全性の違い（4.2 節 表 8 参照）を踏まえ、SSL/TLS サーバがサポートするプロトコルバージョンについての要求設定を以下のように定める。なお、高セキュリティ型の要求設定ではサーバとクライアントの両方が TLS1.2 をサポートしていることが必須となることに注意されたい。

【高セキュリティ型の要求設定】

- TLS1.2 を設定有効にする
- TLS1.1 以前を設定無効（利用不可）にする

TLS1.2	TLS1.1	TLS1.0	SSL3.0	SSL2.0
◎	×	×	×	×

◎：設定有効　　×：設定無効化　　－：実装なし

【推奨セキュリティ型の要求設定】

- SSL3.0 及び SSL2.0 を設定無効（利用不可）にする
- TLS1.1、TLS1.2 については、実装されているのであれば、設定有効にする
- プロトコルバージョンの優先順位が設定できる場合には、設定有効になっているプロトコルバージョンのうち、最も新しいバージョンによる接続を最優先とし、接続できない場合に順番に一つずつ前のプロトコルバージョンでの接続するように設定することが望ましい

TLS1.2	TLS1.1	TLS1.0	SSL3.0	SSL2.0
◎	○	○	×	×
－	◎	○	×	×
－	－	◎	×	×

○：設定有効（◎：優先するのが望ましい）　　×：設定無効化　　－：実装なし

【セキュリティ例外型の要求設定】

- SSL2.0 を設定無効（利用不可）にする
- TLS1.1、TLS1.2 については、実装されているのであれば、設定有効にする
- プロトコルバージョンの優先順位が設定できる場合には、設定有効になっているプロトコルバージョンのうち、最も新しいバージョンによる接続を最優先とし、接続できない場合に順番に一つずつ前のプロトコルバージョンでの接続するように設定することが望ましい

	TLS1.2	TLS1.1	TLS1.0	SSL3.0	SSL2.0
3 つのうちのいずれか	◎	○	○	○	×
	－	◎	○	○	×
	－	－	◎	○	×

○：設定有効（◎：優先するのが望ましい）　×：設定無効化　－：実装なし

4.2 プロトコルバージョンごとの安全性の違い

SSL2.0 から TLS1.2 までの各プロトコルバージョンにおける安全性の違いを表 8 にまとめる。

表 8　プロトコルバージョンでの安全性の違い

SSL/TLS への攻撃方法に対する耐性	TLS1.2	TLS1.1	TLS1.0	SSL3.0	SSL2.0
ダウングレード攻撃(最弱の暗号アルゴリズムを強制的に使わせることができる)	安全	安全	安全	安全	脆弱
バージョンロールバック攻撃（SSL2.0 を強制的に使わせることができる）	安全	安全	安全	安全	脆弱
ブロック暗号の CBC モード利用時の脆弱性を利用した攻撃(BEAST/POODLE 攻撃など)	安全	安全	パッチ適用要	脆弱	脆弱
利用可能な暗号アルゴリズム	**TLS1.2**	**TLS1.1**	**TLS1.0**	**SSL3.0**	**SSL2.0**
128 ビットブロック暗号（AES, Camellia）	可	可	可	不可	不可
認証付き秘匿モード（GCM, CCM）	可	不可	不可	不可	不可
楕円曲線暗号	可	可	可	不可	不可
SHA-2 ハッシュ関数（SHA-256, SHA-384）	可	不可	不可	不可	不可

【コラム①】SSL/TLS から TLS へ －プロトコルとしての本格的な世代交代へ－

インターネットは、1960 年代の ARPANET 開発等を起源に、1980 年代に学術ネットワークとして世界中に広がっていったネットワークである。この時代には、「ネチケット（ネットワーク＋エチケット：基本的なネットワーク利用ルールを意味する造語）」という言葉が存在したように、「限られた善良なユーザが暗黙の利用ルールを守ってインターネットを利用する」という性善説に立った運用がなされており、セキュリティ確保はあまり重要な要件ではなかった。

1990 年代にビジネス利用が徐々に解禁されると、悪意を持ったユーザがインターネットに入り込むことが容易になり、またネットワーク初心者も増えた結果、性善説に立ってインターネットを運用することが難しい状況になった。そのような状況になって、セキュリティ確保の重要性が高まるなか、SSL (Secure Sockets Layer)が誕生した。

SSL が画期的なのは、ブラウザベンダであった Netscape 社が開発したことで当初からブラウザに標準搭載され、セキュリティの予備知識を持たないユーザにもセキュアなインターネット環境を提供したことである。この結果、オンラインバンキング・オンラインショッピング等を利用するユーザ数が爆発的に増え、インターネットビジネスの隆盛につながっていったことは疑いの余地がない。IETF が定める正式名称 TLS (Transport Layer Security)よりも SSL の方がはるかに知名度があることはその証左といえるだろう。本ガイドラインが「SSL/TLS」と謳っているのもそのためである。

しかし、2010 年代に入ると急速に SSL の安全性は低下する。

SSL3.0 で使える暗号アルゴリズムは、2000 年の輸出規制緩和以前に定められたこともあって、RC4 と Triple DES 等であるが、これらに対する暗号解読手法の進展により、今では危殆化した暗号に位置づけられている。例えば、RC4 は無線 LAN の一方式である WEP (Wired Enhanced Privacy)でも使われていたが、2000 年代に現実的な解読方法[15][16]が見つかり WPA/WPA2 への移行が進められた。2013 年以降、SSL/TLS での RC4 利用に対しても様々な攻撃手法が提案[17]されている。Triple DES も、2016 年に Sweet32 とよばれる攻撃手法[18]が公表されたことを受け、2017 年 11 月に NIST は Triple DES の利用方法の見直しを実施する[19]とともに、利用終了期限を含めた今後のスケジュール検討に入る[20]ことを表明した。

また、2014 年に発表された POODLE (Padding Oracle On Downgraded Legacy Encryption)攻撃[21]は SSL3.0 の仕様上の脆弱性に起因する攻撃方法であったことから、SSL3.0 を利用不可にするしか回避策がなかった。

このため、この数年間でほぼ全てのブラウザで SSL3.0 を利用不可とする設定が行われている。

[15] https://iacr.org/archive/asiacrypt2005/390/390.pdf
[16] https://eprint.iacr.org/2007/120.pdf
[17] https://www.rc4nomore.com/
[18] https://sweet32.info/
[19] https://csrc.nist.gov/publications/detail/sp/800-67/rev-2/final
[20] https://csrc.nist.gov/news/2017/update-to-current-use-and-deprecation-of-tdea
[21] https://www.openssl.org/~bodo/ssl-poodle.pdf

最近では、より根本的な対策として 2.1.3 節で紹介したように、SSL3.0 の流れを汲む TLS1.0、TLS1.1、TLS1.2 から外れ、新しい方針で作られた TLS1.3 が誕生した。しかも、TLS1.3 は IETF での標準化前からブラウザ等への搭載が始まるなど、今までにない速いテンポで準備が進んでいる。

実際、NIST では、2020 年 1 月 1 日までに TLS1.2 をサポートすること、及び TLS1.3 をサポートし移行する計画を作ることを求めるガイドライン案[22]を公表している。

[22] https://csrc.nist.gov/publications/detail/sp/800-52/rev-2/draft

5. サーバ証明書の設定

サーバ証明書は、①クライアントに対して、情報を送受信するサーバが意図する相手（サーバの運営組織等）によって管理されるサーバであることを確認する手段を提供することと、②SSL/TLS による暗号通信を行うために必要なサーバの公開鍵情報をクライアントに正しく伝えること、の 2 つの役割を持っている。

これらの役割を正しく実行するために、5.1 節にサーバ証明書についての要求設定をまとめる。5.2 節以降にはサーバ証明書の設定を決める際の検討項目の概要を記す。

5.1 サーバ証明書についての要求設定

後述する 5.2 節以降の内容を踏まえ、サーバ証明書についての要求設定を以下のように定める。なお、本ガイドライン公開時点（2018 年 5 月）においては、推奨セキュリティ型の要求設定は高セキュリティ型と同様とする。

高セキュリティ型(推奨セキュリティ型)の要求設定では、少なくともハッシュ関数として SHA-256 が使えることが必須条件となることに注意されたい。例えば、SHA-256 が使えないブラウザ（クライアント）では、サーバ証明書の検証ができず、警告表示が出るか、当該サーバとの接続は不能となる。このことは、DSA や ECDSA を使う場合についても同様である。

一方、セキュリティ例外型の要求設定では、ハッシュ関数として SHA-1 の利用も許容しており、過去のシステムとの相互接続性は高い。ただし、SHA-1 を利用したサーバ証明書はパブリック認証局から発行してもらうことが出来なくなったので、自らプライベート認証局を運用しなければならなくなるなど、非常に運用管理の負荷がかかることを強く認識する必要がある。また、現在の主要ブラウザでは SHA-1 を使うサーバ証明書に対して無効化されていることに注意すること。

DSA については、5.3 節で示すように電子政府推奨暗号リストに選定されており、安全性上の問題はない。しかし、サーバ証明書としては現時点でほとんど利用されておらず、技術的にも RSA や ECDSA と比較して大きなメリットがあるとは言えないことから、本ガイドラインでは積極的には DSA の利用を勧めない[23]。

【高セキュリティ型の要求設定】

- 本ガイドライン公開時点（2018 年 5 月）で、多くの認証局から入手可能なサーバ証明書のうち、安全性が高いものを利用する。

サーバ証明書のアルゴリズムと鍵長	サーバ証明書の発行・更新を要求する際に生成する鍵情報（Subject Public Key Info）でのアルゴリズム（Subject Public Key Algorithm）と鍵長としては、以下のいずれかを必須とする。

[23] 本ガイドラインでは、DSA は利用しないことを要求設定の前提としているため、6 章の暗号スイートの設定からも DSA を利用する暗号スイートが除外されていることに留意されたい。

	● RSA（OID = 1.2.840.113549.1.1.1）で鍵長は 2048 ビット以上 ● 楕円曲線暗号で鍵長 256 ビット以上（NIST P-256 の場合の OID = 1.2.840.10045.3.1.7） また、認証局が発行するサーバ証明書での署名アルゴリズム（Certificate Signature Algorithm）と鍵長については、以下のいずれかを必須とする。 ● RSA 署名と SHA-256 の組合せ（sha256WithRSAEncryption; OID = 1.2.840.113549.1.1.11）で鍵長 2048 ビット以上 ● ECDSA と SHA-256 の組合せ（ecdsa-with-SHA256; OID = 1.2.840.10045.4.3.2）で鍵長 256 ビット（NIST P-256）以上
サーバ証明書の発行・更新時の鍵情報の生成	● サーバ証明書の発行・更新を要求する際には、既存の鍵情報は再利用せず、必ず新たに公開鍵と秘密鍵の鍵ペアを生成しなければならない ● 上記の指示をサーバ管理者への仕様書、運用手順書、ガイドライン等に明示しなければならない
クライアントでの警告表示の回避	● 当該サーバに接続することが想定されている全てのクライアントに対して、以下のいずれかの手段を用いて警告表示が出ないようにしなければならない ➢ パブリック認証局からサーバ証明書を入手する ➢ 警告表示が出るクライアントの利用を禁ずる措置を取る ➢ 5.4.2 節の例外規定に従って、信頼できるプライベート認証局のルート CA 証明書をインストールする

【推奨セキュリティ型の要求設定（高セキュリティ型の要求設定と同じ）】

● 本ガイドライン公開時点（2018 年 5 月）で、多くの認証局から入手可能なサーバ証明書のうち、安全性が高いものを利用する。

サーバ証明書の暗号アルゴリズムと鍵長	サーバ証明書の発行・更新を要求する際に生成する鍵情報（Subject Public Key Info）でのアルゴリズム（Subject Public Key Algorithm）と鍵長としては、以下のいずれかを必須とする。 ● RSA（OID = 1.2.840.113549.1.1.1）で鍵長は 2048 ビット以上 ● 楕円曲線暗号で鍵長 256 ビット以上（NIST P-256 の場合の OID = 1.2.840.10045.3.1.7） また、認証局が発行するサーバ証明書での署名アルゴリズム（Certificate Signature Algorithm）と鍵長については、以下のいずれかを必須とする。 ● RSA 署名と SHA-256 の組合せ（sha256WithRSAEncryption; OID = 1.2.840.113549.1.1.11）で鍵長 2048 ビット以上 ● ECDSA と SHA-256 の組合せ（ecdsa-with-SHA256; OID = 1.2.840.10045.4.3.2）で鍵長 256 ビット（NIST P-256）以上

サーバ証明書の発行・更新時の鍵情報の生成	● サーバ証明書の発行・更新を要求する際には、既存の鍵情報は再利用せず、必ず新たに公開鍵と秘密鍵の鍵ペアを生成しなければならない ● 上記の指示をサーバ管理者への仕様書、運用手順書、ガイドライン等に明示しなければならない
クライアントでの警告表示の回避	● 当該サーバに接続することが想定されている全てのクライアントに対して、以下のいずれかの手段を用いて警告表示が出ないようにするか、警告表示が出るブラウザはサポート対象外であることを明示しなければならない ➢ パブリック認証局からサーバ証明書を入手する ➢ 警告表示が出るクライアントの利用を禁ずる措置を取る ➢ 5.4.2 節の例外規定に従って、信頼できるプライベート認証局のルート CA 証明書をインストールする

【セキュリティ例外型の要求設定】

- 本ガイドライン公開時点（2018 年 5 月）で、多くの認証局から入手可能なサーバ証明書のうち、許容可能な水準以上の安全性を確保しているサーバ証明書で、最も相互接続性が高いものを利用する。なお、過去のシステム･ブラウザ等との相互接続性の確保の観点から、SHA-1 を利用するサーバ証明書がどうしても必要である場合には、パブリック認証局から発行してもらうことが出来なくなったので、自らプライベート認証局を運用しなければならなくなった。これは、SSL/TLS サーバの運用だけでなく、認証局の運用も含めて安全に管理する必要があることを意味し、非常に運用管理の負荷がかかることを強く認識する必要がある。このため、真にやむを得ない場合を除いては、SHA-1 を利用するサーバ証明書の利用は勧めない。
- セキュリティ例外型においては、楕円曲線暗号を利用したサーバ証明書の場合、十分な相互接続性が確保できるとは必ずしも言えないため、RSA の利用を勧める。

サーバ証明書の暗号アルゴリズムと鍵長	サーバ証明書の発行・更新を要求する際に生成する鍵情報（Subject Public Key Info）でのアルゴリズム（Subject Public Key Algorithm）と鍵長としては、以下のいずれかを必須とする。 ● RSA（OID = 1.2.840.113549.1.1.1）で鍵長は 2048 ビット以上 また、認証局が発行するサーバ証明書での署名アルゴリズム（Certificate Signature Algorithm）と鍵長については、以下のいずれかを必須とする。なお、SHA-1 との組合せは、真にやむを得ない場合を除いて、勧めない。 ● RSA 署名と SHA-256 の組合せ（sha256WithRSAEncryption; OID = 1.2.840.113549.1.1.11）で鍵長 2048 ビット以上 ● RSA 署名と SHA-1 の組合せ（sha1WithRSAEncryption; OID = 1.2.840.113549.1.1.5）で鍵長 2048 ビット以上

	※ 過去のシステム・ブラウザ等との相互接続性の確保を最優先するなら SHA-1 を利用したサーバ証明書を使うことも妨げるものではないが、非常に運用管理の負荷がかかることを強く認識しなければならない。 ※ また、現在の主要ブラウザでは SHA-1 を使うサーバ証明書に対して無効化されていることに注意すること。詳細については 8.3.1 節を参照のこと
サーバ証明書の発行・更新時の鍵情報の生成	● サーバ証明書の発行・更新を要求する際には、既存の鍵情報は再利用せず、必ず新たに公開鍵と秘密鍵の鍵ペアを生成しなければならない ● 上記の指示をサーバ管理者への仕様書、運用手順書、ガイドライン等に明示しなければならない
クライアントでの警告表示の回避	● 当該サーバに接続することが想定されている全てのクライアントに対して、以下のいずれかの手段を用いて警告表示が出ないようにするか、警告表示が出るブラウザはサポート対象外であることを明示しなければならない ➢ パブリック認証局からサーバ証明書を入手する ➢ 警告表示が出るクライアントの利用を禁ずる措置を取る ➢ 5.4.2 節の例外規定に従って、信頼できるプライベート認証局のルート CA 証明書をインストールする

5.2 サーバ証明書に記載されている情報

サーバ証明書には、表 9 に示す情報が記載されている。これらの情報は、証明書プロパティの「詳細」で見ることができる。

これらのうち、当該サーバ証明書を発行した認証局が「Issuer（発行者）」となり、当該認証局がサーバ証明書に施すアルゴリズムが「Certificate Signature Algorithm（署名アルゴリズム）」、実際の署名値が「Certificate Signature Value」として記載される。

SSL/TLS サーバを運用するものは「Subject（サブジェクトー発行対象）」となり、当該サーバ自身が利用する公開鍵の情報が「Subject Public Key Info（公開鍵情報）」として記載される。公開鍵情報には「Subject Public Key Algorithm（公開鍵を使う時の暗号アルゴリズム）」と「Subject's Public Key（実際の公開鍵の値）」が含まれており、その公開鍵をどのように使うかは「Certificate Key Usage（キー使用法）」に記載される。

例えば、Subject Public Key Algorithm に「RSA」、Certificate Key Usage に「Signing, Key Encipherment」とある場合には、Subject's Public Key に書かれた公開鍵は、対応する秘密鍵で作られた RSA 署名（Signing）の検証用途にも、セッション鍵を送付する RSA 暗号化（Key Encipherment）用途にも使えることを意味する。

表 9　サーバ証明書に記載される情報

証明書のバージョン	Version
シリアル番号	Serial Number
署名アルゴリズム	Certificate Signature Algorithm
発行者	Issuer
有効期間（開始～終了）	Validity (Not Before ～ Not After)
サブジェクト（発行対象）	Subject
（サブジェクトが使う）公開鍵情報[24]	Subject Public Key Info (Algorithm, Public Key Value)
拡張情報	Extensions
キー使用法	Certificate Key Usage
署名	Certificate Signature Value

