本書は古いバージョンです。
最新版 Ver.3.1.0 は以下の URL から
ダウンロードできます。
https://www.ipa.go.jp/security/crypto/guideline/gmcbt80000005ufv-att/ipa-cryptrec-gl-3001-3.1.0.pdf

TLS 暗号設定ガイドライン

～安全なウェブサイトのために（暗号設定対策編）～

Ver. 3.0.1

作成
CRYPTREC
Cryptography Research and Evaluation Commitees

発行
IPA 独立行政法人情報処理推進機構
セキュリティセンター

本書は古いバージョンです。
最新版は以下の URL からダウンロードできます。
「TLS 暗号設定ガイドライン」
https://www.ipa.go.jp/security/crypto/guideline/ssl_crypt_config.html

CRYPTREC GL-3001-3.0.1

TLS 暗号設定ガイドライン

2020 年 7 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

【修正履歴】

修正日	修正内容
2020.7.7（Ver.3.0.1）	● Appendix A の URL 誤植を修正
2020.7.7　(Ver.3.0)	● ガイドラインの名称を「SSL/TLS 暗号設定ガイドライン」から「TLS 暗号設定ガイドライン」に変更 ● 最新動向を踏まえ、内容を全面改訂（大きな差分の説明は Appendix E 参照）
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
2015.5.8　(Ver.1.0)	初版発行

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、2020 年 3 月時点における TLS 通信での安全性と相互接続性のバランスを踏まえた TLS サーバの設定方法を示すものである。

特に、SSL/TLS 暗号設定ガイドライン（version 1.x/2.x）発行以降、TLS1.3 発行[RFC8446]や SSL3.0 禁止[RFC7525]、ChaCha20-Poly1305 追加[RFC7905]、RC4 禁止[RFC7465]など、同ガイドラインに記載されている内容に大きく影響する規格化が相次いで行われており、それに伴い SSL/TLS の利用環境も大きく変化した（2.7 節参照）。

今回の改訂にあたっては、このような規格化状況及びサポート状況等の各種動向を踏まえ、プロトコルバージョンの要求設定において TLS1.3 の採用及び SSL3.0 の禁止を行った。これに伴い、各設定基準における要求設定についても大幅な変更が行われており、SSL/TLS 暗号設定ガイドライン（version 1.x/2.x）における設定基準から一段階高い安全性を求めるようになった項目も多い。例えば、推奨セキュリティ型で利用が認められていた TLS1.0 や TLS1.1 は、本ガイドラインではセキュリティ例外型のみで利用可能となった。また、鍵交換では Perfect Forward Secrecy の特性をもつ ECDHE や DHE をさらに強く推奨するようにした。

このため、SSL/TLS 暗号設定ガイドライン（version 1.x/2.x）を利用している場合であっても、本ガイドラインでの要求設定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、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、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 TLS サーバの要求設定について示す。安全性と相互接続性を踏まえたうえで、選択した設定基準としての最低限の安全性を確保するために必ず満たさなければならない項目である「遵守項目」と、当該設定基準としてよりよい安全性を実現するために満たすことが望ましい項目である「推奨項目」を決めている。

7 章では、チェックリストの対象には含めていないが、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、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、安全性と相互接続性を踏まえたうえで設定すべき要求設定として 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。

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。本ガイドラインの作成時点（2020 年 3 月）では、「推奨セキュリティ型」がもっとも安全性と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。

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を遵守したことの確認」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

本ガイドラインでは、主に Web に TLS を利用するシステムを対象に、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 TLS サーバの構築を発注するシステム担当者を想定読者としている。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。

なお、Web 以外の TLS プロトコルの利用については、参照できる部分も多いと考えているが、例えば鍵事前共有型（PSK: Pre-Shared Key）の利用方法など、十分に検討されていない項目があることに留意されたい。

1.3 ガイドラインの検討体制

本ガイドラインへの改訂にあたっては、CRYPTREC 暗号技術活用委員会の配下に設置された TLS 暗号設定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。

TLS 暗号設定ガイドラインワーキンググループは表 1 のメンバーにより構成されている。

表 1　TLS 暗号設定ガイドラインワーキンググループの構成（2020 年 6 月時点）

主査	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	漆嶌　賢二	GMO グローバルサイン株式会社 プロダクトマネジメント部　部長
委員	垣内　由梨香	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	菅野　哲	株式会社レピダム　代表取締役
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	北村　英志	グーグル合同会社　デベロッパーリレーションズ デベロッパーアドボケイト
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 主任研究員
委員	杉尾　信行	株式会社 NTT ドコモ　情報セキュリティ部
委員	杉原　弘祐	セコムトラストシステムズ株式会社 情報セキュリティサービス本部　セキュアサービス 1 部
委員	松本　照吾	アマゾンウェブサービスジャパン株式会社 パブリックセクター コンサルティング本部 シニアセキュリティコンサルタント

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 TLS の概要

2.1.1 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）[1]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[2]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[3]攻撃を回避することができる。

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[4]）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[5]や CCM[6]モードの利用が可能になった。

TLS1.3 [RFC8446]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（現在 IETF で標準化が進められているトランスポートプロトコル。内部的に TLS1.3 を利用する）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。

表 2 に TLS のバージョンの概要をまとめる。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）[7]」を参照されたい。

[1] https://ietf.org/
[2] CBC: Cipher Block Chaining
[3] BEAST: Browser Exploit Against SSL/TLS
[4] MAC: Message Authentication Code
[5] GCM: Galois/Counter Mode
[6] CCM: Counter with CBC-MAC
[7] https://www.cryptrec.go.jp/report/cryptrec-gl-2002-2013.pdf

表 2　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
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[8]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 ● SSL2.0 は利用すべきではなく、2018 年頃以降に出荷されているサーバやブラウザでは SSL3.0 は初期状態で利用不可となっている
TLS1.0 （RFC2246） （1999）	● 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 ● 2020 年 1 月時点での SSL Pulse の調査結果によれば、約 15 万の主要なサイトについて TLS1.0 が利用できるのは 62.9%
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等、TLS1.0 の安全性強化を図っている ● 2020 年 1 月時点での SSL Pulse の調査結果によれば約 15 万の主要なサイトについて TLS1.1 が利用できるのは 73.0%
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 ● 2020 年 1 月時点での SSL Pulse の調査結果によれば約 15 万の主要なサイトについて TLS1.2 が利用できるのは 96.5%

[8] POODLE: Padding Oracle On Downgraded Legacy Encryption

バージョン	概要
TLS1.3 （RFC8446） （2018）	● 暗号スイートの表記方法が変更。署名と鍵交換を暗号スイートから分離 ● 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 ● Perfect Forward Secrecy (PFS)を実現するため、静的な RSA、DH を削除 ● ハンドシェイクのデータを暗号化して保護 ● HMAC ベースの導出関数（HKDF-Expand(・), HKDF-Extract(・)）を使った鍵導出に変更 ● TLS1.2 互換に配慮し、ClientHello、ServerHello、ChangeCipherSpec が規定された ● リネゴシエーション、圧縮が削除 ● Triple DES、DSA、RC4、MD5、SHA-1、SHA-224、認証暗号（AEAD: Authenticated Encryption with Associated Data）でない CBC モードを削除 ● 共通鍵暗号は AES-GCM、AES-CCM、ChaCha20-Poly1305 のみが規定された。このうち、AES-GCM が必須、ChaCha20-Poly1305 が推奨になった ● 鍵交換は DHE、ECDHE、PSK のみが規定され、いずれかの利用が必須になった ● 署名は RSA-PSS、RSASSA-PKCS1-v1_5、ECDSA が必須になった。EdDSA も利用可能 ● 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、X25519 が推奨になった ● 2020 年 1 月時点での SSL Pulse の調査結果によれば約 15 万の主要なサイトについて TLS1.3 が利用できるのは 22.0%

2.1.2 SSL/TLS 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。図 1 の(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。