5.3 サーバ証明書で利用可能な候補となる暗号アルゴリズム

本ガイドラインにおいて「サーバ証明書で利用可能な候補となる暗号アルゴリズム」とは、サーバ証明書の仕様に合致するものに採用されている「署名」と「ハッシュ関数」のうち、CRYPTREC 暗号リスト（2.2.1 節参照）にも掲載されているものとする。具体的には、表 10 に示した「署名」と「ハッシュ関数」である。

現在発行されているサーバ証明書は、大多数が RSA と SHA-256 との組合せによるものである。

また、RSA の鍵長が 2048 ビット以上なのに対し、処理性能の低下を避けるために鍵長 256 ビットの ECDSA を採用するケースも増えてきている。実際に、従来 RSA しかサーバ証明書を発行しなかった認証局でも、ECDSA に対応したサーバ証明書を発行するようになってきている。

表 10　サーバ証明書で利用可能な候補となる暗号アルゴリズム

技術分類	リストの種類	アルゴリズム名
署名	電子政府推奨暗号リスト	RSASSA PKCS#1 v1.5（RSA）
		DSA
		ECDSA
ハッシュ関数	電子政府推奨暗号リスト	SHA-256
	運用監視暗号リスト	SHA-1

[24] Windows の証明書プロパティでは『公開キー』と表記されているが、本文中では『公開鍵』で表記を統一する。

5.4 サーバ証明書で考慮すべきこと

5.4.1 信頼できないサーバ証明書の利用は止める

ブラウザなどをはじめとするサーバ証明書を検証するアプリケーションには、一定の基準に準拠した認証局の証明書（ルート CA 証明書）があらかじめ登録されており、これらの認証局（とその下位認証局）はパブリック認証局と呼ばれている。一般に、パブリック認証局が、第三者の立場から確認したサーバの運営組織等の情報を記載したサーバ証明書を発行し、ブラウザに予め搭載されたルート CA 証明書と組合せて検証を行うことで、サーバ証明書の信頼性を確保する。これにより、当該サーバ証明書の正当性が確認できれば、ブラウザは警告表示することなく、当該サーバへの接続を行う。

一方、証明書の発行プログラムさえあれば誰もがサーバ証明書を作ることができるが、これではサーバ構築者が“自分は正当なサーバ”であると自己主張しているに過ぎない。このようなサーバ証明書は“オレオレ証明書”ともいわれ、ブラウザでは当該サーバ証明書の正当性が確認できない“危険なサーバ”として警告が表示される。

本来、SSL/TLS における重要な役割の一つが接続するサーバの認証であり、その認証をサーバ証明書で行う仕組みである以上、“危険なサーバ”との警告表示が出るにもかかわらず、その警告を無視して接続するよう指示しなければならないサーバ構築の仕方をすべきではない。

5.4.2 ルート CA 証明書の安易な手動インストールは避ける

5.4.1 節のようにして発行されたサーバ証明書を利用した SSL/TLS サーバを“危険なサーバ”として認識させない手段として、当該サーバ証明書の正当性を確認するためのルート CA 証明書を、ブラウザ（クライアント）の「信頼できるルート CA」に手動でインストールする方法がある。

しかし、安易に「信頼できるルート CA」として手動インストールをすることは、“危険なサーバ”との警告を意図的に無視することにつながる。また、5.4.1 節に記載したパブリック認証局のルート CA 証明書とは異なり、これら手動インストールしたルート CA 証明書はブラウザベンダによって管理されていない。このため、万が一、当該ルート CA 証明書の安全性に問題が生じた場合でも、ブラウザベンダによって自動的に無効化されることはなく、インストールした当該ルート CA 証明書を利用者自身が手動で削除する必要がある。もし、削除を怠ると不正なサーバ証明書を誤信するリスクが増大する。

したがって、ルート CA 証明書の手動インストールは原則として避けるべきであり、ましてや利用者に対して手動インストールを求めるような運用をすべきではない。

例外的にルート CA 証明書の手動インストールを行う必要がある場合には、ルート CA 証明書の安全性に問題が生じた場合にインストールを勧めた主体によって、利用者のブラウザから当該ルート CA 証明書の無効化や削除ができるようにする等、インストールした利用者に対して具体的に責任を負うことができる体制を整えるべきである。

例えば、企業・団体等が自身の管理する端末に対して、当該組織が自前で構築した、言わばプライベートなルート CA 証明書をシステム管理部門等の管理下でインストールし、また当該ルー

ト CA 証明書の安全性に問題が生じた場合には、速やかに当該部門が各端末に対して当該ルート CA 証明書を無効化する措置を講ずることができるような体制である。具体的には、組織等において一定のポリシーに基づいて端末管理を行っている場合、管理システムなどから各端末にルート CA 証明書を自動更新（インストールおよび削除）する仕組みを提供するなどである。一例として Windows クライアントに対して Active Directory から自動更新する場合の構成例を Appendix D.2 に示す。

このような仕組みを用いて各端末にインストールされたルート CA 証明書の安全性に問題が生じた場合には、当該組織の責任において、当該ルート CA 証明書を速やかに自動削除するなどの無効化する措置を講じなければならない。

5.4.3 サーバ証明書で利用すべき鍵長

署名の安全性は鍵長にも大きく影響される。通常、同じアルゴリズムであれば、鍵長が長いほど安全性を高くすることができる。ただし、あまりにも長くしすぎると処理時間が過大にかかるようになり、実用性を損なうことにもつながる。

CRYPTREC では、素因数分解問題の困難性に関する調査研究に基づいて RSA の安全性に関する見積りを作成している。これによれば、RSA 2048 ビットを素因数分解するのにおおむね 10^{25}～10^{27} FLOPS 程度の計算量が必要との見積もりを出しており、劇的な素因数分解手法の発見がない限り、計算機性能の向上を考慮しても世界最速の計算機が 1 年かけて解読可能となるのは 2035 年以降と予想している。また、楕円曲線上の離散対数問題の困難性に関する調査研究も行われており、ECDSA 192 ビットを解くのにおおむね 10^{24}～10^{25} FLOPS 程度の計算量が必要と見積もっている。詳細については、CRYPTREC Report 2016[25] 図 3.3、図 3.4 を参照されたい。

以上の報告と、内閣官房情報セキュリティセンター（現：内閣サイバーセキュリティセンター）が公表している「政府機関の情報システムにおいて使用されている暗号アルゴリズム SHA-1 及び RSA1024 に係る移行指針[26]」を踏まえれば、本ガイドライン公開時点（2018 年 5 月）でサーバ証明書が利用すべき鍵長は、RSA は 2048 ビット以上、ECDSA は 256 ビット以上が妥当である。

5.4.4 サーバ証明書を発行・更新する際に新しい鍵情報を生成する重要性

サーバ証明書を取得する際に、公開鍵と秘密鍵の鍵ペアの生成・運用・管理が正しく行われないと、暗号化された通信データが第三者に復号されてしまうなどの問題が発生するリスクがある。例えば、とりわけ危険なのは、サーバ機器の出荷時に設定されているデフォルト鍵や、デフォルト設定のまま生成した鍵ペアを利用した場合、意図せず第三者と同じ秘密鍵を共有してしまう恐れがある。

また、何らかの理由により秘密鍵が漏えいした恐れがあり、サーバ証明書を再発行する必要性に迫られた時に、前回使用した CSR（Certificate Signing Request：サーバ証明書を発行するための署名要求）を使い回すと、同じ公開鍵と秘密鍵の鍵ペアのまま新しいサーバ証明書が作られるの

[25] http://www.cryptrec.go.jp/report/cryptrec-rp-0002-2016.pdf
[26] https://www.nisc.go.jp/active/general/pdf/angou_ikoushishin.pdf

で、古いサーバ証明書を失効させ、新しいサーバ証明書を再発行することの意味がなくなる。

こうしたリスクを防ぐためには、サーバ管理者は、サーバ証明書を取得・更新する際に既存の鍵ペアを使い回すことを厳に慎み、毎回新しく生成した鍵ペアを使った CSR でサーバ証明書を取得・更新しなければならない。

【コラム②】DNS の CAA (Certification Authority Authorization) リソースレコード

Web サイト管理者は、DNS リソースレコードの一種である CAA に、1 つ以上の認証局事業者（の所有する DNS ドメインネーム）を記載する事により、所有する DNS ドメインネームに対し証明書を発行可能な認証局事業者を指定できる。

DNS の CAA リソースレコード(以下単に CAA)は 2013 年に RFC 6844 として定められたものの、広くは使われていなかった。しかしながら、2017 年 9 月に CA 及びブラウザベンダの業界団体である「CA/ブラウザフォーラム」が、認証局事業者に対し CAA の確認を必須化した事により、徐々に利用されつつある。なお、SSL Pulse[27]によると、CAA の普及率は 2018 年 4 月時点で 3%超となっている。

CAA の第一の目的は、他の認証局事業者の意図しない証明書誤発行を削減する事である。証明書発行後に、その証明書が適切か否かを判断する為の TLSA リソースレコード（RFC 6698 記載の DANE で利用される）とは目的が異なる点に注意されたい。

CAA の設定は、①証明書を発行する認証局事業者のドメインネームを、②DNS ドメインネーム所有者が、③所定のタグの値へ記載する、事により行われる。本コラムでは、以上の三つのプロセスについて、順に説明を行う。

①証明書を発行する認証局事業者のドメインネームを、各認証局事業者の案内ページ等[28]で確認する。

②DNS リソースレコードを管理している主体(例えば DNS サービスプロバイダ)に、CAA を設定するよう依頼を行う。設定方法は各 DNS サービスプロバイダの案内ページ等を参照する。

③証明書を発行する認証局事業者のドメインネームを issue タグの値へ記載する。ワイルドカード証明書を発行する認証局事業者を別に指定したい時は issuewild タグの値へ記載する。なお、ワイルドカード証明書の発行を完全に禁止したい場合は、issuewild タグの値へ空文字（""）を記載する。

ここで、CAA に記載がない場合は、任意の認証局事業者が証明書を発行できることとなる。もっとも、そのドメインに CAA が設定されていなくても、CNAME や上位ドメインに CAA が設定されている場合は、その設定が反映されるので注意が必要となる。

現状の仕様では、issuewild タグの値で明示的に禁止していない場合は、issue タグの値で指定した認証局事業者は、ワイルドカード証明書も発行することが可能となっている。しかし、issue タグに指定された認証局事業者がワイルドカード証明書を発行可能である事は、直感的でないとの見方もあり、2018 年 4 月現在 CA/ブラウザフォーラムにて改定が検討されており、変更される可能性がある点は注意が必要となる。

[27] https://www.ssllabs.com/ssl-pulse/

[28] CA/ブラウザフォーラムに登録されたドメインネーム一覧は以下で確認できる。
https://ccadb-public.secure.force.com/mozilla/AllCAAIdentifiersReport

6. 暗号スイートの設定

暗号スイートは「鍵交換__署名__暗号化__ハッシュ関数」の組によって構成される。

例えば、「TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384」であれば、鍵交換には「DHE」、署名には「RSA」、暗号化には「鍵長 256 ビット GCM モードの Camellia（CAMELLIA_256_GCM）」、ハッシュ関数には「SHA-384」が使われることを意味する。「TLS_RSA_WITH_AES_128_CBC_SHA」であれば、鍵交換と署名には「RSA」、暗号化には「鍵長 128 ビット CBC モードの AES（AES_128_CBC）」、ハッシュ関数には「SHA-1」が使われることを意味する。

実際の SSL/TLS 通信においては、サーバとクライアント間での暗号化通信前の事前通信（ハンドシェイク）時に、両者の合意により一つの暗号スイートを選択する。暗号スイートが選択された後は、選択された暗号スイートに記載の鍵交換、署名、暗号化、ハッシュ関数の方式により SSL/TLS における各種処理が行われる。つまり、SSL/TLS における安全性にとって、暗号スイートをどのように設定するかが最も重要なファクタであることを意味する。

6.1 節に暗号スイートについての要求設定をまとめる。6.2 節から 6.4 節では暗号スイートの設定を決めるうえでの重要な検討項目の概要を記す。

6.1 暗号スイートについての要求設定

一般に、暗号スイートの優先順位の上位から順にサーバとクライアントの両者が合意できる暗号スイートを見つけていく。このため、暗号スイートの選択のみならず、優先順位の設定が重要となる。

その際、多くのブラウザ（クライアント）との相互接続性を確保するためには、多くの製品に共通して実装されている暗号スイートを設定することが不可欠である点に注意する必要がある。一方、高い安全性を実現するためには、比較的新しい製品でしか実装されていないが、高い安全性を持つ暗号アルゴリズムで構成される暗号スイートを設定する必要がある。

上記の点と 6.2 節～6.4 節での内容を踏まえ、本ガイドラインでは、暗号スイートについての要求設定を以下のように定める。なお、本節では、要求設定の概要についてのみ記載する。詳細な要求設定については、各々の該当節を参照すること。

【高セキュリティ型の要求設定】

高セキュリティ型の要求設定の概要は以下の通り。詳細な要求設定は 6.5.1 節を参照のこと。

- 以下の条件を満たす暗号スイートを選定する。
  - CRYPTREC 暗号リストに掲載されているアルゴリズムのみで構成される。
  - 暗号化として 128 ビット安全性以上を有する。
  - 安全性向上への寄与が高いと期待されることから、認証付き秘匿モードを採用する。
  - Perfect Forward Secrecy（後述）の特性を満たす。

- ただし、本ガイドラインではサーバ証明書で DSA を利用しないことを要求設定の前提としている（5.1 節参照）ため、DSA を含む暗号スイートは選定しない。

- 暗号スイートの優先順位は以下の通りとする。
  - 選定した暗号スイートをグループ α とグループ β に分類し、安全性の高いグループを優先する。グループ分けの基準はブロック暗号の鍵長によるものとする。
- 上記以外の暗号スイートは利用除外とする。
- 鍵交換で DHE を利用する場合には鍵長 2048 ビット以上、ECDHE を利用する場合には鍵長 256 ビット以上の設定を必須とする。

【推奨セキュリティ型の要求設定】

推奨セキュリティ型の要求設定の概要は以下の通り。詳細な要求設定は 6.5.2 節を参照のこと。

- 以下の条件を満たす暗号スイートを選定する。
  - CRYPTREC 暗号リストに掲載されているアルゴリズムのみで構成される。
  - 暗号化として 128 ビット安全性以上を有する。
  - ただし、本ガイドラインではサーバ証明書で DSA を利用しないことを要求設定の前提としている（5.1 節参照）ため、DSA を含む暗号スイートは選定しない。
- 暗号スイートの優先順位は以下の通りとする。
  - 選定した暗号スイートを、安全性と実用性とのバランスの観点に立って、グループ A、グループ B、・・・、グループ F とグループ分けをする。
  - 以下の条件でグループごとの優先順位を付ける。
    - 本ガイドライン公開時点（2018 年 5 月）では、通常の利用形態において、128 ビット安全性があれば十分な安全性を確保できることから 128 ビット安全性を優先する。ただし、256 ビット安全性を優先することを妨げるものではない。
    - 鍵交換に関しては、Perfect Forward Secrecy の特性の有無と実装状況に鑑み、DHE、次いで RSA の順番での優先順位とする。
- 上記以外の暗号スイートは利用除外とする。
- 鍵交換で DHE を利用する場合には鍵長 1024 ビット以上[29]、ECDHE/ECDH を利用する場合には鍵長 256 ビット以上、RSA を利用する場合には鍵長 2048 ビット以上の設定を必須とする。

【セキュリティ例外型の要求設定】

セキュリティ例外型の要求設定の概要は以下の通り。詳細な要求設定は 6.5.3 節を参照のこと。

[29] ①暗号解読以外の様々な秘密鍵の漏えいリスクを考えれば PFS の特性を優先させるほうが望ましい、②6.3.3 節に示すように DHE を利用する場合、多くの場合で 1024 ビットが選択される環境である、②DHE であれば秘密鍵漏えいの影響が当該セッション通信のみに限定される、ことを踏まえ、本ガイドラインの発行時点での DHE の推奨鍵長は 1024 ビット以上とする

- 以下の条件を満たす暗号スイートを選定する。
  - CRYPTREC 暗号リストに掲載されているアルゴリズムのみで構成される。
  - ただし、今までほとんど使われていない楕円曲線暗号と Triple DES や RC4 の組合せを今後使っていく積極的な理由は見いだせないことから、楕円曲線暗号と Triple DES、RC4 の組み合わせは選定しない。
  - また、本ガイドラインではサーバ証明書で DSA を利用しないことを要求設定の前提としている（5.1 節参照）ため、DSA を含む暗号スイートも選定しない。
- 暗号スイートの優先順位は以下の通りとする。
  - 選定した暗号スイートを、安全性と実用性とのバランスの観点に立って、グループ A、グループ B、・・・とグループ分けをする。なお、グループ A からグループ F までは推奨セキュリティ型と同様であり、推奨セキュリティ型での優先順位のつけ方を適用する。
- 上記以外の暗号スイートは利用除外とする。
- 鍵交換で DHE を利用する場合には鍵長 1024 ビット以上、ECDHE/ECDH を利用する場合には鍵長 256 ビット以上、RSA を利用する場合には鍵長 2048 ビット以上の設定を必須とする。

6.2 暗号スイートで利用可能な候補となる暗号アルゴリズム

本ガイドラインにおいて「暗号スイートで利用可能な候補となる暗号アルゴリズム」とは、SSL/TLS 用の暗号スイートとして IETF で規格化されたものに採用されている暗号アルゴリズムのうち、CRYPTREC 暗号リスト（2.2.1 節参照）にも掲載されているものとする。具体的には、表 11 に示した暗号アルゴリズムである。

表 11　暗号スイートで利用可能な候補となる暗号アルゴリズム

暗号スイートでの標記	CRYPTREC 暗号リストでの標記		
	技術分類	リストの種類	アルゴリズム名
鍵交換	鍵共有・守秘	電子政府推奨暗号リスト	DH（Ephemeral DH を含む）
			ECDH（Ephemeral DH を含む）
		運用監視暗号リスト	RSAES PKCS#1 v1.5（RSA）
署名	署名	電子政府推奨暗号リスト	RSASSA PKCS#1 v1.5（RSA）
			DSA
			ECDSA
暗号化	128 ビットブロック暗号	電子政府推奨暗号リスト	AES（鍵長 128 ビット、256 ビット）
			Camellia（鍵長 128 ビット、256 ビット）
	暗号利用モード	電子政府推奨暗号リスト	CBC
			GCM
ハッシュ関数	ハッシュ関数	電子政府推奨暗号リスト	SHA-256
			SHA-384
		運用監視暗号リスト	SHA-1

暗号スイートで利用可能な候補となる暗号アルゴリズム（続）

以下は SSL3.0 でのみ利用可			
暗号化	64 ビット ブロック暗号	運用監視暗号リスト	3-key Triple DES
	ストリーム 暗号	運用監視暗号リスト	128-bit RC4

なお、Triple DES と RC4 は運用監視暗号リストに掲載されており、また安全でかつ高速な共通鍵暗号として AES や Camellia が利用可能であることから、本ガイドラインでは TLS1.0 以上の場合には Triple DES と RC4 の利用は認めない。

6.3 鍵交換で考慮すべきこと

SSL/TLS の仕様では、実際のデータを暗号化する際に利用する“セッション鍵”はセッションごとに（あるいは任意の要求時点で）更新される。したがって、何らかの理由により、ある時点でのセッション鍵が漏えいした場合でも、当該セッション以外のデータは依然として保護された状態にある。

一方、セッション鍵は暗号通信を始める前にサーバとクライアントとで共有しておく必要があるため、事前通信（ハンドシェイク）の段階でセッション鍵を共有するための処理が行われる。この処理のために使われるのが、表 11 での「鍵共有・守秘」に掲載されている暗号アルゴリズムである。

6.3.1 秘密鍵漏えい時の影響範囲を狭める手法の採用（Perfect Forward Secrecy の重要性）

秘密鍵が漏えいする原因は暗号アルゴリズムの解読によるものばかりではない。むしろ、プログラムなどの実装ミスや秘密鍵の運用・管理ミス、あるいはサイバー攻撃やウイルス感染によるものなど、暗号アルゴリズムの解読以外が原因となって秘密鍵が漏えいする場合のほうが圧倒的に多い。

過去には、OpenSSL Heartbleed Bug や Dual_EC_DRBG の脆弱性などが原因による秘密鍵の漏えいといった事件も起きており、“秘密鍵が漏えいする”リスクそのものは決して無視できるものではない。スノーデン事件でも話題になったように、秘密鍵の運用・管理そのものに問題がある場合も想定される。

上述した通り、SSL/TLS では、毎回変わるセッション鍵をサーバとクライアントが共有することでセッションごとに違った秘密鍵を使って暗号通信をしており、仮にある時点でのセッション鍵が漏えいした場合でも当該セッション以外のデータは依然として保護されている。

しかし、多くの場合、セッション鍵の交換には固定の鍵情報を使って行っている。このため、どんな理由であれ、もし仮に鍵交換で使う暗号アルゴリズムの“秘密鍵”が漏えいした場合、当該秘密鍵で復号できるセッション鍵はすべて漏えいしたことと同義となる。つまり、SSL/TLS で

の通信データをためておき、年月が経って、当時の鍵交換で使った暗号アルゴリズムの“秘密鍵”が入手できたならば、過去にさかのぼって、ためておいた通信データの中身が読み出せることを意味している。

そこで、過去の SSL/TLS での通信データの秘匿を確保する観点から、鍵交換で使った暗号アルゴリズムの“秘密鍵”に毎回異なる乱数を付加することにより、見かけ上、毎回異なる秘密鍵を使ってセッション鍵の共有を行うようにする方法がある。これによって、仮に鍵交換で使う暗号アルゴリズムの“秘密鍵”が何らかの理由で漏えいしたとしても、当該セッション鍵の共有のために利用した乱数がわからなければ、当該セッション鍵そのものは求められず、過去に遡及して通信データの中身が読まれる危険性を回避することができる。

このような性質のことを、Perfect Forward Secrecy、または単に Forward Secrecy と呼んでいる。なお、本ガイドラインでは Perfect Forward Secrecy（あるいは PFS）に統一して呼ぶこととする。

現在の SSL/TLS で使う暗号スイートの中で、Perfect Forward Secrecy の特性を持つのは Ephemeral DH と Ephemeral ECDH と呼ばれる方式であり、それぞれ DHE、ECDHE と表記される。

6.3.2 鍵交換で利用すべき鍵長

5.4.3 節で述べたことと同様、鍵交換においても、鍵長を長くすれば処理時間や消費リソースなども増えるため、安全性と処理性能、消費リソースなどのトレードオフを考えて適切な鍵長を選択する必要がある。

例えば、処理性能や消費リソースの制約が厳しい組込み機器などの場合、鍵長 4096 ビットの RSA 暗号を利用して得られるメリットよりもデメリットの方が大きくなる可能性がある。CRYPTREC の見積もりでは、劇的な素因数分解手法の発見がない限り、計算機性能の向上を考慮しても世界最速の計算機が 1 年かけて鍵長 2048 ビットの RSA を解読可能となるのは 2035 年以降と予想している。また、NIST SP800-57 では鍵長 2048 ビットは 2030 年までは利用可とされている（2.2.2 節 表 5 参照）。したがって、2030 年を超えて利用することを想定していないシステムやサービスであれば、2048 ビットより大きい鍵長を使うメリットは乏しいといえる。

内閣官房情報セキュリティセンター（現：内閣サイバーセキュリティセンター）が公表している「政府機関の情報システムにおいて使用されている暗号アルゴリズム SHA-1 及び RSA1024 に係る移行指針」、並びに CRYPTREC が公開している公開鍵暗号についての安全性予測を踏まえれば、本ガイドライン公開時点（2018 年 5 月）での鍵交換で利用すべき鍵長は、RSA は 2048 ビット以上、ECDH/ECDHE は 256 ビット以上が妥当である。なお、RSA に関しては、サーバ証明書の申請段階で鍵長 2048 ビット以上を設定することで実現する。

6.3.3 DHE/ECDHE での鍵長の設定状況についての注意

鍵交換について、暗号スイート上は鍵長の規定がない。このため、同じ暗号スイートを使っていても、利用可能な鍵長は製品依存になっていることに注意する必要がある。特に、鍵交換で RSA を使う場合と、DHE や ECDHE/ECDH を使う場合とでは、鍵長の扱いが全く異なるので、それぞれについて適切な設定を行っておく必要がある。

RSA での鍵交換を行う場合にはサーバ証明書に記載された公開鍵を使うことになっており、本ガイドラインの発行時点では鍵長 2048 ビットの公開鍵がサーバ証明書に通常記載されている。このことは、RSA での鍵交換を行う場合、サーバ証明書を正当に受理する限り、どのサーバもブラウザも当該サーバ証明書によって利用する鍵長が 2048 ビットにコントロールされていることを意味する。例え鍵長 2048 ビットの RSA が使えないブラウザがあったとしても、鍵交換が不成立・通信エラーになるだけであり、2048 ビット以外の鍵長が使われることはない。

つまり、RSA での鍵交換に関しては、サーバ証明書の発行時に利用する鍵長を正しく決め、その鍵長に基づくサーバ証明書を発行してもらえばよく、ほとんどの場合、サーバやブラウザ等に特別な設定をする必要はない。

一方、DHE、ECDH/ECDHE については、利用する鍵長がサーバ証明書で明示的にコントロールされるのではなく、個々のサーバやブラウザでの鍵パラメータの設定によって決められる。このため、どの鍵長が利用されるかは、使用する製品での鍵パラメータの設定状況に大きく依存する。例えば、デフォルトで使用する鍵長が製品やバージョンによって異なることが知られており、2013 年夏頃までは鍵長 1024 ビットの DHE しか使えない製品やバージョンも少なくなかった。有名なところでは、Apache 2.4.6 以前、Java 7（JDK7）以前、Windows Server 2012 などがある。

図 8 の 2017 年 1 月の Alexa の調査結果[30]によれば、約 47 万の主要なサイトについて、DHE が利用できるのは約 55.7%であり、そのうちの約 64.7%（全体では約 36.0%）が鍵長 2048 ビットを採用している。一方、ECDHE が利用できるのは約 92.2%であり、そのうちの約 92.7%（全体では約 85.4%）が鍵長 256 ビットを採用している。

このことは、DHE を利用した場合は鍵長 2048 ビットが、ECDHE を利用した場合は鍵長 256 ビットが採用される可能性が極めて高いことを意味している。

DHE で鍵長 2048 ビットとして使う場合には、鍵長 2048 ビットをサポートしているバージョンを使ったうえで、デフォルトで使用可となっているか、もしくは使用可のオプション設定を行うことが必要である。

【明示的に鍵長 2048 ビットを指定できる代表例】

- OpenSSL
- Apache 2.4.7 以降
- lighttpd 1.4.29 以降
- nginx
- Java 8 以降

【明示的に鍵長を指定できるが、鍵長 2048 ビットをサポートしていない代表例】

- Apache 2.4.6 以前
- Java 7 以前

例えば、Java 7 以前では DHE で扱える鍵長は 64 ビット刻みで 512 ビットから 1024 ビットまでである。これらの製品を利用する場合には、必ず鍵長を 1024 ビットに指定して利用すること。

[30] https://securitypitfalls.wordpress.com/2017/04/17/january-2017-scan-results/

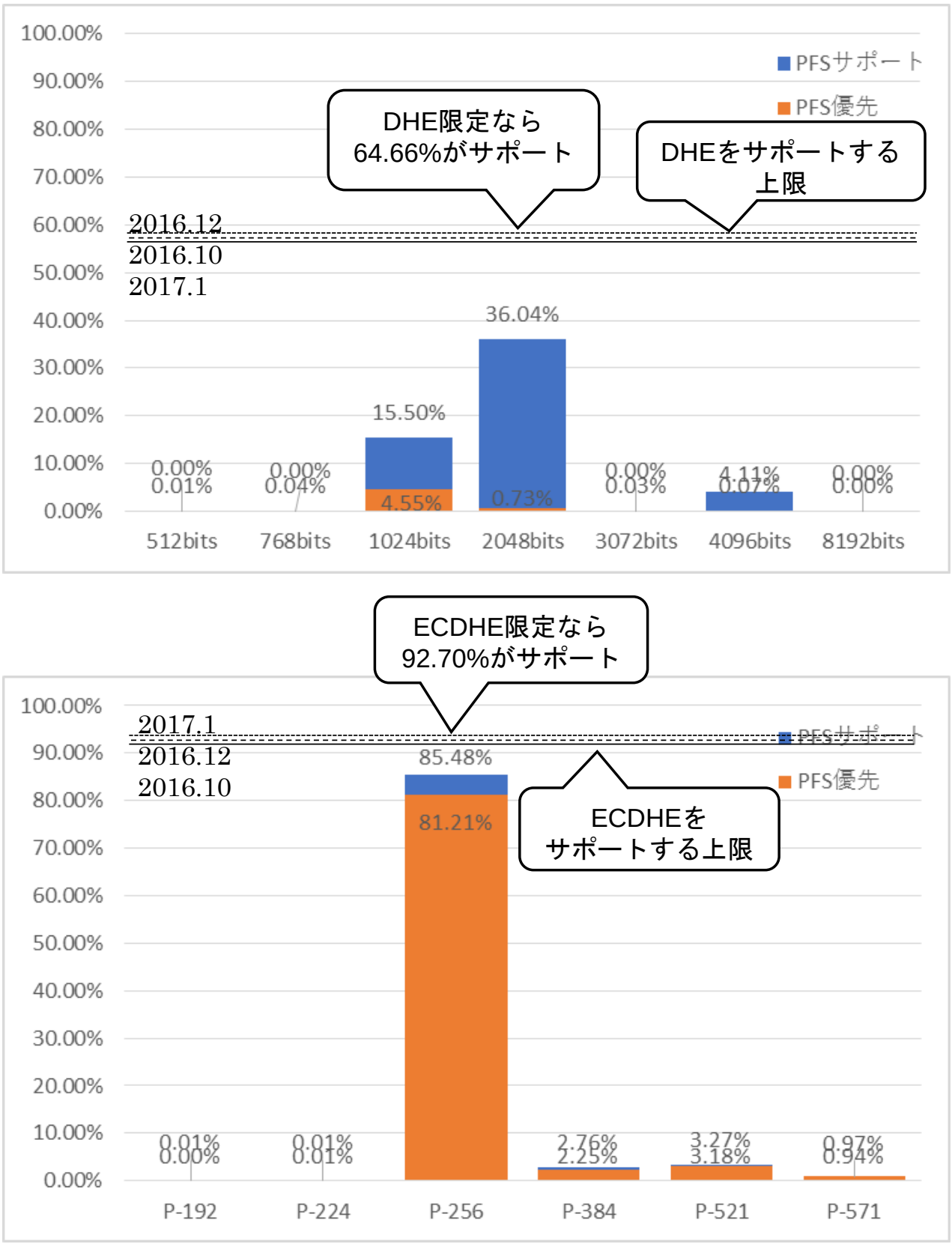


図 8　DHE/ECDHE の鍵長の設定状況（Alexa の調査結果を加工）

【明示的に鍵長を指定できない代表例】

- Apache Tomcat
- Microsoft IIS

これらについては、DHE の鍵長を指定することができず、クライアント側からの指定により 1024 ビット等の弱い鍵パラメータが使われる可能性がある。例えば、サーバ側の設定が鍵長 2048 ビット対応可能だったとしても、ブラウザ（クライアント）側が鍵長 2048 ビットに対応していない場合には、サーバ側は鍵長 1024 ビットを自動的に選択することに注意を要する。

この点は、RSA で鍵交換を行う場合とは大きく事情が異なるため、これらの製品を使う場合には、DHE を含む暗号スイートは選択せず、ECDHE または RSA を含む暗号スイートを使うように設定すべきである。

6.4 暗号スイートについての実装状況

SSL/TLS 用の暗号スイートは IETF で規格化されており、任意に暗号アルゴリズムを選択して「鍵交換＿署名＿暗号化＿ハッシュ関数」の組を自由に作れるわけではない。また、IETF で規格化されている暗号スイートだけでも数多くあるため、実際の製品には実装されていない暗号スイートも多い。

多くの製品に共通して実装されている暗号スイートを設定すれば、相互接続性を広く担保できる可能性が高まる。一方、特定の製品のみに実装されている暗号スイートだけを設定すれば、意図的に当該製品間での接続に限定することができる。

6.5 暗号スイートについての詳細な要求設定

本節では、6.1 節での要求設定の概要に基づき、各々の詳細な要求設定を以下に示す。

なお、鍵交換に PSK または KRB が含まれる暗号スイートは、サーバとクライアントの両方で特別な設定をしなければ利用することができないため、本ガイドラインの対象外とする。

6.5.1 高セキュリティ型での暗号スイートの詳細要求設定

6.1 節の条件を踏まえて、表 12 の通り、選定した暗号スイートをグループ α とグループ β に分類する。グループ分けの基準はブロック暗号の鍵長によるものとし、安全性の高いグループをグループ α に割り当て、優先して設定する。

なお、グループ内での暗号スイートから全部または一部を選択して設定するが、その際の優先順位は任意に定めてよい。また、グループ β の暗号スイートについては選択しなくてもよい。

「除外事項」は設定で除外すべき暗号スイートを示したものである[31]。

[31] 高セキュリティ型の暗号スイート設定では、TLS1.2 でのサポートが必須と指定されている暗号スイート AES128-SHA を利用した通信が接続不可となることに留意されたい

表 12　高セキュリティ型での暗号スイートの要求設定（基本）

グループ α	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x00,0x9F)
	TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384 (0xC0, 0x7D)
グループ β	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x00,0x9E)
	TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256 (0xC0,0x7C)
設定すべき鍵長	鍵交換で DHE を利用する場合には鍵長 2048 ビット以上の設定を必須とする。なお、DHE の鍵長を明示的に設定できない製品を利用する場合には、DHE を含む暗号スイートは選定すべきではない
高セキュリティ型での除外事項	グループ α、グループ β、表 13 以外のすべての暗号スイートを利用除外とする

表 13　高セキュリティ型での暗号スイートの要求設定（楕円曲線暗号の追加分）

グループ α への追加または代替	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xC0,0x2C)
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xC0,0x30)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384 (0xC0,0x87)
	TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384 (0xC0,0x8B)
グループ β への追加または代替	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xC0,0x2B)
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xC0,0x2F)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256 (0xC0,0x86)
	TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256 (0xC0,0x8A)
設定すべき鍵長	鍵交換で ECDHE を利用する場合には鍵長 256 ビット以上の設定を必須とする

6.5.2　推奨セキュリティ型での暗号スイートの詳細要求設定

6.1 節の条件を踏まえて、表 14 の通り、選定した暗号スイートをグループ A、グループ B、・・・とグループ分けをする。グループ分けの基準は安全性と実用性とのバランスの観点に立って行い、優先設定する順番としてグループ A から順に割り当てることを推奨する。なお、256 ビット安全性を優先することを妨げるものではなく、その場合には、グループ D、グループ A、グループ E、グループ B、グループ F、グループ C の順番に優先することを推奨する。

グループ内での暗号スイートから全部または一部を選択して設定するが、その際の優先順位は任意に定めてよい。また、グループ C 以降の暗号スイートについては選択しなくてもよい。

（RFC 必須）は、TLS1.2 を規定する RFC においてサポートが必須と指定されている暗号スイートであり、不特定多数からのアクセスを想定する SSL/TLS サーバにおいては利用可に設定することが推奨される暗号スイートである[32]。