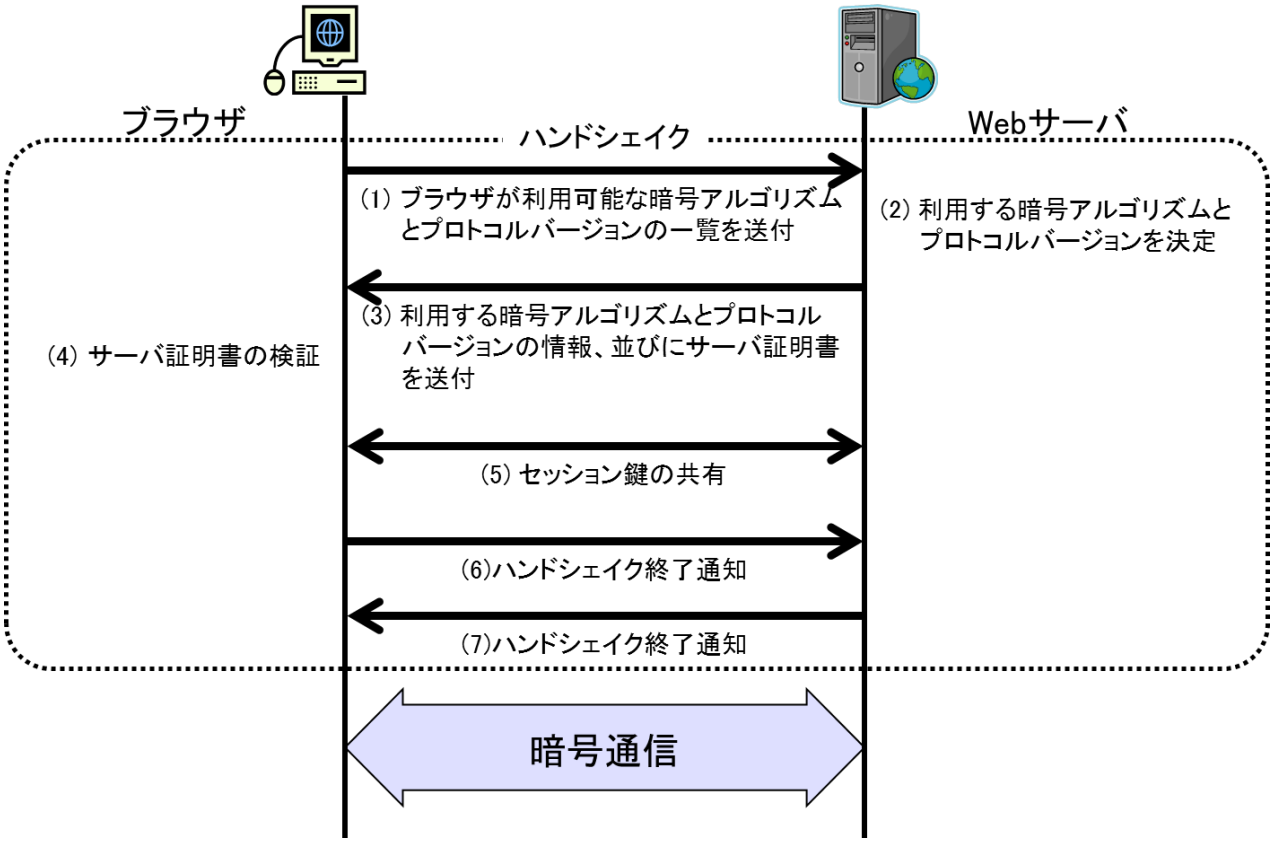


図 1　TLS プロトコル概要

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。

[9] Certificate Authority

(1) 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。

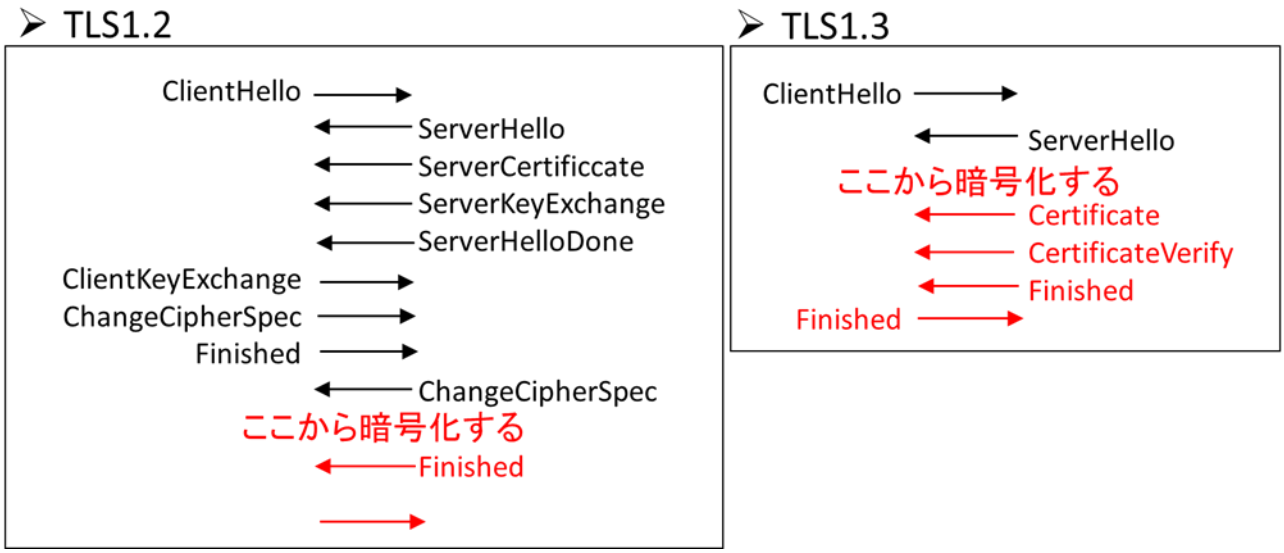


図 2　TLS1.2 と TLS1.3 との暗号化開始個所の比較

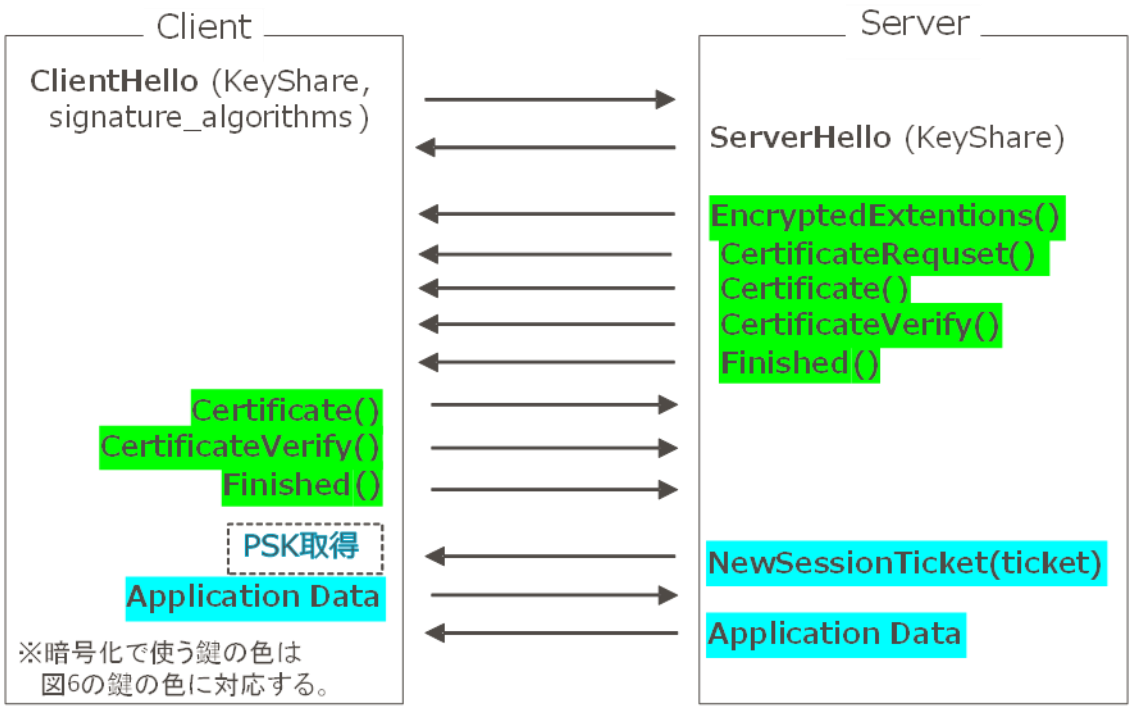


図 3　TLS1.3 のシーケンス図

(2) 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。

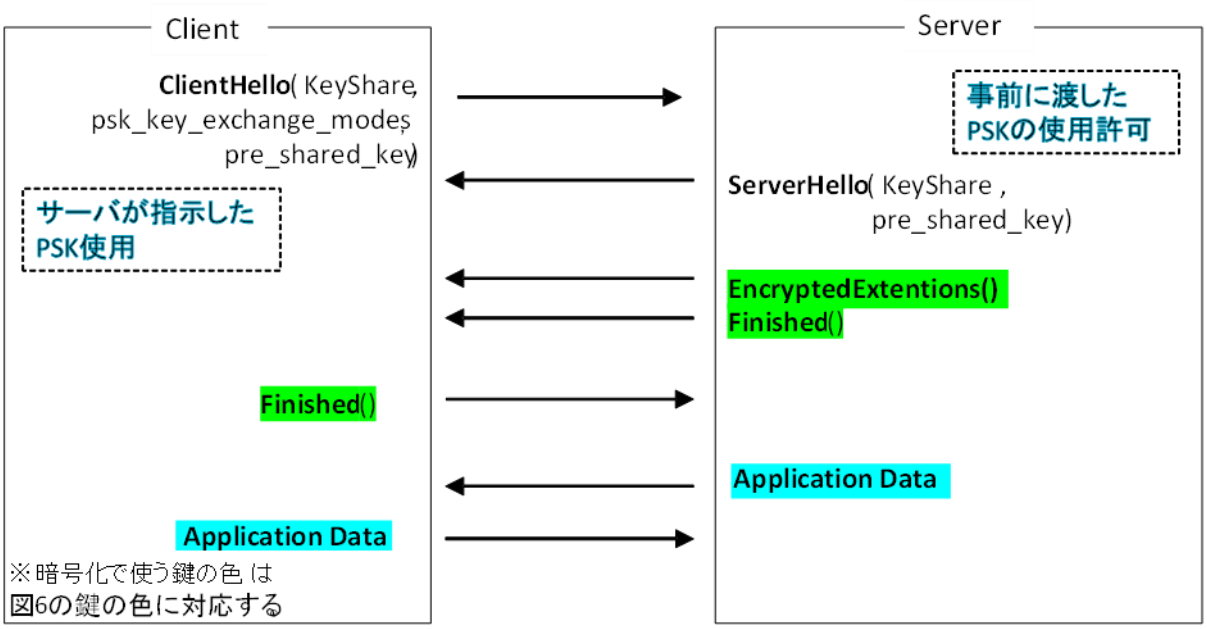


図 4　1-RTT のシーケンス図

(3) 0-RTT でアプリケーションデータを送信する機能が追加された。

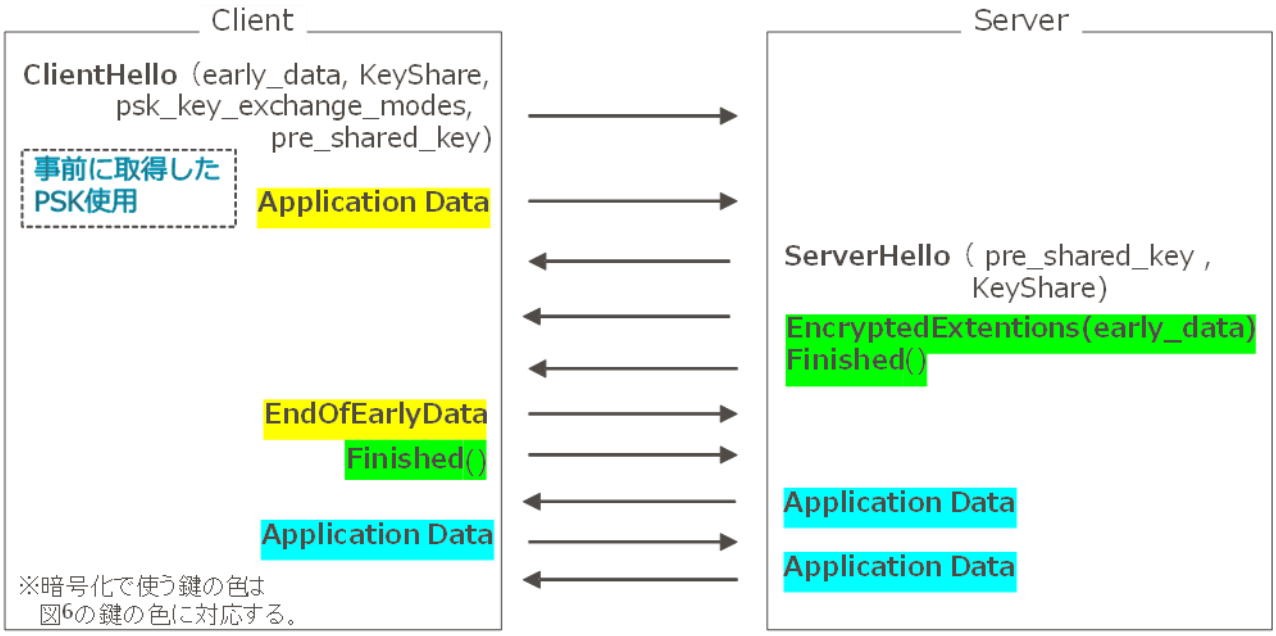


図 5　0-RTT のシーケンス図

(4) 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。

(5) 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。

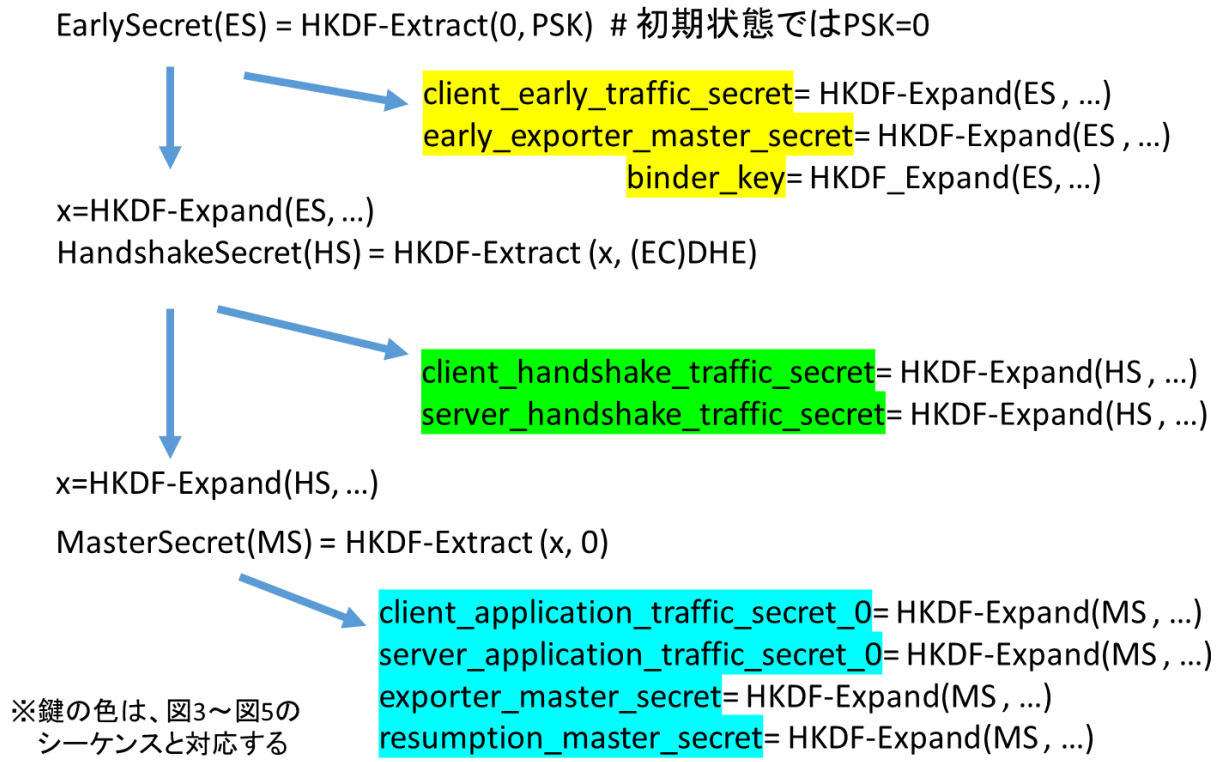


図 6　鍵の導出方法

2.2 プロトコルバージョンごとの安全性の違い

SSL/TLS は、1994 年に SSL2.0 が実装され始めた後、2020 年 3 月現在の最新版となる TLS1.3 まで 6 つのプロトコルバージョンが実装されている。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ことに留意されたい。なお、プロトコルバージョンの詳細な要求設定については、設定基準に対応する該当章を参照すること。

SSL2.0 から TLS1.3 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 3 にまとめる。なお、表 3 では BEAST 攻撃のように「プロトコルの仕様上の脆弱性」のみを対象とすることとし、OpenSSL Heartbleed Bug のように「実装に伴う脆弱性」は対象外としている。また、仕様の記載内容があいまいで当該プロトコルバージョン中の一部のパラメータ等の使用の可否が実装者の解釈に依存し、その結果、実装上の脆弱性が発生した、または発生する余地があったものであって、その後に仕様の補正・明確化が行われたような、必ずしも「仕様上の脆弱性」とまでは言い切れないものも対象外としている。

表 3　プロトコルバージョンでの安全性の違い

SSL/TLS 攻撃方法に対する耐性	TLS1.3	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	脆弱	脆弱	脆弱	脆弱
バージョンロールバック攻撃（意図したよりも古いバージョン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	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	安全	パッチ適用要	脆弱	脆弱

凡例：

- 濃い赤の「脆弱」：
  2015 年 5 月以前に発見され、修正パッチによっても回避できない「仕様上の脆弱性」
- オレンジの「脆弱」：
  2015 年 5 月以降に発見され、修正パッチによっても回避できない「仕様上の脆弱性」
- 黄色の「パッチ適用要」：
  多くの製品実装に影響を及ぼすが、修正パッチによって対策が可能な「仕様上の脆弱性」
- 緑色の「安全」：
  2020 年 3 月時点で上記に該当する「仕様上の脆弱性」が発見されていない状態。なお、「実装に伴う（製品の）脆弱性」は上記の「仕様上の脆弱性」に含めない。

2.3 サーバ証明書についての概要

サーバ証明書は、①当該サーバが、意図する相手によって管理されているサーバであることを確認する手段をクライアントに対して提供することと、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。

サーバ証明書には、表 4 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、本ガイドラインにおいて重要な項目は以下の 3 点である。

- 署名アルゴリズム
- 公開鍵情報（subject Public Key Info）に含まれる公開鍵の暗号アルゴリズム、パラメータ、鍵長
- 拡張情報のひとつ subjectAltName.dNSName に記載される、サーバの FQDN

表 4　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[10]	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

2.4 暗号スイートについての概要

TLS1.2 までの暗号スイートは「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。

[10] 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。

ただし、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。

TLS1.3 の暗号スイートでは、TLS1.2 までの暗号スイートの組から「鍵交換」と「署名」が外され、「暗号化_ハッシュ関数」だけの構成に変更となった。

実際の TLS 通信においては、サーバとクライアント間での事前通信（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（TLS1.3 の場合は「暗号スイート」の他、「鍵交換」と「署名」も含めて）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 TLS における各種処理が行われる。

このため、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。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ので、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。

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、対象とするブラウザ（クライアント）に実装されている暗号スイートを幅広く受け入れる設定をすることになる。ただし、一定以上の安全性を確保するためには、一部の旧式のブラウザ（クライアント）との相互接続性を断念してでも、一定以上の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ことにも留意されたい。

2.5 本ガイドラインでの暗号アルゴリズムに対する考え方

2.5.1 サーバ証明書で利用する暗号アルゴリズムに対する考え方

本ガイドラインにおいては、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」及び「ハッシュ関数」のうち、CRYPTREC 暗号リスト（2.6.1 節参照）の電子政府推奨暗号リストまたは推奨暗号候補リストに掲載されているものの中から原則としてサーバ証明書で利用する暗号アルゴリズムを選択する。

具体的には、表 5 に示した「署名」及び「ハッシュ関数」がサーバ証明書で利用可能な暗号アルゴリズムである。

このうち、DSA については、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が、サーバ証明書としては現時点（2020 年 3 月）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。また、デジタル署名の米国政府標準暗号 FIPS186-5 からも削除される予定であることから、本ガイドラインでは積極的には DSA の利用を勧めない。

表 5　サーバ証明書で利用可能な暗号アルゴリズム

技術分類	リストの種類	アルゴリズム名	備考
署名	電子政府推奨暗号リスト	RSASSA PKCS#1 v1.5（RSA）	
		DSA	本ガイドラインでは積極的には利用を勧めない
		ECDSA	
ハッシュ関数	電子政府推奨暗号リスト	SHA-256	
		SHA-384	現在までに利用実績はほとんどない
		SHA-512	

2.5.2 暗号スイートで利用する暗号アルゴリズムに対する考え方

本ガイドラインにおいては、SSL/TLS 用の暗号スイートとして IANA TLSCipher Suites に登録されているもの及び RFC8446 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（2.6.1 節参照）に掲載されているものの中から原則として暗号スイートで利用する暗号アルゴリズムを選択する。

本節では、本ガイドラインにおける暗号スイートで利用する暗号アルゴリズムに対する原則的な考え方を示す。最終的にどの暗号アルゴリズムが利用できるかについては、どの設定基準（3.1 節参照）を選択したかにより異なるため、詳細については該当章を参照されたい。

■（全ての設定基準における）暗号スイートでの利用推奨暗号アルゴリズム

原則的には電子政府推奨暗号リストまたは推奨暗号候補リストに掲載されている暗号アルゴリズムを「(全ての設定基準における) 暗号スイートでの利用推奨暗号アルゴリズム」として規定する。具体的には、表 6 にその一覧を示す。

例外的に、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おり、またデジタル署名の米国政府標準暗号 FIPS186-5 からも削除される予定であることから、積極的には DSA の利用を勧めない。このため、DSA は表 6 から除外するが、利用を妨げるものではない。

表 6　暗号スイートでの利用推奨暗号アルゴリズム

	CRYPTREC 暗号リストでの標記			備考
	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	DHE (Ephemeral DH)	PFS 特性をもつ DHE を選択するのがセキュリティ上望ましい
			ECDHE (Ephemeral ECDH)	PFS 特性をもつ ECDHE を選択するほうがセキュリティ上望ましい
署名	署名	電子政府推奨暗号リスト	ECDSA	
			RSASSA PKCS#1 v1.5 (RSA)	
			RSASSA-PSS	TLS1.3 のみ利用可能
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	
			Camellia	TLS1.2 までで利用可能
	暗号利用モード	電子政府推奨暗号リスト	CCM	
			CCM_8	CCM の退縮版なので、CCM を利用するほうがセキュリティ上望ましい
			GCM	
	認証暗号	推奨候補暗号リスト	ChaCha20-Poly1305	
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	

■設定基準により暗号スイートでの取り扱いが異なる暗号アルゴリズム

運用監視暗号リストに掲載されている暗号アルゴリズムは、安全性に問題があり互換性維持での利用を前提としているため、原則として、本ガイドラインの「設定基準により暗号スイートでの取り扱いが異なる暗号アルゴリズム」として位置付ける。

例外的に、CBC、DH、ECDH については電子政府推奨暗号リストに記載されているが、以下の理由により、「設定基準により暗号スイートでの取り扱いが異なる暗号アルゴリズム」に含めるものとする。

- CBC については、暗号利用モード単体としては一定の安全性を有しており電子政府推奨暗号リストに掲載されているが、実装上の問題によっては TLS1.0 以前のプロトコルバージョンで利用する場合に TLS としての脆弱性につながる要因が存在することが分かっている。
- DH と ECDH については、電子政府推奨暗号リストに記載の暗号アルゴリズムであり安全性上の問題はないが、2.5.3 節にあるように TLS での利用にあたっては Perfect Forward

Secrecy の性質を有する DHE や ECDHE の暗号スイートを選択するほうがセキュリティ上望ましい。

ここに該当する暗号アルゴリズムは具体的には表 7 に示したものであり、選択する設定基準によって「利用推奨」されることも「利用禁止」されることもあり得る。どちらに該当するかは選択する設定基準の該当章を参照されたい。なお、「利用推奨」とも「利用禁止」とも言及されない場合もあるが、その場合には「利用を推奨しないものの、自らの責任において利用することは妨げない」と理解されたい。