また、「除外事項」は設定で除外すべき暗号スイートを示したものである。

[32] TLS1.1 及び TLS1.0 でのサポートが必須と指定されている暗号スイートは Triple DES を利用するものである。しかし、推奨セキュリティ型を適用する SSL/TLS サーバが接続相手として対象とするブラウザは、BEAST 攻撃等に対するセキュリティパッチが適用されているブラウザであることを考慮すれば、AES が利用可能であり、6.5.2 節の設定であっても事実上問題がないと判断した

表 14　推奨セキュリティ型での暗号スイートの要求設定（基本）

グループ A	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x00,0x9E)
	TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256 (0xC0,0x7C)
	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x00,0x67)
	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256 (0x00,0xBE)
	TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x00,0x33)
	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA (0x00,0x45)
グループ B	TLS_RSA_WITH_AES_128_GCM_SHA256 (0x00,0x9C)
	TLS_RSA_WITH_CAMELLIA_128_GCM_SHA256 (0xC0,0x7A)
	TLS_RSA_WITH_AES_128_CBC_SHA256 (0x00,0x3C)
	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256 (0x00,0xBA)
	TLS_RSA_WITH_AES_128_CBC_SHA (0x00,0x2F) （RFC 必須）
	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (0x00,0x41)
グループ C	該当なし
グループ D	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x00,0x9F)
	TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384 (0xC0, 0x7D)
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x00,0x6B)
	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256 (0x00,0xC4)
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x00,0x39)
	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x00,0x88)
グループ E	TLS_RSA_WITH_AES_256_GCM_SHA384 (0x00,0x9D)
	TLS_RSA_WITH_CAMELLIA_256_GCM_SHA384 (0xC0,0x7B)
	TLS_RSA_WITH_AES_256_CBC_SHA256 (0x00,0x3D)
	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256 (0x00,0xC0)
	TLS_RSA_WITH_AES_256_CBC_SHA (0x00,0x35)
	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (0x00,0x84)
グループ F	該当なし
設定すべき鍵長	鍵交換で DHE を利用する場合には鍵長 1024 ビット以上 、RSA を利用する場合には鍵長 2048 ビット以上の設定を必須とする。なお、DHE の鍵長を明示的に設定できない製品を利用する場合には、DHE を含む暗号スイートは選定すべきではない
推奨セキュリティ型での除外事項	グループ A～グループ F 及び表 15 以外のすべての暗号スイートを利用除外とする

表 15　推奨セキュリティ型での暗号スイートの要求設定（楕円曲線暗号の追加分）

グループ A への追加または代替	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xC0,0x2B)
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xC0,0x2F)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256 (0xC0,0x86)
	TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256 (0xC0,0x8A)

	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xC0,0x23)
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xC0,0x27)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256 (0xC0,0x72)
	TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256 (0xC0,0x76)
	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xC0,0x09)
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xC0,0x13)
グループ C への追加	TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256 (0xC0,0x2D)
	TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256 (0xC0,0x31)
	TLS_ECDH_ECDSA_WITH_CAMELLIA_128_GCM_SHA256 (0xC0,0x88)
	TLS_ECDH_RSA_WITH_CAMELLIA_128_GCM_SHA256 (0xC0,0x8C)
	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256 (0xC0,0x25)
	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256 (0xC0,0x29)
	TLS_ECDH_ECDSA_WITH_CAMELLIA_128_CBC_SHA256 (0xC0,0x74)
	TLS_ECDH_RSA_WITH_CAMELLIA_128_CBC_SHA256 (0xC0,0x78)
	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA (0xC0,0x04)
	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA (0xC0,0x0E)
グループ D への追加または代替	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xC0,0x2C)
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xC0,0x30)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384 (0xC0,0x87)
	TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384 (0xC0,0x8B)
	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xC0,0x24)
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xC0,0x28)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_CBC_SHA384 (0xC0,0x73)
	TLS_ECDHE_RSA_WITH_CAMELLIA_256_CBC_SHA384 (0xC0,0x77)
	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xC0,0x0A)
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xC0,0x14)
グループ F への追加	TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384 (0xC0,0x2E)
	TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384 (0xC0,0x32)
	TLS_ECDH_ECDSA_WITH_CAMELLIA_256_GCM_SHA384 (0xC0,0x89)
	TLS_ECDH_RSA_WITH_CAMELLIA_256_GCM_SHA384 (0xC0,0x8D)
	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384 (0xC0,0x26)
	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384 (0xC0,0x2A)
	TLS_ECDH_ECDSA_WITH_CAMELLIA_256_CBC_SHA384 (0xC0,0x75)
	TLS_ECDH_RSA_WITH_CAMELLIA_256_CBC_SHA384 (0xC0,0x79)
	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA (0xC0,0x05)
	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA (0xC0,0x0F)
設定すべき鍵長	鍵交換で ECDHE または ECDH を利用する場合には鍵長 256 ビット以上の設定を必須とする

6.5.3 セキュリティ例外型での暗号スイートの詳細要求設定

6.1 節の条件を踏まえて、表 16 の通り、選定した暗号スイートをグループ A、グループ B、・・・とグループ分けをする。グループ分けの基準は安全性と実用性とのバランスの観点に立って行い、優先設定する順番としてグループ A から順に割り当てることを推奨する。なお、256 ビット安全性を優先することを妨げるものではなく、その場合には、グループ D、グループ A、グループ E、グループ B、グループ F、グループ C の順番に優先することを推奨する。

グループ A からグループ F までは推奨セキュリティ型と同様であるので、6.5.2 節を参照のこと。セキュリティ例外型では、推奨セキュリティ型に加え、グループ G とグループ H として、以下の暗号スイートグループを追加する。グループ内での暗号スイートから全部または一部を選択して設定するが、その際の優先順位は任意に定めてよい。

（RFC 必須）は、TLS1.2、TLS1.1 及び TLS1.0 を規定する RFC においてサポートが必須と指定されている暗号スイートであり、不特定多数からのアクセスを想定する SSL/TLS サーバにおいては利用可に設定すべき暗号スイートである。

また、「除外事項」は設定で除外すべき暗号スイートを示したものである。

表 16　セキュリティ例外型での暗号スイートの要求設定（基本）

グループ A～ グループ F	推奨セキュリティ型と同じ　（6.5.2 節参照）
グループ G	TLS_RSA_WITH_RC4_128_SHA (0x00,0x05)
グループ H	TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x00,0x16)
	TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x00,0x0A) （RFC 必須）
設定すべき鍵長	鍵交換で DHE を利用する場合には鍵長 1024 ビット以上 、RSA を利用する場合には鍵長 2048 ビット以上の設定を必須とする。なお、DHE の鍵長を明示的に設定できない製品を利用する場合には、DHE を含む暗号スイートは選定すべきではない
セキュリティ例外型での除外事項	グループ A～グループ G 及び表 15 以外のすべての暗号スイートを利用除外とする

7. SSL/TLS を安全に使うために考慮すべきこと

プロトコルとしての脆弱性だけでなく、実装上の脆弱性が発見されることも時おり起きる。

そのような脆弱性が発見されると基本的にはベンダからセキュリティパッチが提供されるので、ベンダが提供するセキュリティパッチを入手可能な状態とし、常にセキュリティパッチを適用して最新の状態にしておくことが望ましい。

それ以外にも、SSL/TLS をより安全に使うために、以下の項目を参考にするとよい。

7.1 サーバ証明書の作成・管理について注意すべきこと

7.1.1 サーバ証明書での脆弱な鍵ペアの使用の回避

OpenSSL などの暗号モジュールにおいて擬似乱数生成機能のエントロピー不足などの脆弱性が存在する場合、これを用いて鍵配送・共有や署名で使う公開鍵と秘密鍵の鍵ペアを生成した際に、結果的に解読容易な鍵ペアが生成されてしまうリスクがある。

こうしたリスクを防ぐためには、サーバ管理者は、鍵ペアの生成時点で脆弱性が指摘されていない暗号モジュールを利用するよう注意すべきである。また、既知の解読可能な鍵ペアでないことを確認するサービスなども提供されている[33]。

7.1.2 推奨されるサーバ証明書の種類

ブラウザなどをはじめとするサーバ証明書を検証するアプリケーションには、一定の基準に準拠した認証局の証明書（ルート CA 証明書）があらかじめ登録されており、これらの認証局（とその下位認証局）はパブリック認証局と呼ばれている。一般に、パブリック認証局が、第三者の立場から確認したサーバの運営組織等の情報を記載したサーバ証明書を発行し、ブラウザに予め搭載されたルート CA 証明書と組合せて検証を行うことで、サーバ証明書の信頼性を確保する。これにより、当該サーバ証明書の正当性が確認できれば、ブラウザは警告表示することなく、当該サーバへの接続を行う。

パブリック認証局から発行されるサーバ証明書は、その用途や利用範囲に応じて表 17 に示す 3 種類に分類される。これらのサーバ証明書のうち、不特定多数の利用者がアクセスする一般的な Web サーバ用途であれば、運営サイトの法的実在性の確認やグリーンバーによる視認性の高さといった優位点がある EV 証明書が利用者にとって一番安心できるサーバ証明書といえる。しかし、本ガイドライン公開時点（2018 年 5 月）においては、Let's Encrypt プロジェクト[34]が DV 証明書を無料配布するなど、EV 証明書と OV 証明書／DV 証明書との入手コストのギャップが拡大しており、またブラウザ以外のアプリケーションではそもそもグリーンバーを表示する場所がないなど、利用形態によっては必ずしも EV 証明書のメリットが十分に生かせないケースもある。

そこで、本ガイドラインでは、不特定多数の利用者がブラウザでアクセスする一般的な Web サ

[33] 例えば https://keytester.cryptosense.com/がある。ただし、安全性を 100%証明するものではないことに注意されたい

[34] https://letsencrypt.org/

ーバ用途について EV 証明書の利用を含めて検討すべきとし、特にドメイン名のなりすましリスクや運営組織の誤認リスクを避けたい場合（例：EC サイトや企業の公式 HP など）については EV 証明書の利用を推奨する。それ以外の利用ケースにおいては、入手コストと各々の証明書で実現される効用とのバランスを考慮して決めるべきである。

表 17　サーバ証明書の種類と違い

サーバ証明書の種類	内容の違い
DV 証明書 (Domain Validation)	サーバの運営組織が、サーバ証明書に記載されるドメインの利用権を有することを確認したうえで発行される証明書。 オンライン申請による短時間発行や低コストで入手できるものが多い、などのメリットがある。 一方、サーバの運営組織の実在性や、ドメイン名と運営組織の関係については確認されないので、自らのドメイン名と非常によく似たドメイン名の DV 証明書を、異なる運営組織が入手・利用可能であることを念頭に置いておく必要がある。場合によっては、不特定の利用者にサーバの運営組織をあえて誤認させる手段に利用される可能性もあることに留意されたい。
OV 証明書 (Organization Validation)	ドメイン名の利用権に加えて、サーバ運営組織の実在性の確認やドメイン名と運営組織との関係などについても確認した上で発行される証明書。 不特定多数の利用者がアクセスするような一般的な Web サーバの用途で利用されるが、①現状では利用者がブラウザで OV 証明書と DV 証明書を明確に識別することは難しい、②サーバ運営組織等の確認項目や確認方法は個々の認証局によって異なる、という課題もある。
EV 証明書 (Extended Validation)	OV 証明書と同様で、ドメイン名の利用権に加えて，サーバ運営組織の実在性等の確認やドメイン名と運営組織との関係などについても確認した上で発行される証明書。 3 つの証明書のなかでは発行コストが最もかかるが、以下の点で DV 証明書や OV 証明書に対して優位点を持つ。 ● 運営組織の法的実在性について、CA/ブラウザフォーラムが規定した国際的な認定基準にもとづいて確認が行われる。このため認証局に依らず一定レベルの確認が保証される ● ブラウザのアドレス表示部分等が緑色になる「グリーンバー」機能が有効に機能する場合には、利用者にとって EV 証明書であることの識別が容易 ● グリーンバーには運営組織も表示されるため，ドメイン名との関係が一目でわかる

7.1.3 サーバ証明書の有効期限

サーバ管理者は、サーバ証明書の更新漏れによって自社のサービスに障害を発生させることがないように、サーバ証明書の有効期間を管理し、更新作業のために必要なリードタイムを考慮した上で、適切な管理方法（例えば、更新作業開始時期の明文化など）を定めることが求められる。

市販されているサーバ証明書の有効期間は、半年程度のもの、1 年程度のもの、2 年程度のもの等様々である[35]。一般に、有効期間が長いほど、サーバ証明書の更新頻度が少なく更新作業の工数を削減できる。しかし、その反面、単純なミスによる更新忘れ、組織改編・担当者異動時の引き継ぎ不備による更新漏れ、鍵危殆化（秘密鍵の漏えい）リスクの増大、サーバ証明書に記載されたサーバの運営組織情報が（組織名変更などにより）正確でなくなるリスクの増大、アルゴリズム Agility（セキュリティ強度の変化に対して、安全な側に移行するための対策に要する時間、迅速さの程度）の低下などが危惧されるようになる。特に、2 年など比較的長い間有効なサーバ証明書を利用する場合には、管理者がサーバ証明書の有効期限切れに気づかず、更新漏れによるサービス障害の発生が大きなリスクとなりえる。

これらを総合的に勘案し、特段の制約が存在しない限り、サーバ管理者は、1 年程度の有効期間を持つサーバ証明書を選択し、サーバ証明書の更新作業を、年次の定型業務と位置付けることが望ましい。

なお、SHA-1 を利用しているサーバ証明書に関しては、速やかに SHA-256 を利用しているサーバ証明書への移行ができるようにするため、有効期間をできるだけ短く設定することが望ましい。

7.1.4 サーバ鍵の適切な管理

サーバ管理者は、サーバ証明書に対応する秘密鍵について、紛失、漏えい等が発生しないように適切に管理しなければならない。秘密鍵の紛失（データ破壊を含む）に備えバックアップを作成し保管する場合には、秘密鍵の危殆化[36]（漏えいなど）が発生しないようにするために、バックアップの方法や保管場所、その他の保管の要件について注意深く設計することが求められる。

サーバ管理者は、秘密鍵が危殆化した際に遅滞なく適切な対処を行うため、必要に応じて、次のような事項について、あらかじめ、方針及び手順を整理し文書化することが推奨される。

- 秘密鍵の危殆化に対応するための体制（関係者と役割、委託先との連携を含む）
- 秘密鍵が危殆化した、またはその恐れがあると判断するための基準
- 秘密鍵の危殆化の原因を調べること、及び、原因の解消を図ること
- 当該サーバ証明書の利用を停止すること（実施の判断基準、手順を含む）
- 当該サーバ証明書を遅滞なく失効すること（実施の判断基準、手順を含む）
- 新しい鍵ペアを生成し、新鍵に対して新しくサーバ証明書の発行を行うこと
- 秘密鍵の危殆化についての情報の開示（通知先、通知の方法、公表の方針等）

---

[35] CA/ブラウザフォーラムによる「Baseline Requirement」でサーバ証明書の有効期限についての要件が規定されている。2011 年 11 月以降に発行するサーバ証明書の有効期限は 60 ヶ月以内とされていたが、その後、2015 年 4 月以降の発行では 39 ヶ月以内、2018 年 3 月以降の発行では 825 日（約 27 ヶ月）以内と、徐々に有効期限が短くなってきている

[36] 安全性上の問題が生じ、信用できなくなる状態のこと

7.1.5 複数サーバに同一のサーバ証明書を利用する場合の注意

負荷分散や冗長化による可用性向上などを目的として複数のサーバに同一のサーバ証明書をインストールして利用する場合、サーバ管理者は、以下の観点で注意が必要である。

- サーバ証明書の更新や再発行の際には、入替作業の対象となる全てのサーバについて漏れなく証明書をインストールしなおすこと
- サーバ証明書の入替えに伴って暗号通信に関わる設定（4 章から 7 章までを参照）の変更を行う場合は、対象となる全てのサーバに漏れなく適用すること

サーバ管理者は、サーバ証明書の入替作業の対象となるサーバに漏れが発生しないよう、サーバ毎にペアとなる秘密鍵や暗号スイートなどの情報を一覧管理し、また外部からの監視／スキャンツールを用いたチェックと組合せるなど、管理方法を定め文書化することが推奨される。

7.1.6 ルート CA 証明書

サーバ証明書の安全性は、その証明書を発行する認証局自体の安全性はもとより、最終的には信頼の起点（トラストアンカー）となる最上位の認証局（ルート CA）の安全性に依拠している。

このことは、ルート CA の用いる暗号アルゴリズムおよび鍵長の安全性が十分でなければ、サーバ証明書の安全性も確保することができないことを意味している。例えば、ルート CA 証明書の安全性に問題が生じ、ブラウザベンダなどが当該ルート CA 証明書を失効させた場合、サーバ証明書自体には問題がなかったとしてもルート CA 証明書とともに失効することとなる。

このようなリスクを避けるためには、サーバ管理者は、信頼の起点（トラストアンカー）となるルート CA についても、当該サーバ証明書と同様の安全性を満たすルート CA 証明書を発行しているルート CA を選ぶべきである。ルート CA 証明書で利用している暗号アルゴリズムおよび鍵長の確認方法については、Appendix D.1 を参照されたい。

7.2 さらに安全性を高めるために

7.2.1 HTTP Strict Transport Security（HSTS）の設定有効化

例えばオンラインショッピングサイトのトップページが暗号化なしの HTTP サイトで、ショッピングを開始する際に HTTPS へリダイレクトされるような構成になっていた場合、リダイレクトを悪意のあるサイトに誘導し、情報を盗むといった中間者攻撃が SSL strip というツールを用いて可能であるという報告が Moxie Marlinspike によってなされた。

この攻撃に対して、HTTP で接続したら、すぐに強制的に HTTPS サイトへリダイレクトし、以降の通信は全て HTTPS とすることによって防御する技術が RFC 6797 で規定されている HTTP Strict Transport Security（HSTS）である。