表 7　設定基準により暗号スイートでの取り扱いが異なる暗号アルゴリズム

	CRYPTREC 暗号リストでの標記			備考
	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	DH は PFS 特性を持たないので、PFS 特性をもつ DHE を選択するほうがセキュリティ上望ましい
			ECDH	ECDH は PFS 特性を持たないので、PFS 特性をもつ ECDHE を選択するほうがセキュリティ上望ましい
		運用監視暗号リスト	RSAES PKCS#1 v1.5 (RSA)	脆弱性が見つかっているが、利用実績上、無視できない
暗号化	暗号利用モード	電子政府推奨暗号リスト	CBC	TLS1.1 以前の実装時に脆弱性が見つかっている
ハッシュ関数	ハッシュ関数	運用監視暗号リスト	SHA-1	HMAC での利用のため、耐衝突困難性のリスクは SHA-1 単体の場合より小さい

■利用禁止暗号アルゴリズム

本ガイドラインでは、IANA TLSCipher Suites に登録及び RFC8446 に採用されている暗号スイートで使われている暗号アルゴリズムのうち、CRYPTREC として安全性評価を実施しておらず安全性が不明であったり実際に暗号解読されたりするなどの技術的要因、もしくは輸出規制や暗号政策をはじめとする政策的要因などにより、**いかなる設定基準であっても利用すべきではない暗号アルゴリズムを「利用禁止暗号アルゴリズム」として新たに規定する。原則的には CRYPTREC 暗号リストに掲載されていない暗号アルゴリズムが該当し、具体的には、表 8 に示した暗号アルゴリズム（2020 年 1 月 7 日時点）である。**

例外は以下の 2 点である。

- 3-key Triple DES と RC4 は CRYPTREC 暗号リストの運用監視暗号リストに掲載されている

が、SSL3.0 以前で主に使われていた暗号アルゴリズムであり、TLS1.0 以降では AES や Camellia などが利用可能であることから、本ガイドラインで SSL3.0 を利用禁止とすることに合わせて表 8 に含めるものとする。
なお、RC4 は、2021 年 3 月 31 日をもって運用監視暗号リストから削除されることが決定した。

- EdDSA については、CRYPTREC 暗号リストには現時点（2020 年 3 月）で掲載されていない暗号アルゴリズムであるが、TLS1.3 やデジタル署名の米国政府標準暗号 FIPS186-5 に採用されたり、最近のオープンソースソフトウェアでの搭載が進んでいたりすることなどを鑑み、表 8 には含めない。現時点（2020 年 3 月）では、本ガイドラインでは利用を推奨しないものの、自らの責任において利用することは妨げない。

表 8　暗号スイートでの利用禁止暗号アルゴリズム（2020 年 1 月 7 日時点）

	CRYPTREC 暗号リストでの標記			備考
	技術分類	リストの種類	アルゴリズム名	
署名	署名	対象外	GOST R 34.10-2012	
暗号化	64 ビット ブロック暗号	対象外	RC2, EXPORT-RC2	
			IDEA	
			DES, EXPORT-DES	
			GOST 28147-89	
			Magma	
		運用監視暗号リスト	3-key Triple DES	SSL3.0 を利用禁止にしたため
	128 ビット ブロック暗号	対象外	Kuznyechik	
			ARIA	
			SEED	
	暗号利用モード	対象外	CTR_OMAC	
	ストリーム暗号	運用監視暗号リスト	RC4*注）	SSL3.0 を利用禁止にしたため
		対象外	EXPORT-RC4	
ハッシュ関数	ハッシュ関数	対象外	MD5	
			GOST R 34.11-2012	

*注）RC4 は、2021 年 3 月 31 日をもって運用監視暗号リストから削除されることが決定した

2.5.3 Perfect Forward Secrecy の重要性－秘密鍵漏えい時の影響範囲を狭める手法

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、「鍵共有・守秘」に掲載されている暗号アルゴリズムである。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、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、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 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 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。

2.5.4 DH(E)/ECDH(E)での鍵長設定についての注意

鍵交換について、暗号スイート上は鍵長の規定がない。これは、鍵交換の安全性は鍵長にも大きく影響されるためであり、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や消費リソース等が過大にかかるようになり、実用性を損なうことにつながる。このため、安全性と処理性能、消費リソース等のトレ

ードオフを考えて適切な鍵長を選択する必要がある。

また、様々な鍵長に対応できるように、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。特に、鍵交換で RSA を使う場合と、DH(E)や ECDH(E)を使う場合とでは、鍵長の扱いが全く異なることに留意する必要がある。

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、現在では鍵長 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。

一方、DH(E)と ECDH(E)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。

このうち、ECDH(E)については比較的最近になって急速に普及してきたことから、当初からセキュリティを考慮して通常 256 ビット以上の鍵長を利用するように実装されている。このため、鍵長をデフォルト設定のままで利用したとしても、セキュリティ上の問題につながるような短い鍵長が使われる可能性はかなり低いと言える。

ところが、DH(E)については、古く SSL の時代から利用されていたこともあり、デフォルト設定での鍵長が製品やバージョンによって大きく異なることが知られている。実際、2.7 節の図 10 の DH 鍵交換の強度からもわかるように、現在の主要なブラウザは鍵長 2048 ビットの DH(E)が利用可能であるにも関わらず、2020 年 1 月時点で 11.0%ものサイトで鍵長 1024 ビット以下の DH(E)が使われている。このことは、デフォルト設定で鍵長 1024 ビットが利用可能になっている製品を利用し続けているサイトやレガシーシステムとの相互接続性確保のためにあえて鍵長 1024 ビット以下も利用可能に設定しているサイトが相当数存在していることを示している。

このように、DH(E)を利用する場合には鍵長の設定を正しく行っておかなければ予期せぬタイミングで 1024 ビット以下の弱い鍵長が使われる可能性が依然として残っていることから、利用を許可する鍵長を明示的に設定（もしくは確認）できない製品を利用する場合には、DH(E)を含む暗号スイートは選択しないようにすべきである。

2.6 暗号アルゴリズムの安全性

2.6.1 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。それぞれのリストの位置づけは以下の通りである。

- 電子政府推奨暗号リスト：
  CRYPTREC により安全性及び実装性能が確認された暗号技術について、市場における利用実績が十分であるか今後の普及が見込まれると判断され、当該技術の利用を推奨するもののリスト
- 推奨候補暗号リスト：
  CRYPTREC により安全性及び実装性能が確認され、今後、電子政府推奨暗号リストに掲載される可能性のある暗号技術のリスト
- 運用監視暗号リスト：
  実際に解読されるリスクが高まるなど、推奨すべき状態ではなくなったと CRYPTREC により確認された暗号技術のうち、互換性維持のために継続利用を容認するもののリスト。互換性維持以外の目的での利用は推奨しない。

「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 30 年度版）[12]」（平成 30 年 7 月 25 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）

[11] https://www.cryptrec.go.jp/list.html
[12] https://www.nisc.go.jp/active/general/pdf/kijyun30.pdf

2.6.2 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、“ビットセキュリティ（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[13]）で表現され、鍵長[14]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 ビットセキュリティ”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 はハッシュ長）が解読計算量の（最大）理論値である。

“ビットセキュリティ”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 ビットセキュリティ”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 5[15]では、表 9 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(E)のこと
- ECDH-xxx：鍵長が xxx ビット（NIST 曲線パラメータ P-xxx を利用）の ECDH(E)のこと
- ECDSA-xxx：鍵長が xxx ビット（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。TLS では、暗号スイートで決めるハッシュ関数は HMAC として利用されるので、こちらの安全性に影響する。
- 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。

[13] 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
[14] ハッシュ関数の場合はハッシュ長（ダイジェスト長ともいう）に相当する
[15] NIST SP800-57, Recommendation for Key Management – Part 1: General (Revision 5)

表 9　NIST SP800-57 でのビットセキュリティの取り扱い方針（WG で加工）

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

2.7 SSL/TLS の利用環境の変化

2015 年 1 月から 2020 年 2 月の期間に発行された SSL/TLS に関する RFC 55 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、TLS1.3 の規格化のほか、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。

また、2019 年 8 月に NIST は TLS に関する新たなガイドライン SP800-52 Revision 2[16]を発表した。Rev.2 では、①2024 年 1 月 1 日までに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 TLS1.3 をサポートすること、②TLS1.2 以上の利用を原則とすること、③RSA での鍵交換を止める、などが規定された。

表 10　2015 年 1 月から 2020 年 2 月の期間に「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

発効年	RFC	Title	プロトコル	サーバ証明書	暗号スイート	内容
2015.2	7465	Prohibiting RC4 Cipher Suites	×	×	○	RC4 禁止
2015.4	7507	TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks	×	×	○	新暗号スイートの定義
2015.5	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
2015.6	7568	Deprecating Secure Sockets Layer Version 3.0	○	×	×	SSL3.0 禁止
2016.3	7836	Guidelines on the Cryptographic Algorithms to Accompany the Usage of Standards GOST R 34.10-2012 and GOST R 34.11-2012	×	×	△	GOST 暗号の追加
2016.6	7905	ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)	×	×	○	ChaCha20-Poly1305 の暗号スイート追加
2016.8	7919	Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)	×	×	△	DHE で利用するパラメータのネゴシエーションを整理
2018.8	8422	Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier	×	×	○	TLS1.2 以前での ECDHE、ECDSA、EdDSA の利用方法を規定

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

発効年	RFC	Title	プロトコル	サーバ証明書	暗号スイート	内容
2018.9	8442	ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS 1.2 and DTLS 1.2	×	×	○	TLS1.2 での ECDHE_PSK の暗号スイートの追加 ※本ガイドラインの対象外の暗号スイート
2018.8	8446	The Transport Layer Security (TLS) Protocol Version 1.3	○	×	○	TLS1.3 の規格化
2018.8	8447	IANA Registry Updates for TLS and DTLS	×	×	△	暗号スイートのスイート番号の管理
2020.2	8734	Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS) Version 1.3	×	×	△	TLS1.3 での Brainpool 曲線の追加。ただし、IETF として勧めた方式ではない

凡例： ○：影響あり ×：影響なし △：関連はあるものの直接的な影響はなし

上記のような規格化の流れもあり、SSL Pulse[17]の集計データによれば、図 7 から図 10 のように、2014 年から 2020 年の間にプロトコルバージョンや暗号アルゴリズム等のサポート状況に大きな変動が発生していることがわかる。

例えば、SSL3.0 禁止の RFC が発行された 2015 年前後でサポート状況が全く異なること、TLS1.1 以前のバージョンはいずれも 2017 年の時よりも 2020 年のほうでサポート率が低下し、TLS1.2 に移行していること、などが図 7 から読み取れる。RC4 についても同様で、RC4 禁止の RFC が発行された 2015 年前後でサポート状況が全く異なることが図 8 から読み取れる。

鍵交換に関しては、2013 年のスノーデン事件をきっかけに 2.5.3 節に記載したように PFS の重要性が認識され、2014 年にはわずか 5%しかサポートされていなかったものが瞬く間にサポートされるようになったことが図 9 からわかる。一方、DH(E)の鍵長については、サーバ証明書でコントロール可能な RSA での鍵長 2048 ビットへの移行と比較するとかなり遅れており、未だに 10%以上で鍵長 1024 ビットの DH(E)が使われていることが図 10 から伺える。

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

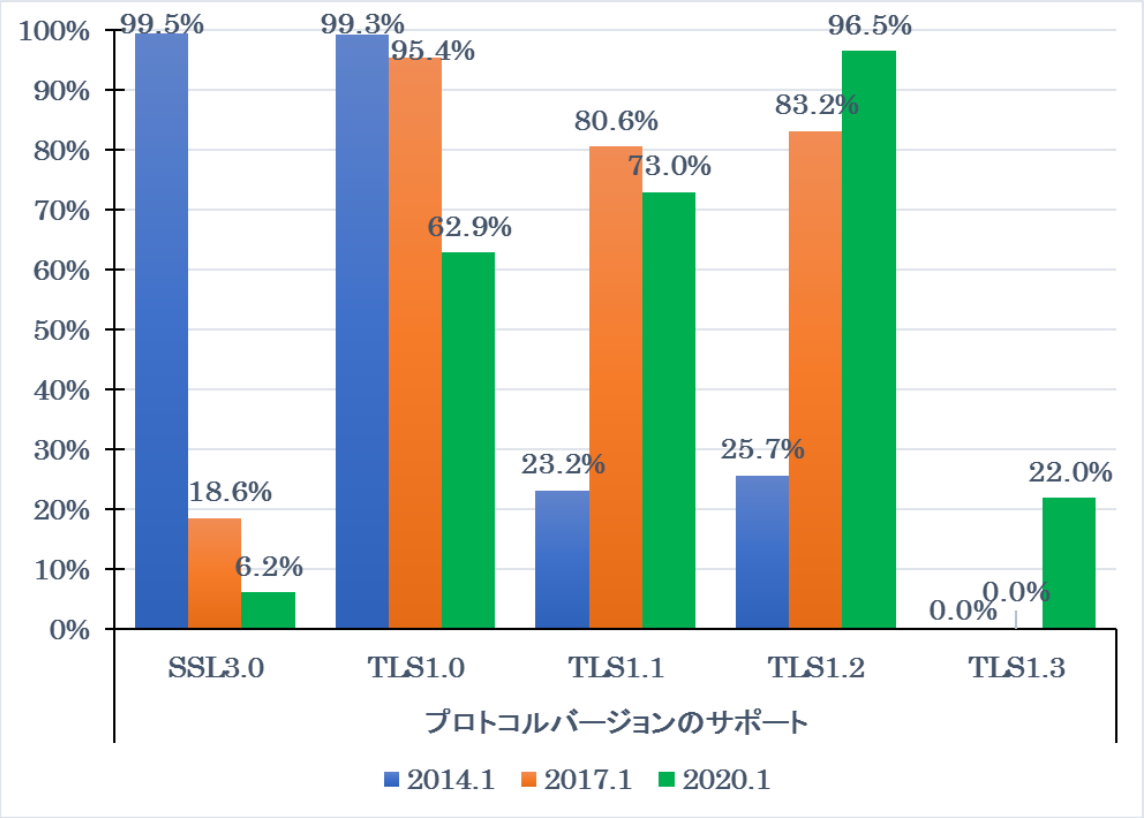


図 7　サポートされるプロトコルバージョンの推移（SSL Pulse による集計データを加工）

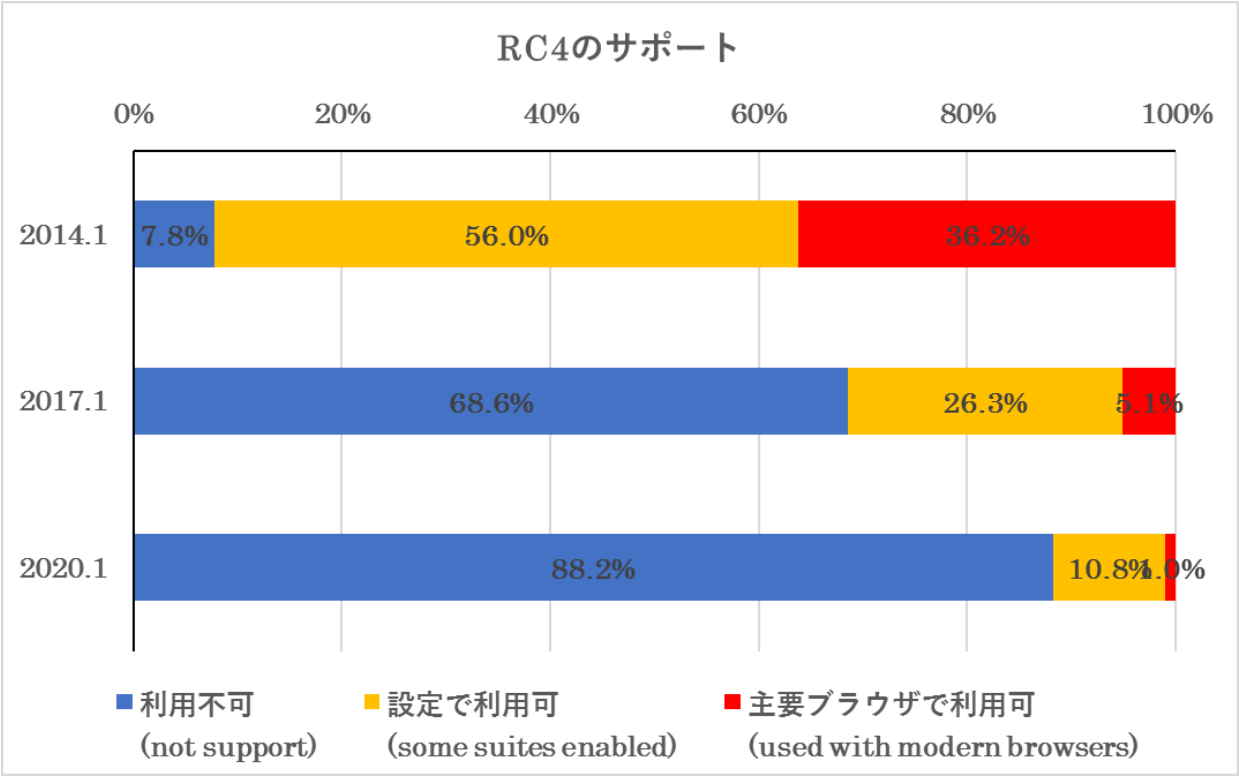


図 8　RC4 のサポート状況（SSL Pulse による集計データを加工）

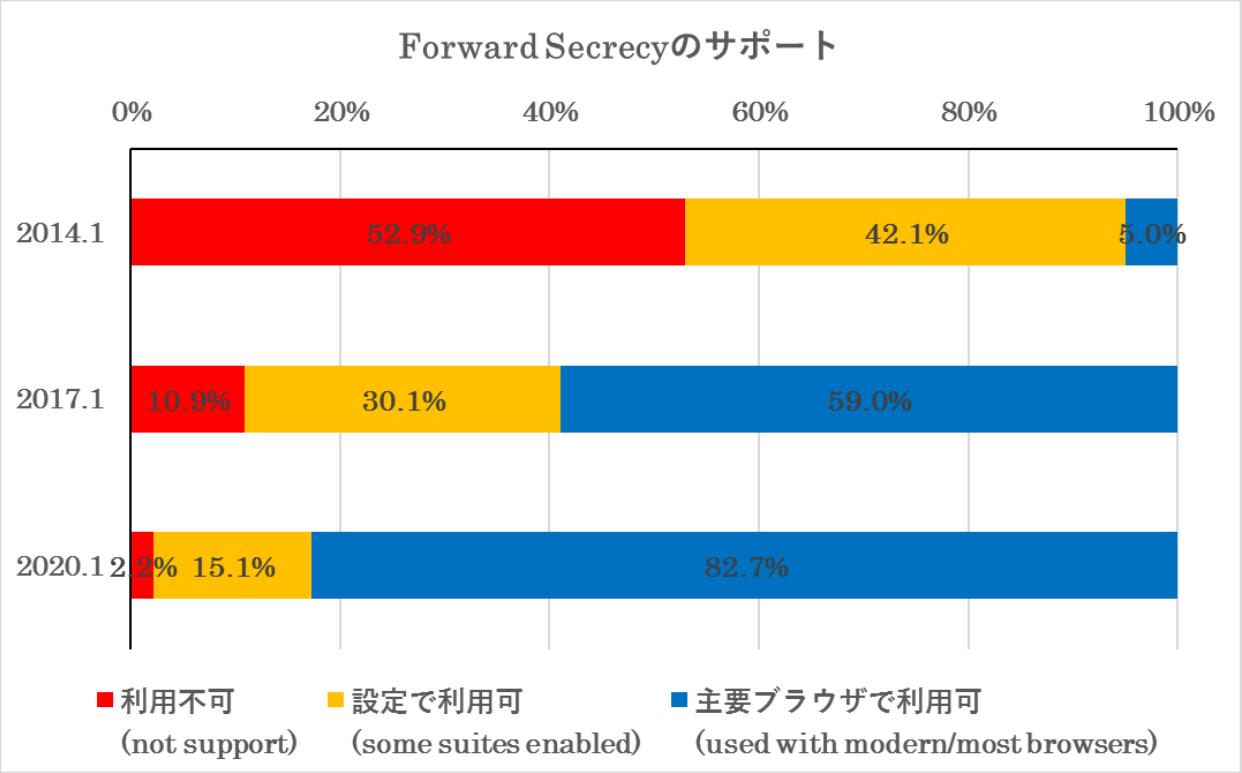


図 9　Perfect Forward Secrecy のサポート状況（SSL Pulse による集計データを加工）

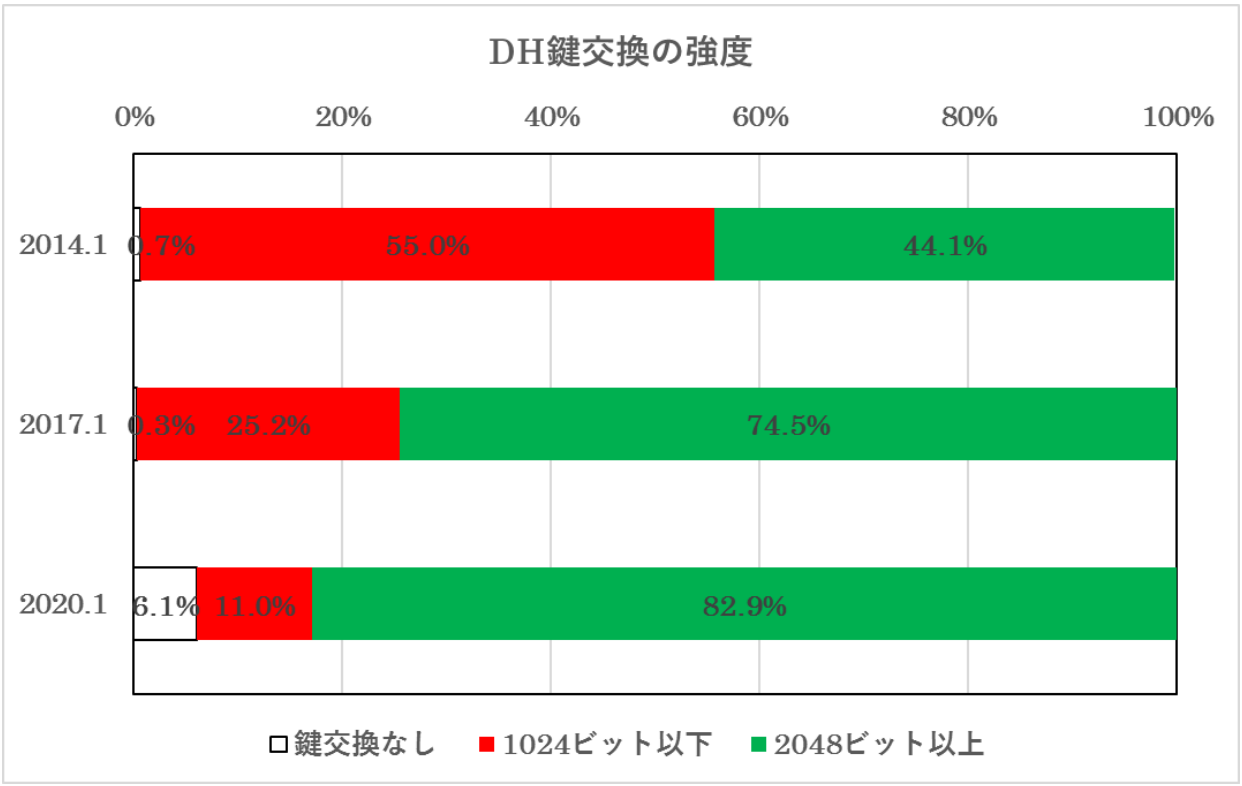


図 10　DH(E)鍵交換の鍵長の推移（SSL Pulse による集計データを加工）

【コラム①】 常時 HTTPS 化に伴う留意点

2015 年頃からウェブの全 HTTPS 化が叫ばれてきた。これは、基本は HTTP、ログイン画面や支払い画面など機密情報を扱うページのみ HTTPS にするという慣習を止め、すべてのウェブページを HTTPS で配信しようというものである。この動きは一部のブラウザベンダのみが提唱してきた話ではなく、すべてのメジャーなブラウザベンダや IETF も巻き込んだインターネット全体の動きに発展してきた。

ウェブサイトの常時 HTTPS 化は手間のかかる作業だが、ユーザの安全性を守るだけでなく、プライバシーに配慮してよりパワフルな機能を実現することができる下地となる。まだ常時 HTTPS 化を行っていないようであれば検討することを勧める。

なぜ今、全 HTTPS 化が叫ばれているのか？

HTTP は通信内容を保証しないため、例えば通信経路の第三者がサービスを改ざんしても、それが本来配信されるべき内容であるかどうかはユーザには分からない。これは大きな問題を引き起こす可能性がある。

例えば、インターネットプロバイダやホテル、ショップ等が提供する Wi-Fi サービス、公共機関等に設置された Wi-Fi サービスなどを利用して通信を行う場合、中継地点に悪意のある第三者が存在する可能性が否定できない。そういった第三者は、コンテンツに広告を挿入したり、マルウェアを埋め込んでユーザを危険にさらしたりするだけでなく、ユーザをトラッキングしたり、盗聴する可能性もある。

通信を HTTPS 化することによって、通信相手が意図したものであることを保証し、通信内容が改ざんされていないことを保証し、そして通信内容を傍受されないことを保証することができるようになる。

また、HTTPS 化することによって、様々なブラウザのパワフルで新しい機能が利用可能になる。例えば Service Worker は、ブラウザのタブを開いていなくてもスクリプトを動かすことを可能にし、ウェブページのオフライン動作やプッシュ通知を可能にする。その他にも生体認証やセキュリティキーを利用可能にする WebAuthn、ブラウザ上の支払いを便利にする Payment Request API なども含まれる。ネットワーク面でも、HTTP の新しいバージョンである HTTP/2 では通信をシリアライズかつ並列化することでパフォーマンスを強化し、さらに次の世代となる HTTP/3 の基礎となる QUIC についても HTTPS が前提となっている。

一方、元々パワフルだった古い機能も HTTP では利用できなくなるような移行が行われた。例えば、位置情報を取得する Geolocation、デバイスの傾きを取得する Device motion / orientation などがこれに該当する。

このように、ウェブ全体を HTTPS 化する動きに伴って、「HTTPS だから安全」と捉えるのではなく、「HTTP を危険」と捉える方向にシフトしつつある。このため、例えば各種ブラウザでは、これまで HTTPS の場合に錠前を表示してきたが、今後は HTTPS では錠前のないデフォルト表示を行い、反対に HTTP の場合はユーザに注意を促す表記を行う計画がある。

図 11 は Google の公開している Transparency Report から引用したものである。日本の HTTPS 対応は他の国々に大きく遅れを取っていたが、2016 年時点では飛び抜けて低かった HTTPS 対応状況が 2017 年を境にここ数年で目覚ましい挽回を見せていることが分かる。

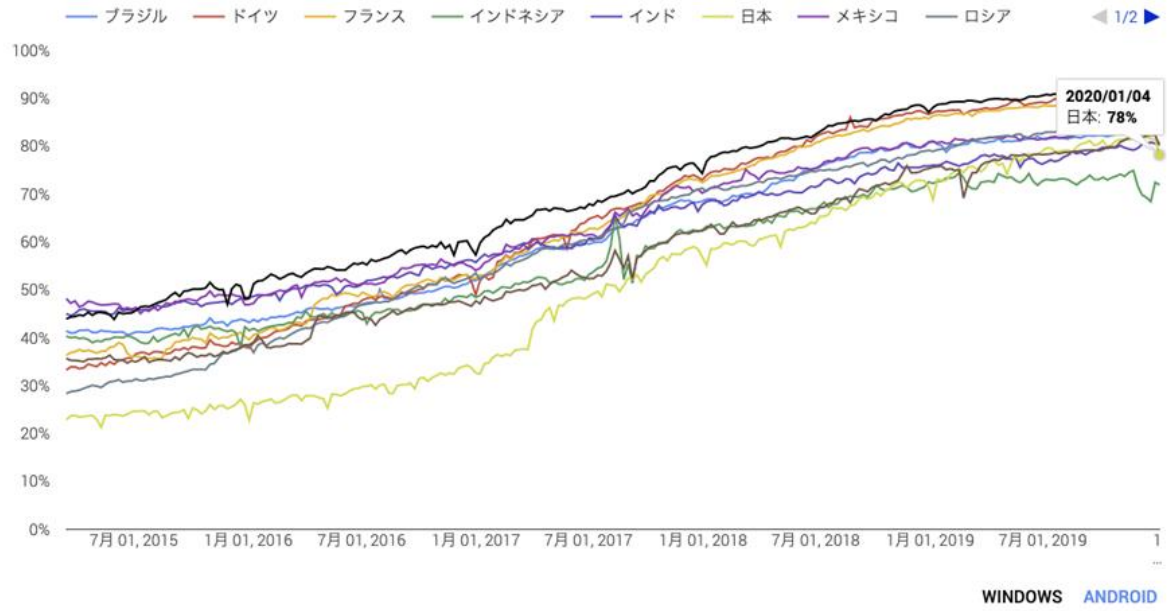


図 11　HTTPS 対応状況（[出典] Google Transparency Report）

常時 HTTPS 化に伴う留意事項

サーバから配信するすべてのリソースを HTTPS 化するには、以下の留意点がある。

- Mixed-Contents

  ブラウザは、表示しているページに暗号化されていない（HTTP の）リソースが含まれている場合、これを Mixed-Contents、つまり「暗号化されているものとされていないものが混在しているコンテンツ」であり、「安全ではない」ものとして扱う。その結果、ブラウザによっては URL バーに「保護されていない通信」などと表示されることがある。さらに、ブラウザによっては、HTTP 通信の部分のロードが最終的にすべてブロックされるようになる予定がある。

  Mixed-Contents を避けるためには、ウェブサイトは、ページに掲載するすべてのコンテンツやリソースについて HTTPS 化されている状態にしなければならない。そのため、自分自身ですべてのコンテンツやリソースを管理していれば対応しやすいが、問題は広告や埋め込みコンテンツなどが自分以外のオーナーによってリソースが管理されている場合である。全 HTTPS 化が叫ばれて久しいので、広告や埋め込みコンテンツの多くはすでに HTTPS 化されているが、もしそうではない場合は、提供元に働きかける必要がある。

  https://developers.google.com/web/fundamentals/security/prevent-mixed-content/what-is-mixed-content

また、すべてのコンテンツやリソースが HTTPS で配信されているかどうかをマニュアルで確認するのは簡単なことではない。そのようなときには、ブラウザの提供する Reporting API を使えば、ブラウザが Mixed Contents を発見次第、レポートを送るという機能を利用することができる。

https://developers.google.com/web/fundamentals/security/prevent-mixed-content/fixing-mixed-content

- ストレージの移行

  元々HTTP だったウェブサイトを HTTPS に変更する際、ブラウザ上のストレージにも注意が必要である。ブラウザは同じドメインでもスキーマが異なれば別オリジンとして扱うため、HTTP と HTTPS の間ではストレージが共有されない。そのため、移行に際しては Local Storage や Indexed DB などのストレージを移し替える必要が出てくる。

- Cookie に Secure オプションを追加

  Cookie には"Secure"属性という HTTPS 通信に限って送信する機能がある。セッション ID などの機微な情報が誤って HTTP 上に流れてしまうと、セッションを盗まれてしまう危険性もあるので、必ず“Secure”属性を付けるようにすべきである。

- HSTS で強制的にデフォルト HTTPS 化

  全 HTTPS 化を進めるにしても、ウェブサイトに残された既存のリンクをすべて残らず変更するのは容易ではない。そうして残った HTTP 経由のリンクでアクセスした場合でも、HTTPS にリダイレクトすれば問題ないと思えるかもしれないが、その（リダイレクトされる前の）アクセスでセッション情報など、重要な情報が平文で送られてしまうのは好ましくない。

  その対策として、HTTP Strict Transport Security（HSTS）ヘッダーを送ることで、ブラウザに強制的に HTTPS でアクセスさせることを記憶させることができる。これを利用することで、それ以降一定期間、同じブラウザからのアクセスは、リンクが HTTP で記述されていても、自動的に HTTPS に変換した上でリクエストが送信される。HSTS については、7.4.1 節も参照されたい。