HSTS に対応した SSL/TLS サーバに HTTPS でアクセスした場合、HTTPS 応答には以下のような HTTP ヘッダが含まれている。

Strict-Transport-Security:max-age=有効期間秒数;includeSubDomains

このヘッダを受け取った HSTS 対応のブラウザは、有効期間の間は当該サーバへは HTTP ではなく全て HTTPS で通信するように自動設定しておく。これにより、以前接続したときに HSTS が有効になっているサーバであれば、何らかの理由で、ブラウザが HTTP で接続しようとしても自動的に HTTPS に切り替えて接続する。

以上のように、HTTPS で安全にサービスを提供したい場合などでは、ユーザに意識させることなくミスを防止でき、ユーザの利便性を向上させることができるので、HSTS の機能を持っているならば有効にすることを推奨する。なお、HSTS は、主要なサーバ、クライアント（ブラウザ）ともに、2018 年 3 月時点の最新バージョンではすべてサポートされている。

7.2.2 リネゴシエーションの脆弱性への対策

リネゴシエーションとは、サーバとクライアントとの間で暗号アルゴリズムや暗号鍵の設定のために使われる事前通信（ハンドシェイク）において、一度確立したセッションに置き換わる新たなセッションを確立する際に、すでに確立したセッションを使って改めてハンドシェイクを行う機能である。

リネゴシエーションの脆弱性とは、クライアントとサーバの間に攻撃者が入る中間者攻撃によって、通信データの先頭部分に任意のデータを挿入することができるというものである（図 9）。これにより、例えば、攻撃者が挿入した HTTP リクエストを、あたかも正当なユーザから送られたリクエストであるかのようにサーバに誤認させるといったことができる。

この脆弱性のポイントは、リネゴシエーションが確立したセッションを使って行われることから、リネゴシエーションの前後の通信が同じ通信相手である、という前提で処理が行われる点にある。ところが、実際に図 9 の（10）で確立したセッションは、クライアントにとって一回目のハンドシェイクで確立したセッション（図 9 の（1）の要求に対するセッション）なのに対して、サーバはリネゴシエーションで確立したセッション（図 9 の（7）の要求に対するセッション）になっている。

それにも関わらず、両者がその食い違いを認識できないため、その結果として、サーバは、リネゴシエーション前の攻撃者からの通信（図 9 の（5）の通信）とリネゴシエーション後のクライアントからの通信（図 9 の（11）（12）の通信）を、同一クライアントからの通信と誤認して受け付けて処理を行うことになり、予期せぬ事態を引き起こす可能性がある。

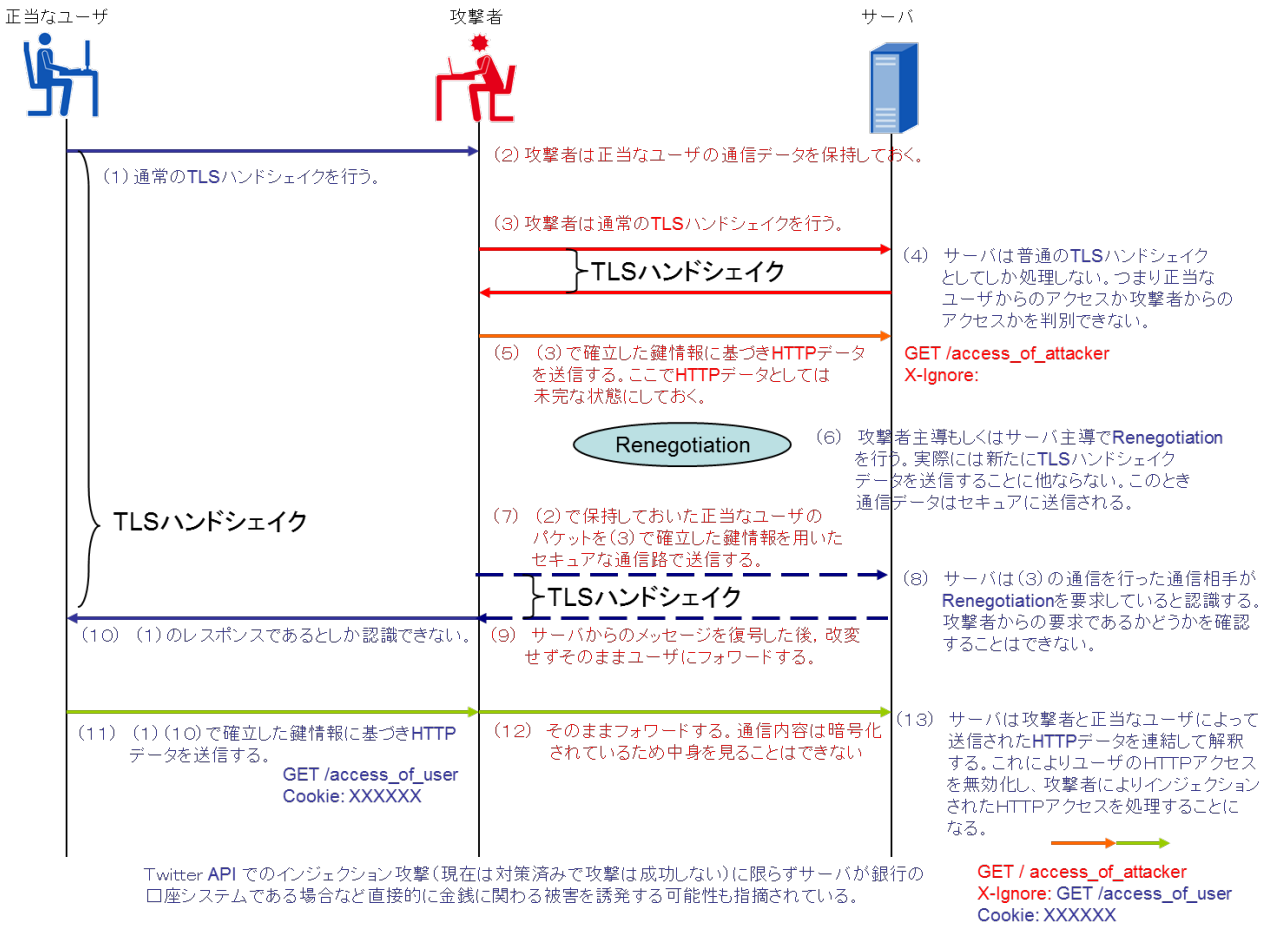


図 9　リネゴシエーションの脆弱性

【推奨対策】

リネゴシエーションに関するプロトコル上の脆弱性であることから、対策としては以下のどちらかの設定とすることを推奨する。

- リネゴシエーションを利用不可とする
- リネゴシエーションの脆弱性対策（RFC5746）を反映したバージョンの製品を利用するとともに、対策が取られていないバージョンの製品からのリネゴシエーション要求は拒否する設定を行う

7.2.3　圧縮機能を利用した実装攻撃への対策

圧縮機能は、何度も出てくる同じ長い文字列を別の短い情報に置き換えることで全体のデータサイズを削減し、通信効率を向上させるために利用するものである。

しかしながら、圧縮対象となる文字列に秘密情報が含まれている場合、圧縮機能によって別の情報に置き換わることによるデータサイズの変動に着目することによって、どの文字列が圧縮されたのかが分かる可能性がある。しかも、着目しているのはデータサイズであるので、データが暗号化されているかどうかは関係がない。

実際にこのような圧縮機能を利用した実装攻撃として、CRIME、TIME、BREACH などがある。これらの攻撃は、SSL/TLS のプロトコル自体の脆弱性ではなく、圧縮機能の特性そのものを利用

した攻撃方法である。したがって、根本的な対策としては「SSL/TLS では圧縮機能を利用しない」こと以外に方法はない。

このため、最近のバージョンの OpenSSL や Windows などでは、デフォルト設定がオフになっていたり、そもそもサポートを取りやめたりしている。

7.2.4 OCSP Stapling の設定有効化

サービス提供の終了やサーバの秘密鍵の漏えいなど、何らかの理由で、サーバ証明書の有効期間内であっても当該サーバ証明書が失効している場合がある。そのため、サーバ証明書の正当性を確認する時には、当該サーバ証明書が失効していないかどうかもあわせて確認すべきである。

サーバ証明書が失効されていないか確認する方法として、CRL[37]と OCSP[38]の二つの方法があるが、CRL はサイト数の増大に伴ってファイルサイズが増大しており、近年では OCSP のみに依存するブラウザが多くを占めている。

ただ、OCSP を使用した場合には 2 つの問題がある。

1) OCSP 実行時の通信エラー処理について明確な規定がなく、ブラウザの実装に依存する。このため、OCSP レスポンダの通信障害等で適切な OCSP 応答が得られない場合にサーバ証明書の失効検証を正しく行わないまま SSL 通信を許可してしまうブラウザも少なくない。そのようなブラウザに対しては、あるサイトのサーバ証明書が失効していたとしても、DDoS 攻撃などにより意図的に OCSP レスポンダに接続させないことにより、当該サイトが有効であるとして SSL/TLS 通信をさせることができる
2) OCSP を使った場合には、あるサイトにアクセスがあったことを OCSP レスポンダも知り得てしまうため、プライバシー上の懸念がある。例えば、ある利用者が、ある会員制のサイトにアクセスした場合、ブラウザはサーバ証明書の失効検証のために当該サイトの OCSP 応答を取得する。そこで、OCSP レスポンダのアクセス履歴から、ある接続元 IP の利用者は、当該サイトの会員であると OCSP レスポンダが知り得ることになる

上記の問題を解決するために、RFC 6066 Transport Layer Security (TLS) Extension: Extension Definition の 8 節で、Certificate Status Request という TLS 拡張が規定されている。これを使うことにより、OCSP 応答を OCSP レスポンダからではなく、アクセス先サイトの Web サーバから取得して SSL/TLS 通信を開始することができる。

- OCSP レスポンダからの OCSP 応答を Web サーバがキャッシュしている限り、ブラウザは OCSP 応答による失効検証を行うことができる
- OCSP 応答を、OCSP レスポンダからではなく、Web サーバから取得するので、当該サイトへのアクセス履歴を OCSP レスポンダが知ることはない

[37] Certificate Revocation List
[38] Online Certificate Status Protocol

なお、OCSP Stapling は主要なサーバ、クライアント（ブラウザ）ともに、2018 年 3 月時点の最新バージョンではすべてサポートされている。

7.2.5 Public Key Pinning の設定有効化

近年、FLAME 攻撃や、DigiNotar、TURKTRUST などの認証局からのサーバ証明書の不正発行など、偽のサーバ証明書を使った攻撃手法が増加傾向にある。これらの攻撃により発行されたサーバ証明書は、認証局が意図して発行したものではないという意味で“偽物”であるが、動作そのものは“本物”と同じふるまいをする。

このため、この種の攻撃に対しては、従来の PKI による、信頼するルート証明書のリストと、証明書チェーンの検証（認証パス検証）だけでは正当なサーバ証明書であるかどうかの判断がつかない。

これを補う目的で導入されつつあるのが、Public Key Pinning（もしくは Certificate Pinning）と呼ばれている技術である。従来の PKI による証明書チェーンの検証に加え、Public Key Pinning では、あるサイト用に期待されるサーバ証明書の公開鍵情報（SPKI; Subject Public Key Info）フィールドの情報のハッシュ値を比較することにより、当該サーバ証明書が正当なものであるかどうかを判断する。

ただし､現状では､多くのブラウザがサポートを取りやめているか取りやめる計画をしており、主要ブラウザでは Mozilla Firefox がサポートしているだけである。

【コラム③】完全 HTTPS 化の落とし穴

USENIX Security 2017 で発表された「Measuring HTTPS Adoption on the Web」の論文[39]を契機に、完全 HTTPS 化（HTTPS-only、常時 SSL 化(AOSSL; Always on SSL)といわれることもある）の流れが世界的に広がっている。日本でも、jp ドメインサイトの HTTPS 化率が欧米に比べてかなり低いと指摘されたことで一時期話題になった。

完全 HTTPS 化とは、今まで HTTP で通信していた Web サーバに対しても SSL/TLS での通信を行うように設定することでセキュリティを向上させることを意図しており、特に Google と Mozilla などが先導している。また、完全 HTTPS 化をする上ではサーバ証明書が必要となるが、Let's Encrypt プロジェクト[40]のように、無償で SSL/TLS サーバ証明書を発行するサービスも登場している。

政府関連では、米国政府の全 Web サーバの完全 HTTPS 化の指示[41]や、日本政府の情報セキュリティ対策のための統一基準群の見直しの中で完全 HTTPS 化の計画[42]が公表されている。

ところで、パスワードや個人情報等、データ保護が必要な Web サーバで SSL/TLS を使うのは当然として、そのような情報を扱わない Web サーバまでもが何故 SSL/TLS を使う必要があるというのだろうか。

その答えは、「通信の暗号化」と並んで、SSL/TLS が持つもう一つの重要な機能である「Web サーバの認証」を行うことにある。これによって、ブラウザが接続しようとしている Web サーバが意図した先の Web サーバであることを確認し、悪意ある第三者がなりすました Web サーバ（例えばフィッシングサイト）へ誘導されることを防止することを意図している。

もっとも、HTTP 用に作られている Web サーバを単に SSL/TLS を使う設定にすれば完全 HTTPS 化が実現しセキュリティが向上する、というほど簡単なものではないことを認識しておく必要がある。

ここでは、4 点ほど課題を指摘しておく。

1)～3)のいずれかの課題に当てはまるような場合には、Web サーバの作りそのものを再検討し必要な対応をした後でないと、完全 HTTP 化をしても期待する効果が得られなかったり、最悪の場合は逆効果になりかねないことがあるので注意されたい。実際、この種の設定誤りが多く発生しているとの報告[43]もある。

また、4)については自らが完全 HTTPS 化をするかどうかに関わらず、完全 HTTPS 化の流れが進むことによってより顕在化するリスクである。実際、完全 HTTPS 化を率先して対応したのがフィッシングサイトだったとする報告[44]があるなど、完全 HTTPS 化に対する認識を逆手に取った攻撃が行われていることに留意する必要がある。

[39] https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/felt
[40] https://letsencrypt.org/
[41] https://https.cio.gov/
[42] https://www.nisc.go.jp/conference/cs/dai17/pdf/17shiryou03.pdf
[43] 奥田、秋山、早川、"Web サイト全体の HTTPS 対応とユーザビリティ及び運用上の課題," SCIS2018
[44] https://news.netcraft.com/archives/2017/05/17/phishing-sites-react-promptly-to-new-browser-changes.html

1) Web サーバの HTTPS のコンテンツの中に HTTP のコンテンツが混在している作りをしている

ブラウザとコンテンツの組合せによって、警告・注意喚起表示（コンテンツの一部がブロックされる）、HTTPS 非対応表示（南京錠が表示されない）、HTTPS 表示（南京錠が表示される）と全く異なる挙動になる。

2) 一部が HTTPS になっている Web サーバでのサーバ証明書をそのまま完全 HTTPS 化でのサーバ証明書に転用する

サーバ証明書に記載されているドメイン名が異なっている場合、サーバ証明書の検証エラーの原因になる。

3) クラウドサービスなどで Web サーバを開設している

どこが SSL/TLS の終端になるかを確認することが必要である。もし、SSL/TLS の終端がクラウドサービス事業者のサーバ（例えば CDN サーバ）の場合、サーバ証明書に含まれている FQDN（Fully Qualified Domain Name）設定が正しくないとサービス事業者のサーバを「正しいサーバ証明書を持たないアクセス先の Web サーバ」とみなして、警告画面が表示される。これは、アクセス先のサーバ証明書に含まれている FQDN が、SSL/TLS の終端であるサービス事業者の CDN サーバが実際に管理するドメイン名と異なることに起因して発生する事象である。

4) 似た URL が第三者に使われるリスクが無視できない／第三者に使われると悪影響が大きい

例えば「ABC-inc.co.jp」が正規の URL の場合に、第三者に「ABC-corp.co.jp」「ABC-inc.com」「ABCinc.co.jp」等といった非常によく似た URL を使われるといったケースである。完全 HTTPS 化以前からの問題ではあるが、完全 HTTP 化によって SSL/TLS によるサーバ認証が行われることで「保護された接続」「安全な接続」等と表示されるようになるため、第三者の URL を正しい URL と誤認する可能性がむしろ高くなる恐れがある。これに対抗するには、視認的に区別可能な EV 証明書を使うなどの対策を採ることが重要となる。

PART II：

ブラウザ＆リモートアクセスの利用について

8. ブラウザを利用する際に注意すべきポイント

8.1 本ガイドラインが対象とするブラウザ

8.1.1 対象とするプラットフォーム

ベンダがセキュリティホールに対する修正を行っている OS を利用すべきである。本ガイドラインの公開時点（2018 年 5 月）で、サポート対象となっているものは以下の通りである。

- デスクトップ向け OS
  - Windows 7 Service Pack 1 （2020 年 4 月 11 日サポート終了）
  - Windows 8.1 （2023 年 1 月 10 日サポート終了）
  - Windows 10 Home/Pro/Pro for Workstation バージョン 1709（提供日 2017 年 10 月 17 日、2019 年 4 月 9 日サポート終了）
  - Windows 10 Home/Pro/Pro for Workstation バージョン 1703（提供日 2017 年 4 月 5 日、2018 年 10 月 9 日サポート終了）
  - Windows 10 Enterprise/Education バージョン 1709（提供日 2017 年 10 月 17 日、2019 年 10 月 9 日サポート終了）
  - Windows 10 Enterprise/Education バージョン 1703（提供日 2017 年 4 月 5 日、2019 年 4 月 9 日サポート終了）
  - Windows 10 Enterprise/Education バージョン 1607（提供日 2016 年 8 月 2 日、2018 年 10 月 10 日サポート終了）
  - Windows 10 Enterprise 2015 LTSB（提供日 2015 年 7 月 29 日、2025 年 10 月 14 日サポート終了）
  - Windows 10 Enterprise 2016 LTSB（提供日 2016 年 8 月 2 日、2026 年 10 月 13 日サポート終了）
  - OS X El Capitan (10.11)（2018 年 3 月 29 日アップデート）
  - macOS Sierra (10.12)（2018 年 3 月 29 日アップデート）
  - macOS High Sierra (10.13)（2018 年 3 月 29 日アップデート）