  なお、HTTP のリンクが残っている限り、初めてそのウェブサイトに訪れる際には HTTP でアクセスせざるを得ないケースは存在する。このため、既存のリンクについても、可能な限り HTTPS 化することを怠らないようにすべきである。

  https://developers.google.com/web/fundamentals/security/encrypt-in-transit/enable-https

PART I：

サーバ構築における設定要求について

3. 設定基準の概要

本章では、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。

このため、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
- サーバ証明書の設定
- 暗号スイートの設定
- TLS を安全に使うために考慮すべきこと

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 段階の設定基準を設ける。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 11 のとおりである。

また、4 章から 6 章にかけて、それぞれの設定基準に対応する形で、プロトコルバージョン、サーバ証明書、暗号スイートの 3 つの項目についての詳細な要求設定を定める。表 12 に要求設定の概要を記す。

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**。本ガイドラインの作成時点（2020 年 3 月）では、「推奨セキュリティ型」がもっとも安全性と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、安全性が低いプロトコルバージョンである TLS1.0、TLS1.1 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 TLS1.0、TLS1.1 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、TLS1.0、TLS1.1 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 TLS1.0、TLS1.1 を利用不可に設定するように変更される可能性がある。

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 11　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 ＜5 章＞	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悪影響を及ぼすと予想される情報を、安全性を優先して TLS 通信を行うような場合に採用する設定基準 **※ 高い安全性の確保を必要とするケースでの利用を想定している** ＜利用例＞ • 特にセキュリティが重要視されるシステムを構築する場合	本ガイドライン作成時点（2020 年 3 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している	本ガイドラインで対象とするブラウザが搭載されている PC、スマートフォン等であれば相互接続性を確保できると期待される。
推奨セキュリティ型 ＜4 章＞	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（相互接続性）の実現をバランスさせて TLS 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＞ • 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広範囲（不特定多数）に提供する場合 • 新規に社内システムを構築する場合	本ガイドライン作成時点（2020 年 3 月）における標準的な安全性水準を実現している	本ガイドラインで対象とするブラウザが搭載されている PC、スマートフォンを含め、多くの製品・システムで相互接続性を確保できると期待される。 ※サポートが終了しているバージョンが古い OS やブラウザ、発売開始から長期間経過している古い機器、IoT 用途などの一部の機器類については接続できない可能性がある。
セキュリティ例外型 ＜6 章＞	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 TLS 通信を行う場合に採用する設定基準 **※推奨セキュリティ型への移行完了までの暫定運用を想定している** ＜利用例＞ • 利用するサーバやクライアントの実装上の制約、又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	本ガイドライン作成時点（2020 年 3 月）において、標準的な安全性水準を満たしていないプロトコルバージョンや暗号スイート等が使用される可能性があることを認識する必要がある	既存システムを含め、ほぼすべての機器に対して相互接続性が確保されると期待される。

表 12　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">【遵守】
想定対象</td><td>高い安全性の確保を
必要とするケース</td><td>一般的な利用形態</td><td>推奨セキュリティ型への
移行完了までの暫定運用</td></tr>
<tr><td colspan="2">【遵守】暗号
アルゴリズム</td><td>高セキュリティ型での
利用禁止暗号を利用不可</td><td>推奨セキュリティ型での
利用禁止暗号を利用不可</td><td>セキュリティ例外型での
利用禁止暗号を利用不可</td></tr>
<tr><td colspan="2">【遵守】
バージョン</td><td>TLS1.3 (必須) 及び TLS1.2
(オプション)</td><td>TLS1.2 (必須) 及び TLS1.3
(オプション)</td><td>TLS1.3 ～ TLS1.0 の
いずれか</td></tr>
<tr><td rowspan="4">【推奨】推奨暗号アルゴリズム設定</td><td>鍵交換</td><td>①256 ビット以上の
ECDHE
②2048 ビット以上の DHE</td><td>①256 ビット以上の ECDHE
②2048 ビット以上の DHE</td><td>①1024 ビット以上の DHE
又は 256 ビット以上の
ECDHE
②2048 ビット以上の RSA,
1024 ビット以上の DH,
256 ビット以上の ECDH
のいずれか</td></tr>
<tr><td>暗号化</td><td colspan="2">128 ビット及び 256 ビットの AES 又は Camellia、
もしくは ChaCha20-Poly1305</td><td>128 ビット及び 256 ビット
の AES 又は Camellia、
もしくは ChaCha20-
Poly1305</td></tr>
<tr><td>暗号利用
モード</td><td>GCM, CCM, CCM_8 の
いずれか</td><td>①GCM, CCM, CCM_8 の
いずれか
②CBC</td><td>①GCM, CCM, CCM_8 の
いずれか
②CBC</td></tr>
<tr><td>ハッシュ
関数</td><td>SHA-384 又は SHA-256</td><td>①SHA-384 又は SHA-256
②SHA-1</td><td>SHA-384, SHA-256,
SHA-1 のいずれか</td></tr>
<tr><td rowspan="3">【遵守】サーバ証明</td><td>公開キー</td><td colspan="2">鍵長 2048 ビット以上の RSA
又は 256 ビット以上の ECC</td><td>鍵長 2048 ビット以上の
RSA</td></tr>
<tr><td>署名アルゴ
リズム</td><td colspan="2">鍵長 2048 ビット以上の RSA
又は 256 ビット以上の ECDSA</td><td>鍵長 2048 ビット以上の
RSA</td></tr>
<tr><td>ハッシュ
関数</td><td>SHA-256 以上</td><td>SHA-256</td><td>SHA-256</td></tr>
</table>

3.2 要求設定における遵守項目と推奨項目

本ガイドラインでは、それぞれの設定基準における要求設定として「遵守項目」と「推奨項目」が決められている。

「遵守項目」とは、選択した設定基準としての最低限の安全性を確保するために必ず満たさなければならない項目のことであり、「・・・しなければならない」と記載されている。この項目を一つでも満たさないものがあれば当該設定基準を達成したとは見なさない。

「推奨項目」とは、当該設定基準としてよりよい安全性を実現するために満たすことが望ましい項目のことであり、「・・・すべきである」と記載されている。この項目は、できるだけ設定するように推奨するものであるが、強制するものではない。状況に応じて判断されたい。

表 13　要求設定における遵守項目と推奨項目

要求設定	遵守項目	プロトコルバージョン	利用禁止プロトコルバージョンを利用不可にする設定
		サーバ証明書	利用する暗号アルゴリズムと鍵長の設定
			発行・更新時の鍵情報の生成方法の明確化
			警告表示の回避方法の明確化
		暗号スイート	利用禁止暗号アルゴリズムを利用不可にする設定
			公開鍵暗号の鍵長の設定
	推奨項目	プロトコルバージョン	利用プロトコルバージョンの優先順位付け
		暗号スイート	利用推奨暗号アルゴリズムのみでの設定
			推奨暗号スイートの優先順位付け

3.3 チェックリスト

図 12 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を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を遵守したことを、発注者が文書として確認する手段を与えること

である。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、以下の方針でチェックの確認を行う。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。

- 「要求設定確認」は選択した設定基準を採用してよいかの確認項目であり、「該当」にチェックが入る場合に限り、当該設定基準を採用してよい
- 「遵守項目」については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必要である
- 「遵守項目」以外については該当章の記載内容と照らし合わせ、設定の実態に即して適切なほうのチェックボックスにチェックを入れる。

＜チェックリストの例＞

【高セキュリティ型チェックリスト】

選択した設定基準に対応したチェックリストのチェックシートを用いる

要求設定が満たされていることを確認したら「済」にチェックを入れる

要求設定の詳細な内容が記載されている章番号

選択したチェックシートを利用してよいかの確認項目であり、「該当」にチェックが入る場合に限り利用してよい

「遵守項目」は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必要である

「遵守項目」以外については記載内容を確認し、設定の実態に即して適切なほうのチェックボックスにチェックを入れる

	チェック項目	参照章		
①要求設定確認	①-1)【遵守項目】高セキュリティ型の設定基準であるか	3.1節	□該	
②プロトバージョン設定	[illegible]たか	5.1節	□済	
	[illegible]たか。ただし、TLS1.2を明確に利用をチェックする	5.1節	□済	□設定せず
	②-3)【遵守項目】SSL2.0からTLS1.1までを設定無効（利用不可）にしたか	5.1節	□済	
③サーバ証明書設定	③-1)【遵守項目】サーバの公開鍵情報（Subject Public Key Info）のSubject Public Key [illegible]gorithm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 ・ 楕円曲線暗[illegible]	5.2節	□済	
	③-2)【遵守項[illegible]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以上の組合せで鍵長2048ビット以上 ・ ECDSAとSHA-256以上の組合せで鍵長256ビット（NIST P-256）以上	5.2節	□済	
	③-3)【遵守項目】サーバ証明書の発行・更新を行う際に、自ら公開鍵と秘密鍵の鍵ペアを生成する場合には、新たな公開鍵と秘密鍵の鍵ペアを生成しているか	5.2節	□済	
	③-4)【遵守項目】上記③-3)についての指示を仕様書や運用手順書等に明記したか	5.2節	□済	
	③-5)【遵守項目】 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	5.2節	□済	
④暗号スイート設定	④-1)【遵守項目】表21記載の暗号アルゴリズムを全てを設定無効（利用不可）にしたか	5.3節	□済	
	④-2) ECDHEを利用する暗号スイートを設定するか。 設定しない場合は「設定せず」をチェックする（④-2-1のチェック不要）		□設定する	□設定せず
	④-2-1)【遵守項目】ECDHEの鍵長を256[illegible]		[illegible]済	
	④-3) DHEを利用する暗号スイートを設定す[illegible] 設定しない場合は「設定せず」をチェック[illegible]		[illegible]設定する	□設定せず
	④-3-1)【遵守項目】DHEの鍵長を2048ビット以上に設定したか		□済	
	④-4)【推奨項目】表22記載の暗号アルゴリズムを組み合わせた暗号スイートのみで設定しているか。 設定しない／できない場合は「設定せず」をチェックする	5.3節	□済	□設定せず
	④-5) 暗号スイートの優先順位が設定できるか。 設定できない場合は「設定不可」をチェックする（④-5-1のチェック不要）		□設定可	□設定不可
	④-5-1)【推奨項目】表24記載の暗号スイートの優先順位で設定したか。優先順位どおりに設定できない／しない場合は「設定せず」をチェックする	5.3節	□済	□設定せず
⑤附録	⑤) Appendix C：暗号スイートの設定例の最新版のドキュメントを参照して設定したか。参照していない場合には「参照せず」をチェックする	Appendix C	□参照済	□参照せず

図 12　チェックリスト（高セキュリティ型の例）

4. 推奨セキュリティ型の要求設定

4.1 プロトコルバージョン

SSL2.0 から TLS1.3 までの安全性の違い（2.2 節参照）を踏まえ、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「遵守項目」を以下のように定める。

【プロトコルバージョンの遵守項目】

- TLS1.2 を設定有効としなければならない。
- SSL2.0 から TLS1.1 までを設定無効（利用不可）にしなければならない。

【プロトコルバージョンの推奨項目】

- TLS1.3 については、実装されているのであれば、設定有効にすべきである。ただし、TLS1.3 が実装されている場合であっても、TLS1.3 を明確に利用しないと判明している場合には TLS1.3 の設定有効化を必須とするものではない。

表 14　推奨セキュリティ型でのプロトコルバージョン設定

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

○：設定有効　　×：設定無効化　　－：実装なし

* TLS1.3 を明確に利用しないと判明している場合には、TLS1.3 設定有効化を必須としない

4.2 サーバ証明書

サーバ証明書についての「遵守項目」を以下のように定める。

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。

【サーバ証明書の遵守項目】

- 本ガイドライン作成時点（2020 年 3 月）で、多くのパブリック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しなければならない。

表 15　推奨セキュリティ型でのサーバ証明書設定

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公開鍵情報（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（ecPublicKey; OID = 1.2.840.10045.2.1）で鍵長 256 ビット以上（例：NIST P-256 の場合、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・更新を行う際に、自ら公開鍵と秘密鍵の鍵ペアを生成する場合には、既存の公開鍵と秘密鍵の鍵ペアを再利用せず、新たな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ブラウザ（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 ➢ 警告表示が出るブラウザ（クライアント）はサポート対象外であることを明示する、または警告表示が出ないサポート対象のブラウザ（クライアント）を明示する ➢ 7.2.8 節に従って、信頼できるプライベート認証局のルート CA 証明書を予めインストールする

4.3 暗号スイート

暗号スイートについての「遵守項目」及び「推奨項目」を以下のように定める。

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。

【暗号スイートの遵守項目】

- 以下の条件に該当する暗号アルゴリズムのいずれかを含む暗号スイートが選択されないようにするため、表 16 に記載される暗号アルゴリズム全てを設定無効（利用不可）としなければならない。
  - 2.5.2 節の表 8 に掲載されている暗号アルゴリズム
  - 2.5.2 節の表 7 に掲載されている暗号アルゴリズムのうち、DH 及び ECDH

表 16　推奨セキュリティ型での利用禁止暗号アルゴリズム一覧

鍵交換		DH, ECDH
署名		GOST R 34.10-2012
暗号化	ブロック暗号	RC2, EXPORT-RC2, IDEA, DES, EXPORT-DES, GOST 28147-89, Magma, 3-key Triple DES, Kuznyechik, ARIA, SEED
	暗号利用モード	CTR_OMAC
	ストリーム暗号	RC4, EXPORT-RC4
ハッシュ関数		MD5, GOST R 34.11-2012

- 鍵交換で ECDHE を利用する場合には鍵長 256 ビット以上を、DHE を利用する場合には鍵長 2048 ビット以上での設定をしなければならない。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。

【暗号スイートの推奨項目】

- 以下の条件に該当する表 17 に記載される暗号アルゴリズムを組み合わせた暗号スイートのみが選択されるようにすべきである。
  - 2.5.2 節の表 6 に掲載されている暗号アルゴリズム
  - 2.5.2 節の表 7 に掲載されている暗号アルゴリズムのうち、CBC 及び SHA-1*)

表 17　推奨セキュリティ型での利用推奨暗号アルゴリズム一覧

鍵交換		ECDHE, DHE
署名		ECDSA, RSASSA PKCS#1 v1.5 (RSA), RSASSA-PSS（TLS1.3 のみ）
暗号化	ブロック暗号	AES, Camellia（TLS1.2 のみ）
	暗号利用モード	GCM, CCM, CCM_8, CBC
	ストリーム暗号	ChaCha20-Poly1305
ハッシュ関数		SHA-256, SHA-384, SHA-1*)

- 安全性が高い暗号スイートから優先的に接続するようにするため、表 18 のようにグループ A、グループ B、・・・の順に暗号スイートの優先順位を設定すべきである。

表 18　推奨セキュリティ型での暗号スイートの優先順位

	【第一優先】 暗号利用モードの種類	【第二優先】 鍵交換の種類	【第三優先】 ハッシュ関数
グループ A	GCM, CCM, CCM_8	ECDHE	SHA-256, SHA-384
グループ B	GCM, CCM, CCM_8	DHE	SHA-256, SHA-384
グループ C	CBC	ECDHE	SHA-256, SHA-384
グループ D	CBC	ECDHE	SHA-1*)
グループ E	CBC	DHE	SHA-256, SHA-384
グループ F	CBC	DHE	SHA-1*)

*) 暗号スイートとしての SHA-1 は HMAC の構成部品として利用されており、2020 年時点においても HMAC-SHA1 の安全性について問題はない。しかしながら、実運用上は、HMAC の構成部品であっても SHA-1 の利用を避けるなどの世の中の動きもあり、今後の動向次第では推奨から外す可能性があることに留意されたい。

上記の方針に従った暗号スイートの推奨設定を表 19 に示す。表 19 は、表 17 に記載される暗号アルゴリズムのみで組み合わせた暗号スイートの全てが網羅されており、さらに表 18 に従ってグループ A、グループ B、グループ C の 3 つにグループ分けされている。

優先順位の設定ができない場合は表 19 全体の暗号スイートから、優先順位の設定ができる場合には各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。

優先順位を設定する際には、グループ A に含まれる暗号スイート、グループ B に含まれる暗号スイート、グループ C に含まれる暗号スイートの順になるように調整する。この際、各グループ内での暗号スイートの優先順位は任意に定めてよい。また、グループ B 以降の暗号スイートについては選択しなくてもよい。