- スマートフォン向け OS
  - 当該端末で利用できる最新の Android（2018 年 3 月時点で最新バージョンは Android 8.x）
  - 当該端末で利用できる最新の iOS（2018 年 3 月時点で最新バージョンは iOS 11.x）

8.1.2 対象とするブラウザのバージョン

ブラウザは、少なくとも提供ベンダがサポートしているバージョンのものを利用すべきである。本ガイドラインの公開時点（2018 年 5 月）でサポートしている、8.1.1 節に挙げた OS 上で動作するブラウザのバージョンは以下のとおりである。

- Microsoft Internet Explorer 11
- Microsoft Edge
- Apple Safari 最新版
- Google Chrome 最新版
- Mozilla Firefox 最新版
- Mobile Safari（iOS）

8.2 設定に関する確認項目

8.2.1 基本原則

8.1 節で対象とするブラウザは、インストール時のデフォルト設定で利用することを各ベンダは推奨しているので、企業の情報システム担当からの特別な指示がある場合などを除き、原則としてデフォルト設定を変えずに利用することを強く推奨する。

【基本原則】

- ベンダがサポートしているブラウザであって、更新プログラムを必ず適用し、最新状態にして利用する
- 自動更新を有効化しておく
- 企業の情報システム担当からの特別な指示がある場合などに限り、社内ポリシーに従う

8.2.2 設定項目

設定項目を標準機能で提供していないブラウザ

以下のブラウザは、設定変更オプションが提供されておらず、そもそも設定変更ができない。

- PC 版 Web ブラウザ
  - Apple Safari
  - Google Chrome

- スマートフォンに含まれる Web ブラウザ
  - Google Chrome
  - Mobile Safari (iOS)

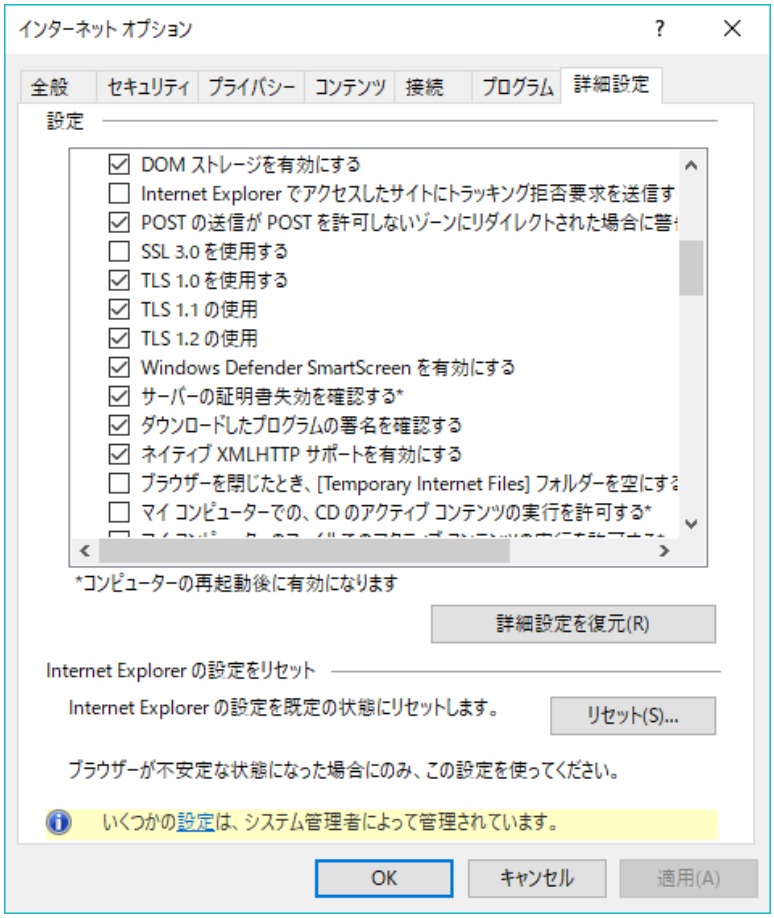
設定項目を標準機能で提供しているブラウザ

以下のブラウザは、設定変更オプションが提供されている。ただし、特別な指示がない限り、デフォルト設定を変更すべきではない。

- Microsoft Internet Explorer／Microsoft Edge
  他のブラウザとは異なり、Internet Explorer と Microsoft Edge では、
  “ツール”→“インターネットオプション”→“詳細設定”

を選択すると多数の設定項目が表示され、ユーザが細かく設定できるようになってはいる。しかし、安全性を考慮してデフォルト設定が行われていることから、特段の理由がない場合に設定を変更することは推奨しない。

【プロトコルバージョンの設定】
“ツール”→“インターネットオプション”→“詳細設定”を選択した後、設定項目を“セキュリティ”までスクロールさせると、「SSL3.0 を使用する」「TLS1.0 を使用する」「TLS1.1 を使用」「TLS1.2 を使用」等といったチェックボックスが表示される。ここでのチェックボックスにチェックが入っているプロトコルバージョンが、ブラウザが使うことができるプロトコルバージョンとなる。以下は、Windows10 Internet Explorer 11 の設定画面である。



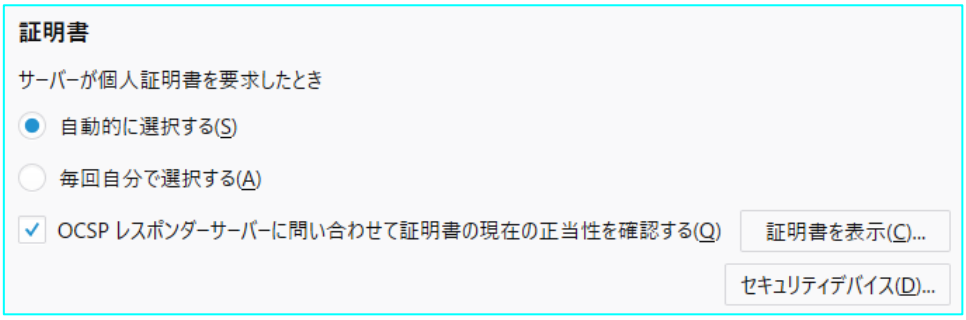
- Firefox

Firefox では、サーバ証明書の検証、失効機能においてどのように処理するかの動作についてのみ設定方法を提供している。この設定については、

“メニュー”→“オプション”→“プライバシーとセキュリティ”→“証明書”

を選択することで設定方法へのダイアログが表示される。

デフォルトの設定は以下のようになっており、特段の理由がない場合に変更することは推奨しない。



8.3 ブラウザ利用時の注意点

8.3.1 SHA-1 を利用するサーバ証明書の警告表示

CA/ブラウザフォーラムでは、2016 年 1 月 1 日以降、パブリック認証局は SHA-1 で署名されたサーバ証明書を発行しないことが決められている。このため、ブラウザベンダ各社では、SHA-1 で署名されたサーバ証明書を無効化する対処をしている。

詳しくは以下のとおりである。

- Microsoft Internet Explorer／Microsoft Edge
  2017 年 5 月 9 日に公開した更新版で、Internet Explorer 11、Edge では、SHA-1 で署名されたサーバ証明書の無効化をしている[45]

- Google Chrome
  Chrome56 から SHA-1 で署名されたサーバ証明書の無効化をしている[46]

- Firefox
  Firefox36 から SHA-1 で署名されたサーバ証明書の無効化をしている[47]

[45] https://technet.microsoft.com/ja-jp/library/security/4010323
[46] https://security.googleblog.com/2016/11/sha-1-certificates-in-chrome.html
[47] https://www.fxsitecompat.com/en-CA/docs/2016/sha-1-certificates-issued-by-public-ca-will-no-longer-be-accepted/

9. その他のトピック

9.1 リモートアクセス VPN over SSL （いわゆる SSL-VPN）

SSL-VPN と呼ばれるものは、正確には SSL を使った“リモートアクセス VPN”の実現方法といえる。SSL-VPN 装置を介して SSL-VPN 装置の奥にあるサーバ（インターネットからは直接アクセスできないサーバ）とクライアント端末をつなぐ形での VPN であり、IPsec-VPN のような特定端末間だけで VPN を構成する、いわゆる拠点間 VPN とは異なる。

したがって、あくまでリモートアクセスでの通信経路上が SSL/TLS で保護されているにすぎないと考え、本ガイドラインの推奨セキュリティ型（または高セキュリティ型）の設定を適用することとし、Appendix A.3（または Appendix A.2）のチェックリストを用いて確認すべきである。

なお、一口に SSL-VPN といっても、実現形態が製品によって全く異なることに注意がいる。実現形態としては、大きく以下の 3 通りに分かれる。

- 通常のブラウザを利用する“クライアントレス型”
- 接続時に自動的に Java や Active X をインストールすることでブラウザだけでなく、アプリケーションでも利用できるようにした“on-demand インストール型”
- 専用のクライアントソフト（通信アダプタなどを含む）をインストール・設定してから利用する“クライアント型”がある。

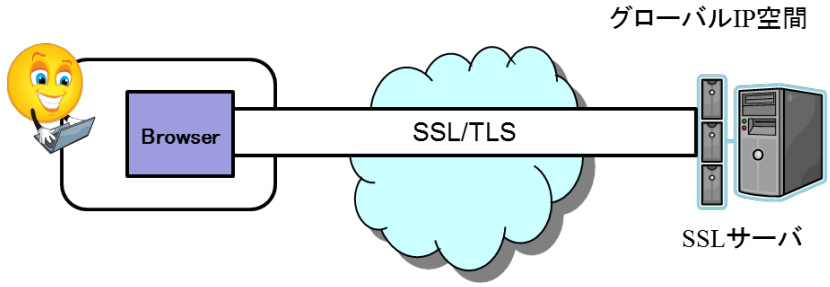
クライアントレス型は、ブラウザさえあればどの端末からでもアクセス可能であり、利便性に優れる一方、SSL との最大の差はグローバル IP をインターネットに公開しているか否か程度の違いといえる。結果として、最初のクライアント認証を SSL/TLS サーバが受け持つか、SSL-VPN 装置が受け持つか程度の差でしかなく、VPN というよりも、本質的には SSL/TLS と同じものとみるべきである。

On-demand インストール型も、接続時に自動的にインストールされることから、特に利用端末に制限を加えるものではなく、クライアントレス型と大きく異なるわけではない。むしろ、ブラウザでしか使えなかったクライアントレス型を、他のアプリケーションでも利用できるように拡張したという位置づけのものである。

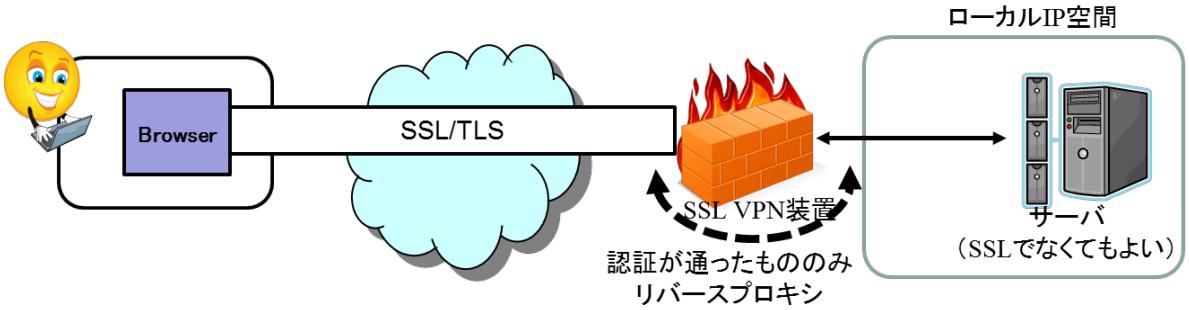
一方、クライアント型は上記の 2 つのタイプとは明らかに異なり、専用のクライアントソフトがインストールされた端末との間でのみアクセスする。つまり、誤って偽サーバに接続することがなく、また内部サーバにアクセスできる端末も厳格に制限できるため、端末に IPsec-VPN ソフトをインストールして構成するモバイル型の IPsec-VPN に近い形での運用形態となる。

機密度の高い情報を扱うのだとすれば、少なくともクライアント型での SSL-VPN を利用すべきである。

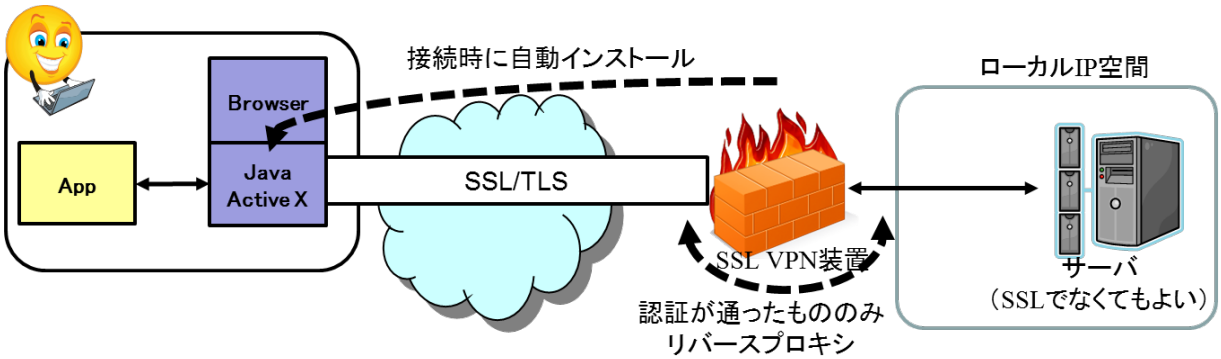
【参考：通常の SSL/TLS】



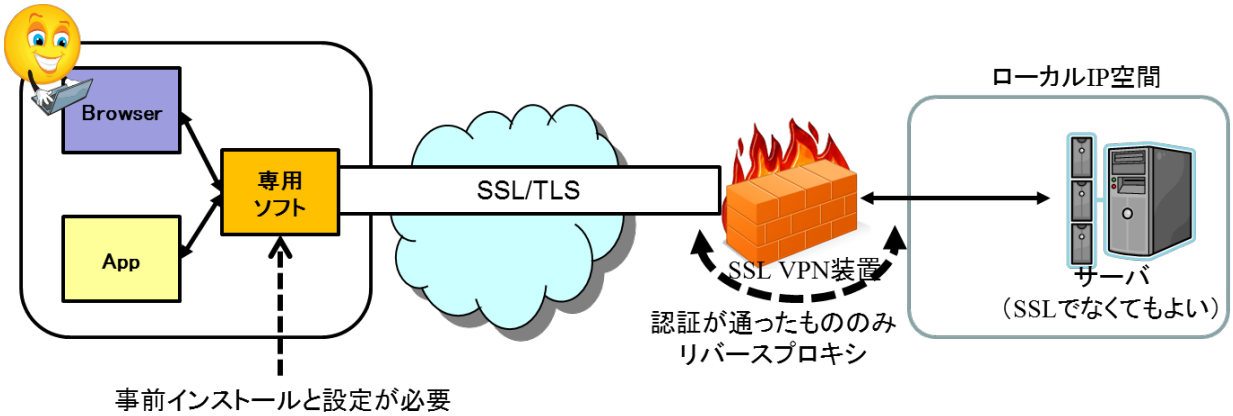
【クライアントレス型（ブラウザベース）】



【On-demand インストール型（Java や Active X を使ってブラウザ以外でも利用可能）】



【クライアント型（専用ソフトベース）】



Appendix：

付録

Appendix A：チェックリスト

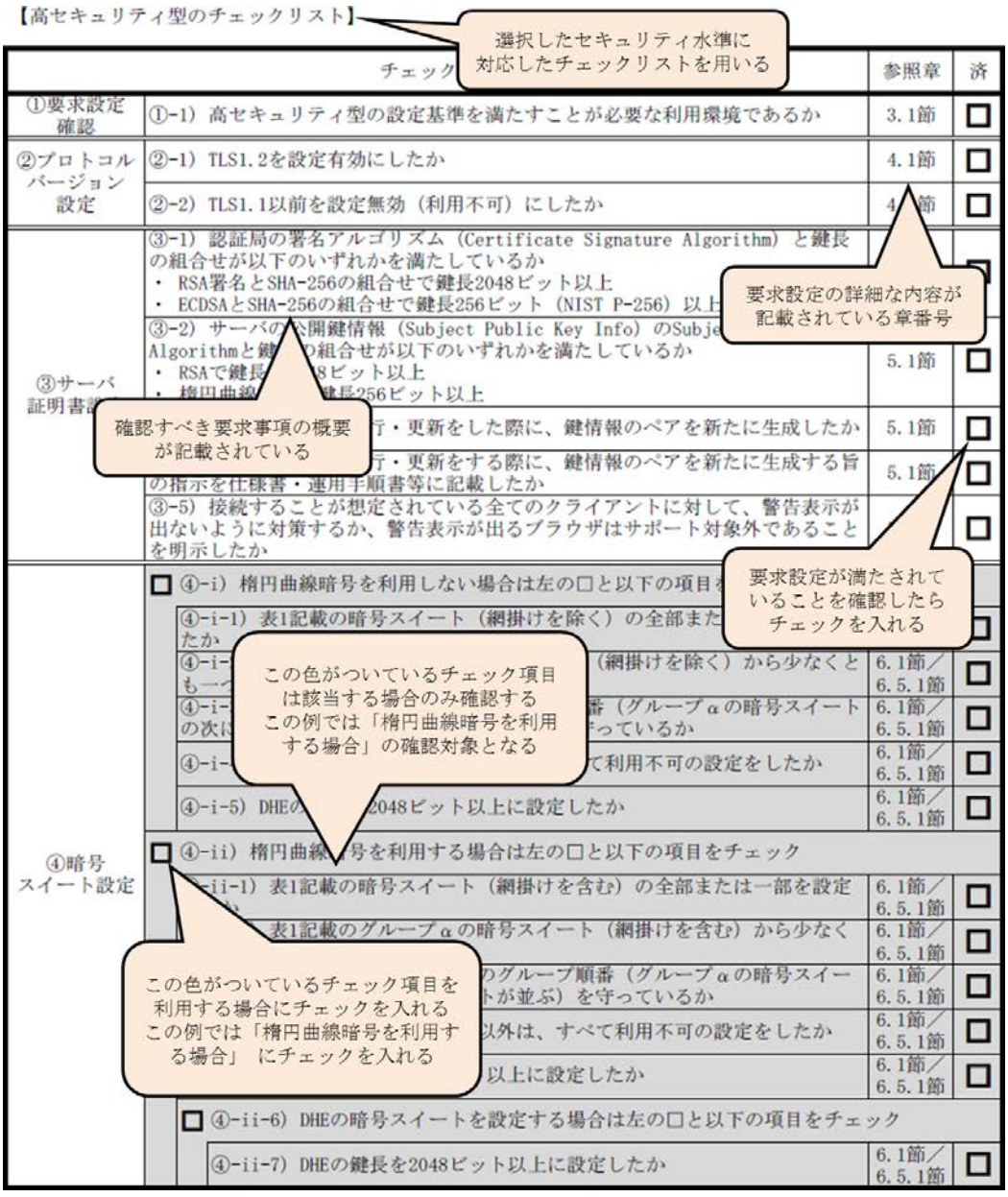
チェックリストの原本は以下の URL からも入手可能である。

[PDF 版] https://www.ipa.go.jp/security/ipg/documents/ipa-cryptrec-gl-3001-2.0-checklists.pdf