表 19　推奨セキュリティ型での暗号スイートの推奨設定

	TLS1.2 を利用する場合
グループ A	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256
	TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256
	TLS_ECDHE_ECDSA_WITH_AES_128_CCM
	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384
	TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384
	TLS_ECDHE_ECDSA_WITH_AES_256_CCM
	TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8
	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
グループ B	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
	TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256
	TLS_DHE_RSA_WITH_AES_128_CCM
	TLS_DHE_RSA_WITH_AES_128_CCM_8
	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
	TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384
	TLS_DHE_RSA_WITH_AES_256_CCM
	TLS_DHE_RSA_WITH_AES_256_CCM_8
	TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
グループ C	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256
	TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256
	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_CBC_SHA384
	TLS_ECDHE_RSA_WITH_CAMELLIA_256_CBC_SHA384

グループ D	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
グループ E	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256
グループ F	TLS_DHE_RSA_WITH_AES_128_CBC_SHA
	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA
	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA

	TLS1.3 を利用する場合
グループ A	TLS_AES_128_GCM_SHA256
	TLS_AES_128_CCM_SHA256
	TLS_AES_128_CCM_8_SHA256
	TLS_AES_256_GCM_SHA384
	TLS_CHACHA20_POLY1305_SHA256
鍵交換	ECDHE（優先）
	DHE
署名	ECDSA
	RSA-PSS
	RSASSA-PKCS1-v1_5

5. 高セキュリティ型の要求設定

5.1 プロトコルバージョン

SSL2.0 から TLS1.3 までの安全性の違い（2.2 節参照）を踏まえ、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「遵守項目」を以下のように定める。なお、サーバとクライアントの両方が TLS1.3 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。

【プロトコルバージョンの遵守項目】

- SSL2.0 から TLS1.1 までを設定無効（利用不可）にしなければならない。
- TLS1.3 及び TLS1.2 を設定有効にしなければならない。ただし、TLS1.2 を明確に利用しないと判明している場合には TLS1.2 の設定有効化を必須とするものではない。

表 20　高セキュリティ型でのプロトコルバージョン設定

TLS1.3	TLS1.2*	TLS1.1	TLS1.0	SSL3.0	SSL2.0
○	○	×	×	×	×

○：設定有効　　×：設定無効化　　－：実装なし

* TLS1.2 を明確に利用しないと判明している場合には、TLS1.2 設定有効化を必須としない

5.2 サーバ証明書

サーバ証明書についての「遵守項目」を以下のように定める。

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。

【サーバ証明書の遵守項目】

- 本ガイドライン作成時点（2020 年 3 月）で、パブリック認証局から入手可能なサーバ証明書のうち、非常に安全性が高いものを利用しなければならない。

表 21　高セキュリティ型でのサーバ証明書設定

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公開鍵情報（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（ecPublicKey; OID = 1.2.840.10045.2.1）で鍵長 256 ビット以上（例：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 以上の組合せ（例：SHA-256 の場合：sha256WithRSAEncryption; OID = 1.2.840.113549.1.1.11）で鍵長 2048 ビット以上 ● ECDSA と SHA-256 以上の組合せ（例：SHA-256 の場合：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・更新を行う際に、自ら公開鍵と秘密鍵の鍵ペアを生成する場合には、既存の公開鍵と秘密鍵の鍵ペアを再利用せず、新たな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ブラウザ（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 ➢ 警告表示が出るブラウザ（クライアント）はサポート対象外であることを明示する、または警告表示が出ないサポート対象のブラウザ（クライアント）を明示する ➢ 7.2.8 節に従って、信頼できるプライベート認証局のルート CA 証明書を予めインストールする

5.3 暗号スイート

暗号スイートについての「遵守項目」及び「推奨項目」を以下のように定める。

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。

【暗号スイートの遵守項目】

- 以下の条件に該当する暗号アルゴリズムのいずれかを含む暗号スイートが選択されないようにするため、表 22 に記載される暗号アルゴリズム全てを設定無効（利用不可）としなければならない。
  - 2.5.2 節の表 7 に掲載されている暗号アルゴリズム
  - 2.5.2 節の表 8 に掲載されている暗号アルゴリズム

表 22　高セキュリティ型での利用禁止暗号アルゴリズム一覧

鍵交換		DH, ECDH, RSAES PKCS#1 v1.5 (RSA)
署名		GOST R 34.10-2012
暗号化	ブロック暗号	RC2, EXPORT-RC2, IDEA, DES, EXPORT-DES, GOST 28147-89, Magma, 3-key Triple DES, Kuznyechik, ARIA, SEED
	暗号利用モード	CBC, CTR_OMAC
	ストリーム暗号	RC4, EXPORT-RC4
ハッシュ関数		MD5, SHA-1, GOST R 34.11-2012

- 鍵交換で ECDHE を利用する場合には鍵長 256 ビット以上を、DHE を利用する場合には鍵長 2048 ビット以上での設定をしなければならない。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。

【暗号スイートの推奨項目】

- 以下の条件に該当する表 23 に記載される暗号アルゴリズムを組み合わせた暗号スイートのみが選択されるようにすべきである。
  - 2.5.2 節の表 6 に掲載されている暗号アルゴリズム

表 23　高セキュリティ型での利用推奨暗号アルゴリズム一覧

鍵交換		ECDHE, DHE
署名		ECDSA, RSASSA PKCS#1 v1.5 (RSA), RSASSA-PSS（TLS1.3 のみ）
暗号化	ブロック暗号	AES, Camellia（TLS1.2 のみ）
	暗号利用モード	GCM, CCM, CCM_8
	ストリーム暗号	ChaCha20-Poly1305
ハッシュ関数		SHA-256, SHA-384

- 安全性が高い暗号スイートから優先的に接続するようにするため、のようにグループ A、グループ B の順に暗号スイートの優先順位を設定すべきである。

表 24　高セキュリティ型での暗号スイートの優先順位

	【第一優先】 鍵交換の方式
グループ A	ECDHE
グループ B	DHE

上記の方針に従った暗号スイートの推奨設定を表 25 に示す。表 25 は、表 23 に記載される暗号アルゴリズムのみで組み合わせた暗号スイートの全てが網羅されており、さらに表 24 に従ってグループ A、グループ B の 2 つにグループ分けされている。

優先順位の設定ができない場合は表 25 全体の暗号スイートから、優先順位の設定ができる場合には各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。

優先順位を設定する際には、グループ A に含まれる暗号スイート、グループ B に含まれる暗号スイートの順になるように調整する。この際、各グループ内での暗号スイートの優先順位は任意に定めてよい。また、グループ B の暗号スイートについては選択しなくてもよい。

表 25　高セキュリティ型での暗号スイートの推奨設定

	TLS1.3 を利用する場合
グループ A	TLS_AES_256_GCM_SHA384
	TLS_CHACHA20_POLY1305_SHA256
	TLS_AES_128_GCM_SHA256
	TLS_AES_128_CCM_SHA256
	TLS_AES_128_CCM_8_SHA256
鍵交換	ECDHE（優先）
	DHE
署名	ECDSA
	RSA-PSS
	RSASSA-PKCS1-v1_5

	TLS1.2 を利用する場合
グループ A	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384
	TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384
	TLS_ECDHE_ECDSA_WITH_AES_256_CCM
	TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8
	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256
	TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256
	TLS_ECDHE_ECDSA_WITH_AES_128_CCM
	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
グループ B	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
	TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384
	TLS_DHE_RSA_WITH_AES_256_CCM
	TLS_DHE_RSA_WITH_AES_256_CCM_8
	TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
	TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256
	TLS_DHE_RSA_WITH_AES_128_CCM
	TLS_DHE_RSA_WITH_AES_128_CCM_8

6. セキュリティ例外型の要求設定

6.1 プロトコルバージョン

SSL2.0 から TLS1.3 までの安全性の違い（2.2 節参照）を踏まえ、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「遵守項目」を以下のように定める。

【プロトコルバージョンの遵守項目】

- SSL3.0 及び SSL2.0 を設定無効（利用不可）にしなければならない
- TLS1.2 については、実装されているのであれば、設定有効にしなければならない。
- TLS1.0 及び TLS1.1 については、やむを得ず利用する場合に限り設定を容認する。必要性に応じて設定可否の判断を行わなければならない。

【プロトコルバージョンの推奨項目】

- TLS1.3 については、実装されているのであれば、設定有効にすべきである。ただし、TLS1.3 が実装されている場合であっても、TLS1.3 を明確に利用しないと判明している場合には TLS1.3 の設定有効化を必須とするものではない。
- 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接続するように設定すべきである。

表 26　セキュリティ例外型でのプロトコルバージョン設定

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

○：設定有効　△：やむを得ず利用する場合に限り設定容認　×：設定無効化　－：実装なし

* TLS1.3 を明確に利用しないと判明している場合には、TLS1.3 設定有効化を必須としない

6.2 サーバ証明書

サーバ証明書についての「遵守項目」を以下のように定める。

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。

【サーバ証明書の遵守項目】

- 本ガイドライン作成時点（2020 年 3 月）で、多くのパブリック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しなければならない。なお、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 を利用すべきである。

表 27　セキュリティ例外型でのサーバ証明書設定

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公開鍵情報（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
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・更新を行う際に、自ら公開鍵と秘密鍵の鍵ペアを生成する場合には、既存の公開鍵と秘密鍵の鍵ペアを再利用せず、新たな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ブラウザ（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 ➢ 警告表示が出るブラウザ（クライアント）はサポート対象外であることを明示する、または警告表示が出ないサポート対象のブラウザ（クライアント）を明示する ➢ 7.2.8 節に従って、信頼できるプライベート認証局のルート CA 証明書を予めインストールする

6.3 暗号スイート

暗号スイートについての「遵守項目」及び「推奨項目」を以下のように定める。

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。

【暗号スイートの遵守項目】

- 以下の条件に該当する暗号アルゴリズムのいずれかを含む暗号スイートが選択されないようにするため、表 28 に記載される暗号アルゴリズム全てを設定無効（利用不可）としなければならない。
  - 2.5.2 節の表 8 に掲載されている暗号アルゴリズム

表 28　セキュリティ例外型での利用禁止暗号アルゴリズム一覧

署名		GOST R 34.10-2012
暗号化	ブロック暗号	RC2, EXPORT-RC2, IDEA, DES, EXPORT-DES, GOST 28147-89, Magma, 3-key Triple DES, Kuznyechik, ARIA, SEED
	暗号利用モード	CTR_OMAC
	ストリーム暗号	RC4, EXPORT-RC4
ハッシュ関数		MD5, GOST R 34.11-2012

- 鍵交換で DHE・DH を利用する場合には鍵長 1024 ビット以上を、RSA を利用する場合には鍵長 2048 ビット以上を、ECDHE・ECDH を利用する場合には鍵長 256 ビット以上での設定をしなければならない。なお、DHE・DH 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・DH を含む暗号スイートは選定すべきではない。

【暗号スイートの推奨項目】

- 以下の条件に該当する表 17 に記載される暗号アルゴリズムを組み合わせた暗号スイートのみが選択されるようにすべきである。
  - 2.5.2 節の表 6 に掲載されている暗号アルゴリズム
  - 2.5.2 節の表 7 に掲載されている暗号アルゴリズム

表 29　セキュリティ例外型での利用推奨暗号アルゴリズム一覧

鍵交換		DHE, ECDHE, RSAES PKCS#1 v1.5 (RSA), DH, ECDH
署名		RSASSA PKCS#1 v1.5 (RSA), ECDSA RSASSA-PSS（TLS1.3 のみ）
暗号化	ブロック暗号	AES, Camellia（TLS1.2 まで）
	暗号利用モード	GCM, CCM, CCM_8, CBC
	ストリーム暗号	ChaCha20-Poly1305
ハッシュ関数		SHA-256, SHA-384, SHA-1

- 安全性が高い暗号スイートから優先的に接続するようにするため、表 30 のようにグループ X、グループ Y、グループ Z の順に暗号スイートの優先順位を設定すべきである。

表 30　セキュリティ例外型での暗号スイートの優先順位

	【第一優先】 暗号利用モードの種類	【第二優先】 Perfect Forward Secrecy (PFS)の特性有無
グループ X	GCM, CCM, CCM_8	PFS あり
グループ Y	GCM, CCM, CCM_8	PFS なし
グループ Z	CBC	PFS あり・PFS なし

上記の方針に従った暗号スイートの推奨設定を表 31 に示す。表 31 は、表 29 に記載される暗号アルゴリズムのみで組み合わせた暗号スイートの全てが網羅されており、さらに表 30 に従ってグループ X、グループ Y、グループ Z の 3 つにグループ分けされている。

優先順位の設定ができない場合は表 31 全体の暗号スイートから、優先順位の設定ができる場合には各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。

優先順位を設定する際には、グループ X に含まれる暗号スイート、グループ Y に含まれる暗号スイート、グループ Z に含まれる暗号スイートの順になるように調整する。この際、各グループ内での暗号スイートの優先順位は任意に定めてよい。また、グループ Y 以降の暗号スイートについては選択しなくてもよい。

表 31　セキュリティ例外型での暗号スイートの推奨設定

	TLS1.2 を利用する場合
グループ X	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
	TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256
	TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256
	TLS_DHE_RSA_WITH_AES_128_CCM
	TLS_ECDHE_ECDSA_WITH_AES_128_CCM
	TLS_DHE_RSA_WITH_AES_128_CCM_8
	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
	TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384
	TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384
	TLS_DHE_RSA_WITH_AES_256_CCM
	TLS_ECDHE_ECDSA_WITH_AES_256_CCM
	TLS_DHE_RSA_WITH_AES_256_CCM_8
	TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8
	TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
グループ Y	TLS_RSA_WITH_AES_128_GCM_SHA256
	TLS_DH_RSA_WITH_AES_128_GCM_SHA256
	TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
	TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256
	TLS_RSA_WITH_CAMELLIA_128_GCM_SHA256
	TLS_DH_RSA_WITH_CAMELLIA_128_GCM_SHA256
	TLS_ECDH_ECDSA_WITH_CAMELLIA_128_GCM_SHA256
	TLS_ECDH_RSA_WITH_CAMELLIA_128_GCM_SHA256
	TLS_RSA_WITH_AES_128_CCM
	TLS_RSA_WITH_AES_128_CCM_8
	TLS_RSA_WITH_AES_256_GCM_SHA384

グループ Y（続）	TLS_DH_RSA_WITH_AES_256_GCM_SHA384
	TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
	TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
	TLS_RSA_WITH_CAMELLIA_256_GCM_SHA384
	TLS_DH_RSA_WITH_CAMELLIA_256_GCM_SHA384
	TLS_ECDH_ECDSA_WITH_CAMELLIA_256_GCM_SHA384
	TLS_ECDH_RSA_WITH_CAMELLIA_256_GCM_SHA384
	TLS_RSA_WITH_AES_256_CCM
	TLS_RSA_WITH_AES_256_CCM_8
グループ Z	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256
	TLS_DHE_RSA_WITH_AES_128_CBC_SHA
	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA
	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256
	TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256
	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA
	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA
	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_CBC_SHA384
	TLS_ECDHE_RSA_WITH_CAMELLIA_256_CBC_SHA384
	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
	TLS_RSA_WITH_AES_128_CBC_SHA256
	TLS_DH_RSA_WITH_AES_128_CBC_SHA256
	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256
	TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA256
	TLS_ECDH_ECDSA_WITH_CAMELLIA_128_CBC_SHA256
	TLS_ECDH_RSA_WITH_CAMELLIA_128_CBC_SHA256

グループ Z（続）	TLS_RSA_WITH_AES_128_CBC_SHA
	TLS_DH_RSA_WITH_AES_128_CBC_SHA
	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
	TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA
	TLS_RSA_WITH_AES_256_CBC_SHA256
	TLS_DH_RSA_WITH_AES_256_CBC_SHA256
	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256
	TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA256
	TLS_ECDH_ECDSA_WITH_CAMELLIA_256_CBC_SHA384
	TLS_ECDH_RSA_WITH_CAMELLIA_256_CBC_SHA384
	TLS_RSA_WITH_AES_256_CBC_SHA
	TLS_DH_RSA_WITH_AES_256_CBC_SHA
	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
	TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA

	TLS1.3 を利用する場合
グループ A	TLS_AES_128_GCM_SHA256
	TLS_AES_128_CCM_SHA256
	TLS_AES_128_CCM_8_SHA256
	TLS_AES_256_GCM_SHA384
	TLS_CHACHA20_POLY1305_SHA256
鍵交換	DHE
	ECDHE
署名	ECDSA
	RSA-PSS
	RSASSA-PKCS1-v1_5

7. TLS を安全に使うために考慮すべきこと

TLS をより安全に使うために、以下の項目についても考慮すべきである。

7.1 最新のセキュリティパッチの適用

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べきである。

7.2 サーバ証明書の作成・管理について注意すべきこと

7.2.1 サーバ証明書での脆弱な鍵ペアの使用の回避

擬似乱数生成機能に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。

7.2.2 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、(鍵ペアにおける) 秘密鍵の生成・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リスクがある。

また、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ことになり、以前の秘密鍵が漏えいした場合に最新の暗号化通信も復号でき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、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・更新することが望ましい。よって、本ガイドラインではサーバ証明書の遵守項目として位置付けている。

7.2.3 サーバ証明書の更新忘れ防止に対する対策例

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[18]。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。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、アルゴリズムアジリティ（暗号アルゴリズムの危殆化に対して、別の安全な暗号アルゴリズムに移行するための対策に要する容易性）の低下などが危惧されるようになる。特に、2 年など比較的長い期間有効の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、通常は有効期限の 3 ヶ月程度前からサーバ証明書の発行会社から有効期限切れに関するリマインドメールなどが送られてくるようになるため、リマインドメールに対して適切に対応する体制（責任者・担当者を明確にする、更新手順を決めるなど）を明確化しておくべきである。また、特段の制約が存在しない限り、1 年程度の有効期間を持つサーバ証明書を選択し、リマインダメールに依存することなく、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。

また、その他の手段として、証明書の自動発行・更新プロトコルである ACME を実装した発行サービスを利用し、自動で証明書を更新する設定を組み込むことも対策として挙げられる。これにより、更新作業コストの削減が見込める。ただし、自動更新機能を利用するにあたり、導入先のサーバ環境に適合するクライアントツールの選定など初期設定が必要であることを注意する必要がある。なお、ACME に関する詳細はコラム②を参照されたい。

[18] 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。

【コラム②】 サーバ証明書の自動発行・更新プロトコル

証明書の発行・更新等の管理を自動化するプロトコルの普及が進んでいる。これは ACME（Automated Certification Management Environment）プロトコルと呼ばれ、2015 年の Internet-Draft 段階から Let's Encrypt プロジェクトでの実装が進むことによって普及が進んだもので、2019 年 3 月に RFC 8555 として策定された。

ACME を利用した実装の多くは、証明書を導入するサーバと証明書を発行する認証局との間のやり取りをサポートしたものであり、両者が ACME プロトコルに対応しており、かつ発行する証明書が DV 証明書である時に有用である。これは、RFC 8555 において認証局が行うべき身元確認スキームとして「ドメイン確認」にフォーカスしているためである.

このプロトコルを用いることで証明書発行から更新も含めて自動化可能となるが、以下の点は少なくとも留意しておくべきであろう。

- 実装に瑕疵や脆弱性があれば、証明書の発行・更新が適切に処理されずに停止する可能性がある
- 暗号スィートの設定変更などは ACME の対象外であり、サーバ管理者にて別途対応する必要がある

7.2.4 サーバで使用する鍵ペアの適切な管理

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（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、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、文書化しておくべきである。

- 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）

CRYPTREC では、「暗号鍵管理システム設計指針（基本編）」という暗号鍵管理に関するガイドラインを発行[19]している。上記の検討にあたっては、特に同ガイドラインの 2 章（暗号鍵管理の在り方）、5 章（暗号アルゴリズム運用のための暗号鍵管理オペレーション対策）、6 章（暗号アルゴリズムの選択）、7 章（暗号アルゴリズム運用に必要な鍵情報の管理）も参考にされたい。

7.2.5 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 32 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が行われる EV 証明書が利用者にとって本来一番安心できるサーバ証明書である。実際、欧州では、信頼されるサービス（Trusted Service）であることを示すために、Qualified Website Authentication Certificate（QWAC）という EV と同等の実在確認と

[19] https://www.ipa.go.jp/security/vuln/ckms.html
https://www.cryptrec.go.jp/report/cryptrec-gl-3002-1.0.pdf

組織認証を行う証明書の利用が始まっている。厳密な実在確認を行うため取得コストはかかるものの、以前は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」表示による視認性の高さが EV 証明書の大きなアドバンテージであった。しかし、現在の主要ブラウザではグリーンバー表示を廃止する傾向にあり、OV 証明書や DV 証明書との視認性の差は以前よりは低下している。もっとも、アドレスバーにある「錠前」マークをクリックすると表示される「証明書の簡易ビューア」では、EV 証明書は OV 証明書や DV 証明書よりも多くの情報が表示されたり緑色で表示されたりするなど、依然として視認性の差は存在している。

図 13　証明書の簡易ビューアの表示例（左：EV 証明書の場合、右：EV 証明書でない場合）

OV 証明書は、EV 証明書と同様、不特定多数の利用者がアクセスする一般的な Web サーバ用途で用いられることが多く、運営サイトの実在性の確認も行われる。

表 32 に示した通り、EV 証明書ほどは厳格な実在確認は行われないため、EV 証明書と比較すると取得コストが安価である。その一方、ブラウザにおける視認性はやや低いうえ、ブラウザによる取り扱い方法の差が大きく、EV 証明書と同様に主体者名（運営組織名などの発行先情報）が表示されるものもあれば、DV 証明書との識別が難しいものもある。

DV 証明書は、唯一個人でも取得することができる証明書である。しかも、Let's Encrypt プロジェクト[20]が DV 証明書を無料発行するなど、非常に入手コストが安い。一方で、運営サイトの実在性の確認は行われないため、フィッシングサイトなどに利用されるサーバ証明書のほぼすべてが DV 証明書を使っている。したがって、企業が DV 証明書を使うのは、不特定多数の利用者がアクセスする一般的な Web サーバよりも、特定少数の利用者向けの Web サーバや実証実験などのテストサーバなどにとどめることが望ましい。

そこで、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 サーバ、とりわけ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 証明書の利用をまず検討すべきである。それ以外の利用ケースにおい

[20] https://letsencrypt.org/

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。例えば、ブラウザ以外のスマートフォン用アプリケーションでの利用など、アドレスバー視認性の高さが求められないようなケースでは、EV 証明書のメリットが十分に生かせないので、OV 証明書や DV 証明書も有力な選択肢となる。

表 32　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く、コラム②にある ACME プロトコルとの親和性が高い、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、ドメイン名以外の主体者名（運営組織名などの発行先情報）は基本的に表示されない。このため、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、異なる運営組織に入手・利用されやすい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（コラム④も参照のこと）。
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、ブラウザによっては証明書の簡易ビューアに主体者名（運営組織名などの発行先情報）が表示される。しかし、①証明書ビューアで確認しない限り、ブラウザのアドレス表示部分では OV 証明書と DV 証明書を識別できない、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、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」表示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である。また、グリーンバーには運営組織も表示される実装もあるため、そのような実装をしたブラウザではドメイン名との関係が一目でわかる ● 多くのブラウザでは、証明書の簡易ビューアで主体者名（運営組織名などの発行先情報）が表示される。その際、ブラウザによっては緑色で表示され、EV 証明書であることの識別が容易である。

【コラム③】サーバ証明書解析からフィッシングサイトを見つけ出せるか？

コラム④にもあるように、フィッシングサイトの HTTPS 化が急速に進んでいる。その際、必ず必要となるのが「（証明書の仕組みとしては正当なフィッシングサイト用の）サーバ証明書」である。

ここで注目すべきは、フィッシングサイトは、その性質上、短期間の使い捨てで変化していくため、フィッシングサイト用のサーバ証明書の有効期間は長くする必要がない。また、金銭目的であることが多いことを考えればコストメリットを重視し不必要なコストはかけたくないはずである。実際、F5 Labs が公表した記事[21]によれば、多くのフィッシングサイトでは cPanel と Let's Encrypt が発行する「**無料のサーバ証明書を使い**」、フィッシングサイトの 36%は「**90 日しか活動しておらず**」、サーバ証明書の「**自動発行プロセスを活用している**」と強く推定されるとのことである。

このようなフィッシングサイト構築者の行動パターンを逆手に取り、実際のフィッシングサイトで使われたサーバ証明書をもとに、今後フィッシングサイトに使われる可能性があるサーバ証明書を検知しよう、という研究発表[22]が CSS2019 であった。

本研究では、OpenPhish が 2018 年 10 月～2019 年 1 月に収集したデータセットの中から実際のフィッシングサイトで使用されたサーバ証明書 1,634 個を抽出し、先生役としてのフィッシングデータとしている。ちなみに、1,634 個のうち、cPanel と Let's Encrypt が発行するものは合計で 1,566 個、95.8%にも達していることが示されている。

これらフィッシングサイトで使われたサーバ証明書のコモンネームの類似性を分析して作成した検知用テンプレートから未知のフィッシングサイトで使われるサーバ証明書を事前に検出できる可能性があることを明らかにした。実際、69 個の検知用テンプレートを作成した後、Censys が保有する cPanel と Let's Encrypt が同時期に発行した約 3,800 万個のサーバ証明書について探索したところ、1,650 個の証明書が検出された。特にある 1 つのテンプレートは 900 を超えるサーバ証明書のコモンネームと合致しており、それらをより詳細に調べたところ、93%の証明書が Let's Encrypt から発行されていた。加えて、フィッシングサイト作成サービスの存在を発見するに至った。

フィッシングサイトの検出手法といえば、今までドメイン名や URL、メール等の分析結果を基にしたものが一般的であったが、フィッシングサイトの HTTPS 化に伴い必然的にサーバ証明書が必要となることから「サーバ証明書解析に基づく検出手法」が有効な対策の一つになる可能性があり、今後の研究が期待されるところである。

一方、サーバ運営者は、サーバ証明書の選択にあたって、無料のサーバ証明書がフィッシングサイトにも多く使われていることに十分注意を払うべきであり、場合によっては無料のサーバ証明書は使わないといった配慮が必要となる。

---

[21] F5 Labs, "2019 PHISHING AND FRAUD REPORT," https://www.f5.com/content/dam/f5-labs-v2/article/pdfs/F5Labs_2019_Phishing_and_Fraud_Report.pdf

[22] 櫻井、渡邉、奥田、秋山、森、「サーバ証明書解析によるフィッシングサイトの発見手法」、CSS2019

7.2.6 DNS の CAA（Certification Authority Authorization）設定による証明書不正発行の防止

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、CAA）は 2013 年に RFC 6844 として定められ、2019 年 11 月に RFC 8659 として改訂された。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[23]。なお、SSL Pulse[24]によると、CAA の普及率は 2020 年 5 月時点で 7.3%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 [25]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。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[26]で確認する。
② DNS 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、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。

---

[23] CA/ブラウザフォーラムによる「Baseline Requirement」においては、現時点では認証局は RFC 6844、RFC 8659 のいずれかを採用すればよいとされている。
[24] https://www.ssllabs.com/ssl-pulse/
[25] RFC 6698 DNS-based Authentication of Named Entity (DANE): Transport Layer Security (TLS) Protocol: TLSA
[26] 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

7.2.7 複数サーバに同一のサーバ証明書（ワイルドカード証明書／マルチドメイン証明書）を利用する場合の注意点

サーバ証明書では 1 枚の証明書に複数のドメイン名（FQDN）あるいはワイルドカードを用いたドメイン名を記載することができ、前者はマルチドメイン証明書、後者はワイルドカード証明書と呼ばれている。

これらのサーバ証明書は、各サーバに同じ鍵ペアと証明書を共有することができるため、負荷分散や冗長化による可用性向上などを目的とした複数サーバの構成管理などに有用である。

サーバ管理者としては、鍵ペアとそれにひもづく複数のサーバとを両方管理するケースと、（鍵ペアはホスティング業者などで別途管理されており）サーバだけを管理するケースが考えられ、本節では前者にフォーカスする。

サーバ管理者は、複数のサーバで同一の鍵ペア・証明書を共有することになるため、以下の点に留意する必要がある[27]。

- 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すべてのサーバについて漏れなく鍵ペア・証明書を入れ替えること
- サーバ証明書の入替に伴って暗号スイートの設定変更などを行う場合は、対象とするすべてのサーバについて漏れなく適用すること

これらの運用を円滑に行うために、サーバ管理者は、サーバ証明書の作業対象となるサーバに漏れが発生しないよう、各サーバに対応する鍵ペアや証明書、暗号スイートなどの情報を一覧管理し、管理方法についても規定・文書化することが推奨される。

7.2.8 プライベート認証局の利用の注意点

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、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。

例外的に、社内向けシステムやクローズドシステムなど、利用者や利用端末が限定される環境である場合には、プライベート認証局を立ち上げ、その管理下でシステムが運用されることがある。このような場合、プライベート認証局が発行したサーバ証明書を使ったサーバを“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プライベー

[27] 特に、異なるポリシーによって管理される複数のドメイン名や組織を同一証明書に混在させる場合は、ポリシー間の整合や各組織との調整等も含めて留意する必要がある。

ト認証局のルート 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。

ただし、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プライベート認証局のルート 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、ブラウザ（クライアント）にインストールした当該ルート CA 証明書を手動で削除する必要がある。もし削除を怠ると、ブラウザ（クライアント）側では不正なサーバ証明書であっても正しいサーバ証明書と誤認するリスクが増大する。

このため、例外的にプライベート認証局のルート CA 証明書を手動インストールする必要があるシステムの場合には、当該ルート 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実施・管理する主体によって、全ての対象ブラウザ（クライアント）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､具体的な対策を実施する体制を整えるべきである｡例えば、企業が自身の管理する端末に対してシステム管理部門が運用しているプライベート認証局のルート CA 証明書を搭載し、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（インストール及び削除）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たシステムにおいて、当該ルート CA 証明書の安全性に問題が生じた場合には、速やかにシステム管理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じなければならない。

7.3 委託先のサーバ（PaaS／SaaS）を利用する場合の注意点

本節では、委託先のサーバ（PaaS／SaaS）を利用する場合の注意点について記載する。

A) 提供されるレイヤにより、利用者がコントロール可能な範囲が異なることに留意

クラウド等のサービスにコンピューティングリソースを配置する場合、クラウドサービスプロバイダ（CSP）によって提供されるレイヤによって、利用者による管理の要否が異なる場合がある。CSP が管理すべきレイヤが多い場合、運用上の負荷やコストを軽減することができるが、統制上のオーナーシップをより利用者が保有することが重要な場合もあるため、サービスの選択において、組織の統制モデル、実現可能性、運用やコストへの負担を踏まえたうえで選択することが望まれる。

- **管理レイヤの際による違い**

  SaaS に近いサービスであれば、経路暗号化がサービスに組み込み済みのもの（利用者がコ

ントロールせず、CSP がコントロール）が存在する。IaaS や PaaS レイヤにおいても、利用者が自らサードパーティの証明書を導入するケースもあれば、CSP が証明書サービスを提供しており、利用可能である場合もある。こうしたサービスは、更新管理の自動化や通知の実装は可能なケースもある一方、利用可能な証明書の制限（EV 等の利用制限）も存在する。特にサービスを利用する場合、利用を禁止とすべき技術（危殆化したバージョン等）等をどのように管理可能かを留意すべきである。

B) CSP が提供する証明書の制限

CSP が証明書管理サービスを提供している場合、適用可能なバージョンや対象はサービスの機能に依存する（例えば、CSP のプラットフォーム上のサーバインスタンスにのみ証明書が適用可能等）。コストや管理面によるメリットを理解したうえで、適用範囲を決定する必要がある。

- **主な考慮事項**

  CSP が提供する証明書に対する考慮事項は主に以下の項目があげられる。
  - 適用対象（証明書を導入可能なサーバインスタンスの特定や、連携可能なサービス）
  - コスト（有償、無償）
  - 証明書の種類（EV、OV、DV）
  - 管理範囲（更新処理の自動化等）

C) アーキテクチャによる実装の違いに留意

IaaS や PaaS などを利用する場合、ネットワークアーキテクチャを踏まえて、利用者が設計するケースがある。その場合、ネットワーク上のどの範囲までを経路暗号化の対象とすべきかを考慮したうえで設計する必要がある。一般にインターネットなどの公衆網を暗号化通信の適用範囲とすることは多くのケースで推奨される。その一方、CSP が統制するネットワーク範囲をどの程度、暗号化通信の対象とすべきかは、利用者の設計に委ねられる。