[Excel 版] https://www.ipa.go.jp/security/ipg/documents/ipa-cryptrec-gl-3001-2.0-checklists.xlsx

A.1. チェックリストの利用方法

本チェックリストは、記載のチェック項目について、選択した設定基準に対応した要求設定をもれなく実施したことを確認するためのチェックリストである。選択した設定基準に応じたチェックリストを用い、すべてのチェック項目について、該当章に記載の要求設定に合致していることを確認して「済」にチェックが入ることが求められる。



A.2.　高セキュリティ型のチェックリスト

【高セキュリティ型のチェックリスト】

チェック項目		参照章	済
①要求設定確認	①-1）高セキュリティ型の設定基準を満たすことが必要な利用環境であるか	3.1節	□
②プロトコルバージョン設定	②-1）TLS1.2を設定有効にしたか	4.1節	□
	②-2）TLS1.1以前を設定無効（利用不可）にしたか	4.1節	□
③サーバ証明書設定	③-1）認証局の署名アルゴリズム（Certificate Signature Algorithm）と鍵長の組合せが以下のいずれかを満たしているか ・ RSA署名とSHA-256の組合せで鍵長2048ビット以上 ・ ECDSAとSHA-256の組合せで鍵長256ビット（NIST P-256）以上	5.1節	□
	③-2）サーバの公開鍵情報（Subject Public Key Info）のSubject Public Key Algorithmと鍵長の組合せが以下のいずれかを満たしているか ・ RSAで鍵長は2048ビット以上 ・ 楕円曲線暗号で鍵長256ビット以上	5.1節	□
	③-3）サーバ証明書の発行・更新をした際に、鍵情報のペアを新たに生成したか	5.1節	□
	③-4）サーバ証明書の発行・更新をする際に、鍵情報のペアを新たに生成する旨の指示を仕様書・運用手順書等に記載したか	5.1節	□
	③-5）接続することが想定されている全てのクライアントに対して、警告表示が出ないように対策するか、警告表示が出るブラウザはサポート対象外であることを明示したか	5.1節	□
④暗号スイート設定	□ ④-i）楕円曲線暗号を利用しない場合は左の□と以下の項目をチェック		
	④-i-1）表1記載の暗号スイート（網掛けを除く）の全部または一部を設定したか	6.1節／6.5.1節	□
	④-i-2）表1記載のグループαの暗号スイート（網掛けを除く）から少なくとも一つは設定したか	6.1節／6.5.1節	□
	④-i-3）表1記載の暗号スイートのグループ順番（グループαの暗号スイートの次にグループβの暗号スイートが並ぶ）を守っているか	6.1節／6.5.1節	□
	④-i-4）表1記載の暗号スイート以外は、すべて利用不可の設定をしたか	6.1節／6.5.1節	□
	④-i-5）DHEの鍵長を2048ビット以上に設定したか	6.1節／6.5.1節	□
	□ ④-ii）楕円曲線暗号を利用する場合は左の□と以下の項目をチェック		
	④-ii-1）表1記載の暗号スイート（網掛けを含む）の全部または一部を設定したか	6.1節／6.5.1節	□
	④-ii-2）表1記載のグループαの暗号スイート（網掛けを含む）から少なくとも一つは設定したか	6.1節／6.5.1節	□
	④-ii-3）表1記載の暗号スイートのグループ順番（グループαの暗号スイートの次にグループβの暗号スイートが並ぶ）を守っているか	6.1節／6.5.1節	□
	④-ii-4）表1記載の暗号スイート以外は、すべて利用不可の設定をしたか	6.1節／6.5.1節	□
	④-ii-5）ECDHEの鍵長を256ビット以上に設定したか	6.1節／6.5.1節	□
	□ ④-ii-6）DHEの暗号スイートを設定する場合は左の□と以下の項目をチェック		
	④-ii-7）DHEの鍵長を2048ビット以上に設定したか	6.1節／6.5.1節	□

【表1】

優先順位グループ	暗号スイート名	スイート番号
グループα	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	(0x00,0x9F)
	TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x7D)
	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	(0xC0,0x2C)
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	(0xC0,0x30)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x87)
	TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x8B)
グループβ	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	(0x00,0x9E)
	TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x7C)
	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	(0xC0,0x2B)
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	(0xC0,0x2F)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x86)
	TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x8A)

A.3. 推奨セキュリティ型のチェックリスト

【推奨セキュリティ型のチェックリスト（1/2）】

チェック項目		参照章	済
①要求設定確認	チェック項目なし		
②プロトコルバージョン設定	②-1）TLS1.0を設定有効にしたか	4.1節	□
	②-2）SSL2.0及びSSL3.0を設定無効（利用不可）にしたか	4.1節	□
	□ ②-3）TLS1.2が実装されている場合には左の□と以下の項目をチェック		
	②-4）TLS1.2について設定を有効にしたか	4.1節	□
	□ ②-5）TLS1.1が実装されている場合には左の□と以下の項目をチェック		
	②-6）TLS1.1について設定を有効にしたか	4.1節	□
	□ ②-7）プロトコルバージョン優先順位を設定できる場合には左の□と以下の項目をチェック		
	②-8）設定有効になっているプロトコルバージョンのうち、もっとも新しいバージョンによる接続を最優先にしたか。接続できない場合に、順番に一つずつ前のプロトコルバージョンでの接続するようにしたか	4.1節	□
③サーバ証明書設定	③-1）認証局の署名アルゴリズム（Certificate Signature Algorithm）と鍵長の組合せが以下のいずれかを満たしているか ・ RSA署名とSHA-256の組合せで鍵長2048ビット以上 ・ ECDSAとSHA-256の組合せで鍵長256ビット（NIST P-256）以上	5.1節	□
	③-2）サーバの公開鍵情報（Subject Public Key Info）のSubject Public Key Algorithmと鍵長の組合せが以下のいずれかを満たしているか ・ RSAで鍵長は2048ビット以上 ・ 楕円曲線暗号で鍵長256ビット以上	5.1節	□
	③-3）サーバ証明書の発行・更新をした際に、鍵情報のペアを新たに生成したか	5.1節	□
	③-4）サーバ証明書の発行・更新をする際に、鍵情報のペアを新たに生成する旨の指示を仕様書・運用手順書等に記載したか	5.1節	□
	③-5）接続することが想定されている全てのクライアントに対して、警告表示が出ないように対策するか、警告表示が出るブラウザはサポート対象外であることを明示したか	5.1節	□

（続く）

【推奨セキュリティ型のチェックリスト（2/2）】

	チェック項目	参照章	済
④暗号スイート設定	□ ④-i）楕円曲線暗号を利用しない場合は左の□と以下の項目をチェック		
	④-i-1）表2記載の暗号スイート（網掛けを除く）の全部または一部を設定したか	6.1節／6.5.2節	□
	④-i-2）表2記載のグループA及びグループBの暗号スイート（網掛けを除く）から少なくとも一つは設定したか	6.1節／6.5.2節	□
	④-i-3）表2記載の暗号スイートのグループ順番の制限[注]を守っているか	6.1節／6.5.2節	□
	④-i-4）表2記載の暗号スイート以外は、すべて利用不可の設定をしたか	6.1節／6.5.2節	□
	④-i-5）RSAの鍵長を2048ビット以上に設定したか	6.1節／6.5.2節	□
	□ ④-i-6）DHEを利用する暗号スイートを設定する場合は左の□と以下の項目をチェック		
	④-i-7）DHEの鍵長を1024ビット以上に設定したか	6.1節／6.5.2節	□
	□ ④-i-8）不特定多数に公開されるサービス等において使用されるサーバである場合には左の□と以下の項目をチェック		
	④-i-9）AES128-SHAの暗号スイートを設定したか	6.1節／6.5.2節	□
	□ ④-ii）楕円曲線暗号を利用する場合は左の□と以下の項目をチェック		
	④-ii-1）表2記載の暗号スイート（網掛けを含む）の全部または一部を設定したか	6.1節／6.5.2節	□
	④-ii-2）表2記載のグループA及びグループBの暗号スイート（網掛けを含む）から少なくとも一つは設定したか	6.1節／6.5.2節	□
	④-ii-3）表2記載の暗号スイートのグループ順番の制限[注]を守っているか	6.1節／6.5.2節	□
	④-ii-4）表2記載の暗号スイート以外は、すべて利用不可の設定をしたか	6.1節／6.5.2節	□
	④-ii-5）ECDHE/ECDHの鍵長を256ビット以上に設定したか	6.1節／6.5.2節	□
	④-ii-6）RSAの鍵長を2048ビット以上に設定したか	6.1節／6.5.2節	□
	□ ④-ii-7）DHEを利用する暗号スイートを設定する場合は左の□と以下の項目をチェック		
	④-ii-8）DHEの鍵長を1024ビット以上に設定したか	6.1節／6.5.2節	□
	□ ④-ii-9）不特定多数に公開されるサービス等において使用されるサーバである場合には左の□と以下の項目をチェック		
	④-ii-10）AES128-SHAの暗号スイートを設定したか	6.1節／6.5.2節	□

［注］　許容される暗号スイートのグループ順番は以下のとおり
（128ビット安全性優先の場合）
・グループA→グループB→グループC→グループD→グループE→グループF
（256ビット安全性優先の場合）
・グループD→グループA→グループE→グループB→グループF→グループC

【表2】

優先順位グループ	暗号スイート名	スイート番号
グループA	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	(0x00,0x9E)
	TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x7C)
	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	(0x00,0x67)
	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256	(0x00,0xBE)
	TLS_DHE_RSA_WITH_AES_128_CBC_SHA	(0x00,0x33)
	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA	(0x00,0x45)
	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	(0xC0,0x2B)
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	(0xC0,0x2F)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x86)
	TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x8A)
	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	(0xC0,0x23)
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	(0xC0,0x27)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256	(0xC0,0x72)
	TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256	(0xC0,0x76)
	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	(0xC0,0x09)
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	(0xC0,0x13)
グループB	TLS_RSA_WITH_AES_128_GCM_SHA256	(0x00,0x9C)
	TLS_RSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x7A)
	TLS_RSA_WITH_AES_128_CBC_SHA256	(0x00,0x3C)
	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256	(0x00,0xBA)
	TLS_RSA_WITH_AES_128_CBC_SHA	(0x00,0x2F)
	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA	(0x00,0x41)
グループC	TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256	(0xC0,0x2D)
	TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256	(0xC0,0x31)
	TLS_ECDH_ECDSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x88)
	TLS_ECDH_RSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x8C)
	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	(0xC0,0x25)
	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	(0xC0,0x29)
	TLS_ECDH_ECDSA_WITH_CAMELLIA_128_CBC_SHA256	(0xC0,0x74)
	TLS_ECDH_RSA_WITH_CAMELLIA_128_CBC_SHA256	(0xC0,0x78)
	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	(0xC0,0x04)
	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	(0xC0,0x0E)
グループD	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	(0x00,0x9F)
	TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x7D)
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	(0x00,0x6B)
	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256	(0x00,0xC4)
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA	(0x00,0x39)
	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA	(0x00,0x88)
	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	(0xC0,0x2C)
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	(0xC0,0x30)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x87)
	TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x8B)
	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	(0xC0,0x24)
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	(0xC0,0x28)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_CBC_SHA384	(0xC0,0x73)
	TLS_ECDHE_RSA_WITH_CAMELLIA_256_CBC_SHA384	(0xC0,0x77)
	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	(0xC0,0x0A)
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	(0xC0,0x14)
グループE	TLS_RSA_WITH_AES_256_GCM_SHA384	(0x00,0x9D)
	TLS_RSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x7B)
	TLS_RSA_WITH_AES_256_CBC_SHA256	(0x00,0x3D)
	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256	(0x00,0xC0)
	TLS_RSA_WITH_AES_256_CBC_SHA	(0x00,0x35)
	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA	(0x00,0x84)
グループF	TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	(0xC0,0x2E)
	TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	(0xC0,0x32)
	TLS_ECDH_ECDSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x89)
	TLS_ECDH_RSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x8D)
	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	(0xC0,0x26)
	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	(0xC0,0x2A)
	TLS_ECDH_ECDSA_WITH_CAMELLIA_256_CBC_SHA384	(0xC0,0x75)
	TLS_ECDH_RSA_WITH_CAMELLIA_256_CBC_SHA384	(0xC0,0x79)
	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	(0xC0,0x05)
	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	(0xC0,0x0F)

A.4. セキュリティ例外型のチェックリスト

【セキュリティ例外型のチェックリスト（1/2）】

チェック項目		参照章	済
①要求設定確認	①-1）推奨セキュリティ型以上の設定が現実的ではない等の特殊事情があるケースに該当するか	3.1節	□
	①-2）推奨セキュリティ型への移行完了までの短期暫定運用を前提とし、早期の利用終了期限を含む移行計画を策定するなど、今後の対処方針を具体的に策定しているか	3.1節	□
②プロトコルバージョン設定	②-1）TLS1.0及びSSL3.0を設定有効にしたか	4.1節	□
	②-2）SSL2.0を設定無効（利用不可）にしたか	4.1節	□
	□ ②-3）TLS1.2が実装されている場合には左の□と以下の項目をチェック		
	②-4）TLS1.2について設定を有効にしたか	4.1節	□
	□ ②-5）TLS1.1が実装されている場合には左の□と以下の項目をチェック		
	②-6）TLS1.1について設定を有効にしたか	4.1節	□
	□ ②-7）プロトコルバージョン優先順位を設定できる場合には左の□と以下の項目をチェック		
	②-8）設定有効になっているプロトコルバージョンのうち、もっとも新しいバージョンによる接続を最優先にしたか。接続できない場合に、順番に一つずつ前のプロトコルバージョンでの接続するようにしたか	4.1節	□
③サーバ証明書設定	③-1）認証局の署名アルゴリズム（Certificate Signature Algorithm）と鍵長の組合せが以下のいずれかを満たしているか ・ RSA署名とSHA-256の組合せで鍵長2048ビット以上 ・ RSA署名とSHA-1の組合せで鍵長2048ビット以上	5.1節	□
	③-2）サーバの公開鍵情報（Subject Public Key Info）のSubject Public Key Algorithmと鍵長の組合せが以下を満たしているか ・ RSAで鍵長は2048ビット以上	5.1節	□
	③-3）サーバ証明書の発行・更新をした際に、鍵情報のペアを新たに生成したか	5.1節	□
	③-4）サーバ証明書の発行・更新をする際に、鍵情報のペアを新たに生成する旨の指示を仕様書・運用手順書等に記載したか	5.1節	□
	③-5）接続することが想定されている全てのクライアントに対して、警告表示が出ないように対策するか、警告表示が出るブラウザはサポート対象外であることを明示したか	5.1節	□

（続く）

【セキュリティ例外型のチェックリスト（2/2）】

	チェック項目	参照章	済
④暗号スイート設定	□ ④-i）楕円曲線暗号を利用しない場合は左の□と以下の項目をチェック		
	④-i-1）表3記載の暗号スイート（網掛けを除く）の全部または一部を設定したか	6.1節／6.5.3節	□
	④-i-2）表3記載のグループA及びグループBの暗号スイート（網掛けを除く）から少なくとも一つは設定したか	6.1節／6.5.3節	□
	④-i-3）表3記載のグループGの暗号スイートを設定したか	6.1節／6.5.3節	□
	④-i-4）表3記載の暗号スイートのグループ順番の制限[注]を守っているか	6.1節／6.5.3節	□
	④-i-5）表3記載の暗号スイート以外は、すべて利用不可の設定をしたか	6.1節／6.5.3節	□
	④-i-6）RSAの鍵長を2048ビット以上に設定したか	6.1節／6.5.3節	□
	□ ④-i-7）DHEを利用する暗号スイートを設定する場合は左の□と以下の項目をチェック		
	④-i-8）DHEの鍵長を1024ビット以上に設定したか	6.1節／6.5.3節	□
	□ ④-i-9）不特定多数に公開されるサービス等において使用されるサーバである場合には左の□と以下の項目をチェック		
	④-i-10）AES128-SHAの暗号スイートを設定したか	6.1節／6.5.3節	□
	④-i-11）DHE-DSS-DES-CBC3-SHAとDES-CBC3-SHAの少なくとも一方は設定したか	6.1節／6.5.3節	□
	□ ④-ii）楕円曲線暗号を利用する場合は左の□と以下の項目をチェック		
	④-ii-1）表3記載の暗号スイート（網掛けを含む）の全部または一部を設定したか	6.1節／6.5.3節	□
	④-ii-2）表3記載のグループA及びグループBの暗号スイート（網掛けを含む）から少なくとも一つは設定したか	6.1節／6.5.3節	□
	④-ii-3）表3記載のグループGの暗号スイートを設定したか	6.1節／6.5.3節	□
	④-ii-4）表3記載の暗号スイートのグループ順番の制限[注]を守っているか	6.1節／6.5.3節	□
	④-ii-5）表3記載の暗号スイート以外は、すべて利用不可の設定をしたか	6.1節／6.5.3節	□
	④-ii-6）ECDHE/ECDHの鍵長を256ビット以上に設定したか	6.1節／6.5.2節	□
	④-ii-7）RSAの鍵長を2048ビット以上に設定したか	6.1節／6.5.3節	□
	□ ④-ii-8）DHEを利用する暗号スイートを設定する場合は左の□と以下の項目をチェック		
	④-ii-9）DHEの鍵長を1024ビット以上に設定したか	6.1節／6.5.3節	□
	□ ④-ii-10）不特定多数に公開されるサービス等において使用されるサーバである場合には左の□と以下の項目をチェック		
	④-ii-11）AES128-SHAの暗号スイートを設定したか	6.1節／6.5.3節	□
	④-ii-12）DES-CBC3-SHAの暗号スイートを設定したか	6.1節／6.5.3節	□

［注］　許容される暗号スイートのグループ順番は以下のとおり
（128ビット安全性優先の場合）
・グループA→グループB→グループC→グループD
→グループE→グループF→グループG→グループH

（256ビット安全性優先の場合）
・グループD→グループA→グループE→グループB
→グループF→グループC→グループG→グループH

【表3】

優先順位グループ	暗号スイート名	スイート番号
グループA	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	(0x00,0x9E)
	TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x7C)
	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	(0x00,0x67)
	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256	(0x00,0xBE)
	TLS_DHE_RSA_WITH_AES_128_CBC_SHA	(0x00,0x33)
	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA	(0x00,0x45)
	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	(0xC0,0x2B)
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	(0xC0,0x2F)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x86)
	TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x8A)
	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	(0xC0,0x23)
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	(0xC0,0x27)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256	(0xC0,0x72)
	TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256	(0xC0,0x76)
	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	(0xC0,0x09)
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	(0xC0,0x13)
グループB	TLS_RSA_WITH_AES_128_GCM_SHA256	(0x00,0x9C)
	TLS_RSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x7A)
	TLS_RSA_WITH_AES_128_CBC_SHA256	(0x00,0x3C)
	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256	(0x00,0xBA)
	TLS_RSA_WITH_AES_128_CBC_SHA	(0x00,0x2F)
	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA	(0x00,0x41)
グループC	TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256	(0xC0,0x2D)
	TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256	(0xC0,0x31)
	TLS_ECDH_ECDSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x88)
	TLS_ECDH_RSA_WITH_CAMELLIA_128_GCM_SHA256	(0xC0,0x8C)
	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	(0xC0,0x25)
	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	(0xC0,0x29)
	TLS_ECDH_ECDSA_WITH_CAMELLIA_128_CBC_SHA256	(0xC0,0x74)
	TLS_ECDH_RSA_WITH_CAMELLIA_128_CBC_SHA256	(0xC0,0x78)
	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	(0xC0,0x04)
	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	(0xC0,0x0E)
グループD	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	(0x00,0x9F)
	TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x7D)
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	(0x00,0x6B)
	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256	(0x00,0xC4)
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA	(0x00,0x39)
	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA	(0x00,0x88)
	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	(0xC0,0x2C)
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	(0xC0,0x30)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x87)
	TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x8B)
	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	(0xC0,0x24)
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	(0xC0,0x28)
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_CBC_SHA384	(0xC0,0x73)
	TLS_ECDHE_RSA_WITH_CAMELLIA_256_CBC_SHA384	(0xC0,0x77)
	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	(0xC0,0x0A)
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	(0xC0,0x14)
グループE	TLS_RSA_WITH_AES_256_GCM_SHA384	(0x00,0x9D)
	TLS_RSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x7B)
	TLS_RSA_WITH_AES_256_CBC_SHA256	(0x00,0x3D)
	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256	(0x00,0xC0)
	TLS_RSA_WITH_AES_256_CBC_SHA	(0x00,0x35)
	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA	(0x00,0x84)
グループF	TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	(0xC0,0x2E)
	TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	(0xC0,0x32)
	TLS_ECDH_ECDSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x89)
	TLS_ECDH_RSA_WITH_CAMELLIA_256_GCM_SHA384	(0xC0,0x8D)
	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	(0xC0,0x26)
	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	(0xC0,0x2A)
	TLS_ECDH_ECDSA_WITH_CAMELLIA_256_CBC_SHA384	(0xC0,0x75)
	TLS_ECDH_RSA_WITH_CAMELLIA_256_CBC_SHA384	(0xC0,0x79)
	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	(0xC0,0x05)
	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	(0xC0,0x0F)

【表3（続）】

優先順位グループ	暗号スイート名	スイート番号
グループG	TLS_RSA_WITH_RC4_128_SHA	(0x00,0x05)
グループH	TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	(0x00,0x16)
	TLS_RSA_WITH_3DES_EDE_CBC_SHA	(0x00,0x0A)

Appendix B：サーバ設定編

サーバ設定を行ううえでの参考情報として､設定方法例を記載した参考ガイドを以下の URL にて公開している。

なお、利用するバージョンやディストリビューションの違いにより、設定方法が異なったり、設定ができなかったりする場合があることに留意すること。正式な取扱説明書やマニュアルを参照するとともに、一参考資料として利用されたい。

URL: https://www.ipa.go.jp/security/vuln/ssl_crypt_config.html

Appendix C：暗号スイートの設定例

暗号スイートの設定を行ううえでの参考情報として、設定方法例を記載した参考ガイドを以下の URL にて公開している。

なお、利用するバージョンやディストリビューションの違いにより、設定方法が異なったり、設定ができなかったりする場合があることに留意すること。正式な取扱説明書やマニュアルを参照するとともに、一参考資料として利用されたい。

URL: https://www.ipa.go.jp/security/vuln/ssl_crypt_config.html

Appendix D：ルート CA 証明書の取り扱い

D.1.　ルート CA 証明書の暗号アルゴリズムおよび鍵長の確認方法

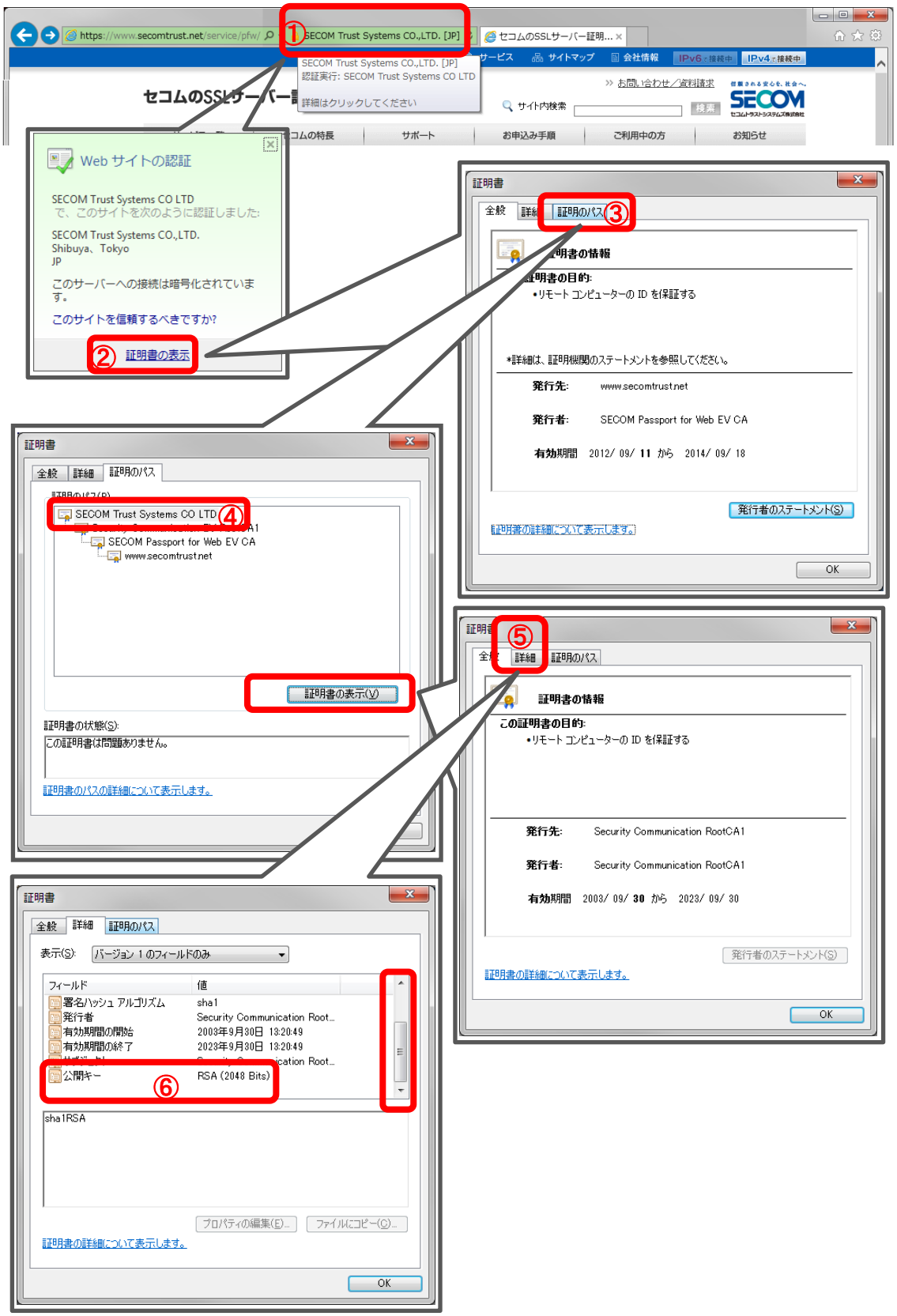
主要な認証事業者のルート CA 証明書の暗号アルゴリズムおよび鍵長を別表に掲載する。

ただし、事業者によってはサーバ証明書発行サービスを複数展開しているケースがあり、サービスによってルート CA が異なる場合があるので、どのサービスがどのルート CA の下で提供されているのかは、各事業者に確認する必要がある。

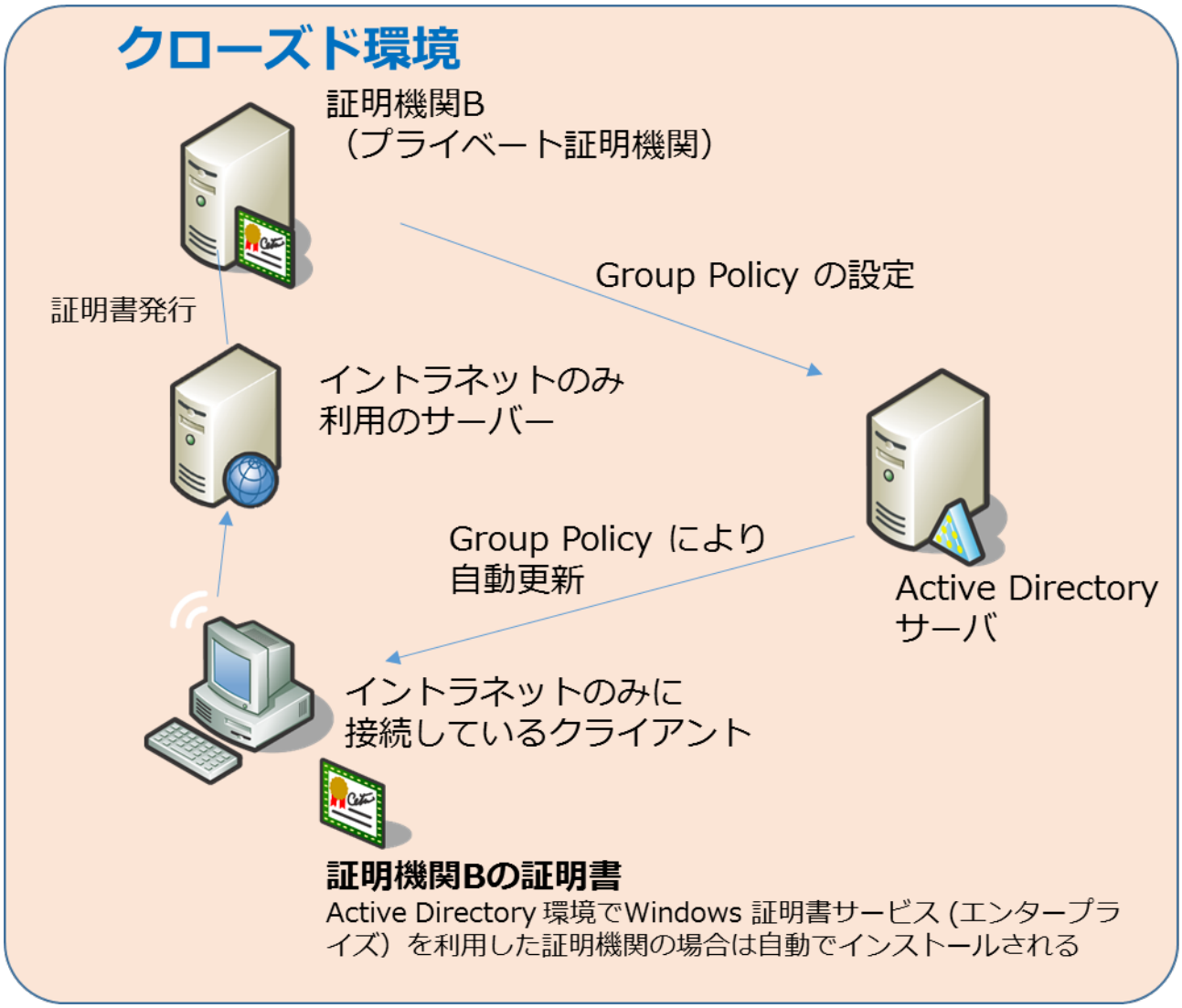
なお､サーバ証明書を発行するサービスから発行された既存のサーバ証明書を利用したサイト、あるいはテストサイトなどの URL がわかっている場合には、当該 URL にアクセスして、以下のような手順を経ることで、ルート CA の公開鍵暗号アルゴリズムおよび鍵長を確認することが可能である。

【Internet Explorer 11 で EV 証明書のサイトにアクセスする場合】

① 南京錠マーク横のサイト運営組織の表示をクリックする
② 「証明書の表示」をクリックする
③ 「証明のパス」タブをクリックする
④ 一番上に表示されている証明書（これがルート CA 証明書に当たる）を選択し､「証明書の表示」をクリックする
⑤ 「詳細」タブをクリックする
⑥ スクロールバーを一番下までスクロールさせ､「公開キー」フィールドに表示されている値（RSA (2048 Bits)）を確認する
この例では、暗号アルゴリズムが RSA、鍵長が 2048 ビットであることがわかる


①
SECOM Trust Systems CO.,LTD. [JP]
認証実行: SECOM Trust Systems CO LTD
詳細はクリックしてください
Web サイトの認証
SECOM Trust Systems CO LTD
で、このサイトを次のように認証しました:
SECOM Trust Systems CO.,LTD.
Shibuya、Tokyo
JP
このサーバーへの接続は暗号化されています。
このサイトを信頼するべきですか?
② 証明書の表示
証明書
全般 詳細 証明のパス ③
証明書の情報
証明書の目的:
•リモート コンピューターの ID を保証する
*詳細は、証明機関のステートメントを参照してください。
発行先: www.secomtrust.net
発行者: SECOM Passport for Web EV CA
有効期間 2012/ 09/ 11 から 2014/ 09/ 18
発行者のステートメント(S)
証明書の詳細について表示します。
OK
④
SECOM Trust Systems CO LTD
SECOM Passport for Web EV CA
www.secomtrust.net
証明書の表示(V)
証明書の状態(S):
この証明書は問題ありません。
証明書のパスの詳細について表示します。
⑤
詳細
証明書の情報
この証明書の目的:
•リモート コンピューターの ID を保証する
発行先: Security Communication RootCA1
発行者: Security Communication RootCA1
有効期間 2003/ 09/ 30 から 2023/ 09/ 30
発行者のステートメント(S)
証明書の詳細について表示します。
OK
表示(S): バージョン 1 のフィールドのみ
フィールド
値
署名ハッシュ アルゴリズム sha1
発行者 Security Communication Root...
有効期間の開始 2003年9月30日 13:20:49
有効期間の終了 2023年9月30日 13:20:49
公開キー RSA (2048 Bits)
⑥
sha1RSA
プロパティの編集(E)...
ファイルにコピー(C)...
証明書の詳細について表示します。
OK

D.2. Active Directory を利用したプライベートルート CA 証明書の自動更新



発行日　2015 年 5 月 22 日　第 1.0 版
　　2015 年 8 月　3 日　第 1.1 版
　　2018 年 5 月　8 日　第 2.0 版

発行者

・〒113-6591
東京都文京区本駒込二丁目 28 番 8 号
独立行政法人 情報処理推進機構
（技術本部　セキュリティセンター　暗号グループ）
INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN
2-28-8 HONKOMAGOME, BUNKYO-KU
TOKYO, 113-6591 JAPAN

・〒184-8795
東京都小金井市貫井北町四丁目 2 番 1 号
国立研究開発法人 情報通信研究機構
（サイバーセキュリティ研究所　セキュリティ基盤研究室）
NATIONAL INSTITUTE OF
INFORMATION AND COMMUNICATIONS TECHNOLOGY
4-2-1 NUKUI-KITAMACHI, KOGANEI
TOKYO, 184-8795 JAPAN