CSP の提供するサービスにもよるが、処理を行うサーバで終端処理を行うパターン、ロードバランサーやゲートウェイサービスで終端処理を行うパターン、CDN（Content Delivery Network）のエッジで終端処理を行うパターンがある。

i) 処理を行うサーバで終端処理を行う

Web サーバ（Web アプリケーションサーバ）で終端処理を行う。

利点： End to End の暗号化を実現する

欠点： コストおよび管理の複雑性、サーバのリソース処理が増加する。処理を行うサーバがスケールアウト（負荷に対応するため、弾力的に増加をする設計）する場合、終端処理を行うサーバの台数分の証明書管理が必要となる場合がある。

ii) ロードバランサーやゲートウェイサービスで終端処理を行う

実際に処理を行うサーバのフロントに配置されたロードバランサー等の疎結合層において終端処理を行う。

利点： 処理のオフロードおよび管理の集中化により、運用やコストの負荷が軽減される

欠点： プロパイダによっては、CSP 内の通信網は平文の通信となる（ただし、CSP 内における通信に対する盗聴リスクを他のコントロールで評価することは可能である）

iii) CDN（Content Delivery Network）のエッジで終端処理を行う

利用者に近いエッジで終端処理を行うことでより暗号化および復号処理の分散や処理の高速化を実現する。

利点： 処理のオフロードおよび管理の集中化により、運用やコストの負荷が軽減される

欠点： プロパイダによっては、エッジから CSP および CSP 内の通信網は平文の通信となる（ただし、CSP 内における通信に対する盗聴リスクを他のコントロールで評価することは可能である）

7.4 さらに安全性を高めるために

7.4.1 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、最初の HTTP 通信は通信上の攻撃者によって改ざんされる可能性がある。リダイレクトして悪意のあるサイトに誘導し情報を抜き取ったり、SSL strip というツールを用いて HTTPS へのリダイレクトを防ぎつつあたかも本物のサイトのように振る舞い情報を盗んだりといった手法の報告が Moxie Marlinspike によってなされた。

この攻撃のように、サーバが HTTPS に対応していてもユーザが http://で始まる URL でサイトにアクセスすると、通信は平文となり、メッセージを改ざんされうる。Web サイトが暗号化なしの HTTP をサポートしている限り、攻撃者がメールで誘導するなど http://のリンクを攻撃対象者に開かせることで、平文通信を行わせることが出来る。

これを防ぐため RFC 6797 で規定されている HTTP Strict Transport Security（HSTS）では、サーバ側から以降 HTTP でのアクセスは行わず HTTPS でアクセスするようブラウザに指示できる。具体的には、HTTPS 応答に以下のような HTTP レスポンスヘッダを含めることでブラウザに指示を送る。

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

この HTTP レスポンスヘッダを受け取った HSTS 対応のブラウザは、有効期間中は当該ドメインへのアクセスは HTTP ではなく全て HTTPS で通信するように自動設定しておく。includeSubDomain が指定されていた場合、サブドメインへのアクセスも同様である。

これにより、以前接続したときに HSTS が有効になっているドメインであれば、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。

7.4.2 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（Certificate Revocation List）とOCSP（Online Certificate Status Protocol）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。

i) 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 TLS 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 TLS 通信をさせることができる

ii) 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。この拡張には以下の特長があり、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 TLS ハンドシェイク中に OCSP レスポンスもできる。

- 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

7.4.3 Public Key Pinning のサポート終了について

HPKP（HTTP Public Key Pinning）は、不正に発行されたサーバ証明書による通信を検知するための仕組みとして 2011 年に Google により提案され[28]、2015 年に RFC 7469 として標準化された[29]。過去には、Chrome、Firefox、Opera など一部のウェブブラウザで HPKP をサポートしていたが、

- 仕組みが複雑で、運用や設定ミスが発生した場合、数か月単位で指定された FQDN での TLS 暗号通信ができなくなるケースが発生した
- HPKP を設定しているサーバの比率はかなり低く、今なお減少傾向にある

などの理由により、2019 年 1 月ごろ Chrome、Firefox は HPKP のサポートを終了した[30][31]。これに伴い、今後 HPKP が広く使われることはないと判断し、本ガイドラインでも HPKP の設定方法の記述は削除することとした。

[28] ImperialViolet: Public Key Pinning, May 2011, https://www.imperialviolet.org/2011/05/04/pinning.html
[29] RFC 7469 Public Key Pinning Extension for HTTP, Apr 2015, https://tools.ietf.org/html/rfc7469
[30] Can I use HPKP https://caniuse.com/#search=hpkp
[31] Chrome Platform Status: Remove HTTP-Based Public Key Pinning (removed) https://www.chromestatus.com/feature/5903385005916160

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

ブラウザは、少なくとも提供ベンダがメンテナンスしているバージョンのものを利用すべきである。

なお、本ガイドラインに記載の情報に関し、マイクロソフト、Google、及び Mozilla についての情報は 2020 年 3 月時点で各社から直接提供していただいたものである。また、Apple についての情報は Apple の公式ホームページから入手したものである。

マイクロソフト

- Internet Explorer 11　　Windows のサポート期限に準じてメンテナンスされる
- Edge　　Windows のサポート期限に準じてメンテナンスされる
  - Windows のサポート期限[32]
    - Windows 8/8.1　　サポート終了日：2023 年 1 月 10 日
    - Windows10　　半年ごとのメジャーアップデートを適用

Google

- Chrome　　最新バージョン（2020 年 5 月時点）は以下のシステム要件を満たす場合に利用できる
  - Windows
    - Windows 7*, Windows 8, Windows 8.1, Windows 10 or later
    - An Intel Pentium 4 processor or later that's SSE2 capable
    - Note: Servers require Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, or Windows Server 2016
      *) Windows 7 のサポートは 2021 年 7 月 15 日まで
  - Mac
    - OS X Yosemite 10.10 or later
  - Linux
    - 64-bit Ubuntu 14.04+, Debian 8+, openSUSE 13.3+, or Fedora Linux 24+
    - An Intel Pentium 4 processor or later that's SSE2 capable
  - Android
    - Android 4.4（KitKat）以上
  - iPhone & iPad
    - iOS 10 or later

[32] 企業向けに特別に提供されている拡張セキュリティ更新プログラム（ESU; Extended Security Updates）を利用している Windows 7 のサポート終了日：2023 年 1 月 10 日

Mozilla

- Firefox　最新バージョン（2020 年 5 月時点）は以下のシステム要件を満たす場合に利用できる
  - Windows
    - Windows 7, Windows 8, Windows 8.1, Windows 10
    - Pentium 4 or newer processor that supports SSE2
    - 512MB of RAM / 2GB of RAM for the 64-bit version
    - 200MB of hard drive space
  - Mac
    - macOS 10.9 or later
    - Macintosh computer with an Intel x86 processor
    - 512 MB of RAM
    - 200 MB hard drive space
  - Linux
    - GTK+ 3.4 or higher
    - GLib 2.22 or higher
    - Pango 1.22 or higher
    - X.Org 1.0 or higher (1.7 or higher is recommended)
    - libstdc++ 4.6.1 or higher
  - Android
    - Android 5.0 以上
  - iPhone / iPad
    - iOS 10.3 or later

Apple[33]

- Safari　最新バージョン（2020 年 1 月時点）は以下のシステム要件を満たす場合に利用できる
  - Mac
    - macOS Catalina（macOS 10.15）
    - macOS Mojave（macOS 10.14.5）
    - macOS High Sierra（macOS 10.13.6）
  - iOS
    - iOS 13

[33] https://developer.apple.com/documentation/safari_release_notes/safari_13_release_notes

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

本ガイドラインに記載の情報に関し、マイクロソフト、Google、及び Mozilla についての情報は 2020 年 3 月時点で各社から直接提供していただいたものである。また、Apple についての情報は Apple の公式ホームページから入手したものである[34]。

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、プロトコルバージョンや暗号スイート、証明書処理などに関する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（デスクトップ）版 Web ブラウザ
  - Google Chrome
  - Safari

- スマートフォンに含まれる Web ブラウザ
  - Google Chrome
  - Mozilla Firefox

設定項目を標準機能で提供しているブラウザ

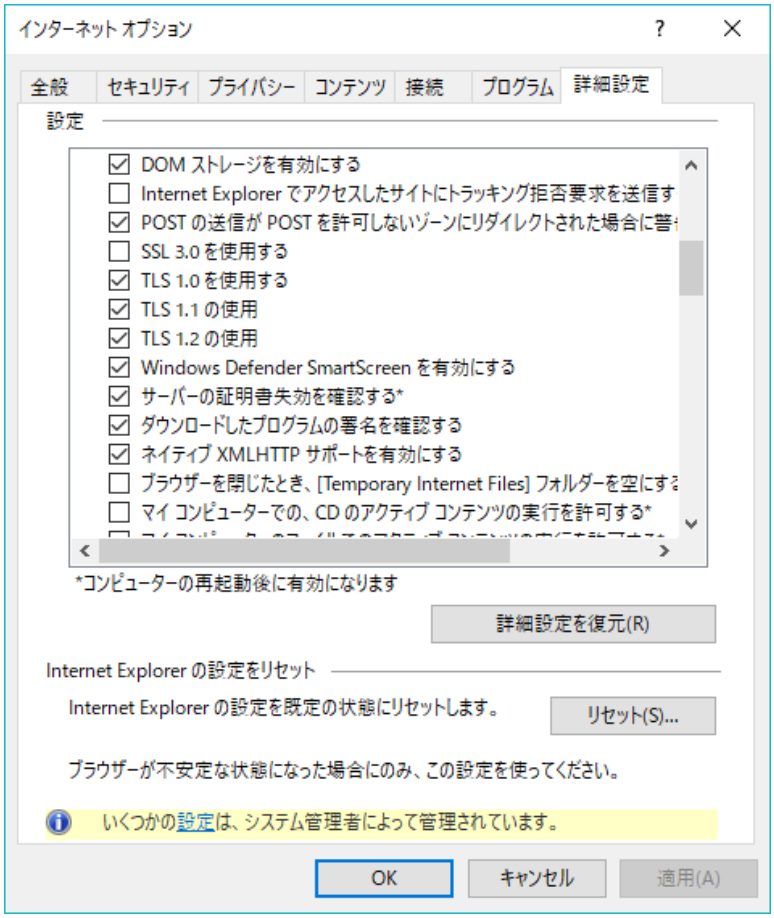
以下のブラウザは、設定変更オプションが提供されている。ただし、特別な指示がない限り、デフォルト設定を変更すべきではない。

- 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”

[34] https://support.apple.com/ja-jp/guide/safari/welcome/mac

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。



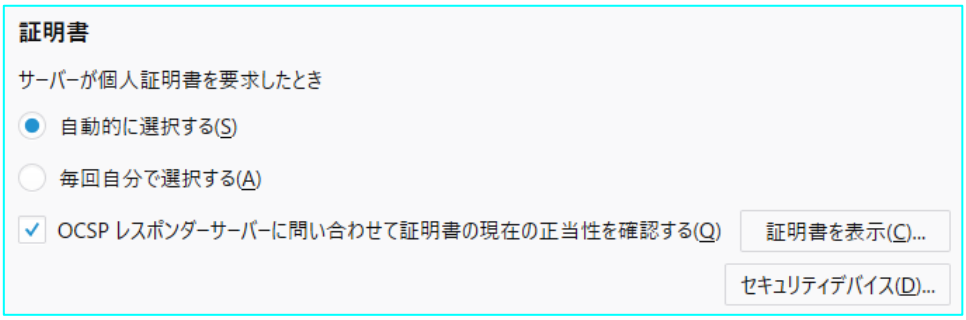
- Mozilla Firefox（デスクトップ版のみ）

  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。



設定項目の強制管理機能

以下のブラウザでは、条件を満たす場合に、ユーザが利用するブラウザの設定変更オプションを強制的に管理者が設定できる。

- Microsoft Internet Explorer／Microsoft Edge
  ローカル管理者権限により設定が可能

- Google Chrome
  エンタープライズ環境であれば、管理者は下記の資料に記載されている設定が可能
  https://services.google.com/fh/files/misc/chromebrowserenterprisesecurityconfigurationguide10.19.pdf

- Mozilla Firefox（デスクトップ版のみ）
  エンタープライズ用途においては、管理者は以下の資料に基づいて Policy Engine を経由した設定が可能
  https://support.mozilla.org/products/firefox-enterprise/policies-customization-enterprise

8.3 ブラウザ利用時の注意点

2010 年代中頃以前のブラウザは、SSL/TLS を利用しない通信、すなわち URL が「http://」から始まるサイトへのアクセスが基本形になっており、SSL/TLS を利用する「https://」から始まるサイトにアクセスするとブラウザ上部のアドレスバーなどに「錠前」マークや「保護された通信」といった表示がされるようになっていた。この表示方法は、ブラウザの開発ベンダやバージョンの違いといったものに依存せず、すべてに共通したものであった。

しかし、この数年で常時 HTTPS 化の対応を行ったサイトが急増したことを受け、主要ブラウザのなかには、SSL/TLS を利用する URL が「https://」から始まるサイトへのアクセスのほうを基本形とするものが出てきている。このようなブラウザでは、以前のブラウザとは反対に、「http://」から始まるサイトにアクセスすると「保護されていない通信」や「安全ではありません」といった表示、あるいは「錠前マークに“／”が付いている」表示などの警告表示が出るようになった。

また、サーバ証明書の表示に関しても最近変化があり、アドレスバーが緑色になったり企業名や団体名を表示したりといった「EV 証明書」特有の表示を取りやめたブラウザが出てきている。そのような対応をした理由は、「EV 証明書」特有の表示を行っても行わなくても利用者の行動に変化はなく、期待した効果が得られていないという Chrome のセキュリティ UX チームの調査結果に基づく判断[35]としている。

EV 証明書特有の表示を取りやめたブラウザでは、どの種類のサーバ証明書を利用していたとしてもアドレスバーでの表示に変わりがなく、錠前マークをクリックして証明書の内容を表示するまで違いが判らない状態になった（7.2.5 節も参照されたい）。

このように、ブラウザのアドレスバーでの表示方法がブラウザの開発ベンダやバージョンによって異なる状況になってきたことを踏まえ、本ガイドラインではアドレスバーでの表示方法についての説明を取りやめる。

代わりに、主要ブラウザについてアドレスバーの表示方法の違いや変更について、フィッシング対策協議会が適宜調査を実施し公表しているので、フィッシング対策協議会の情報を参照されたい。URL は以下の通り。

- 各ブラウザによる SSL/TLS サーバ証明書の表示の違い
  （フィッシング対策協議会 証明書普及促進 WG 成果物）
  https://member.antiphishing.jp/about_ap/wg.html#certificate

【参考】フィッシング対策協議会とは：

フィッシング詐欺に関する事例情報、技術情報の収集及び提供を中心に行うことで、日本国内におけるフィッシング詐欺被害の抑制を目的として、2005 年 4 月に発足した団体である。JPCERT/CC を事務局に、2020 年 1 月時点で 104 の会社や関連団体等が参加している。

https://www.antiphishing.jp/

[35] Google, EV UI Moving to Page Info,
https://chromium.googlesource.com/chromium/src/+/HEAD/docs/security/ev-to-page-info.md

【コラム④】 TLS ではフィッシングが防げない？－TLS で守られる限界を知ろう

「TLS は、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）の認証を行うプロトコルであり、オンラインショッピングやネットバンキング等のサービスには不可欠である。」と説明されている。本ガイドラインでも、利用例として「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広範囲（不特定多数）に提供する場合」を挙げている。

ところが、一方で残念なデータもある。**「半数以上のフィッシングサイトが HTTPS を使っている」**という記事[36]が公開されている。2017 年以降のフィッシングサイトでの HTTPS 利用率の伸びは驚異的で、その後も増え続け、2016 年 3Q にはわずか 2.8%に過ぎなかったのに 2019 年 1Q に 58%、2019 年 4Q に 70%超にもなっている。フィッシングサイトがこのような対応を進めた理由は明らかで、このころから世界中で常時 HTTPS 化が始まるとともにブラウザのアドレスバー表示が変わったからに他ならない。HTTP のままでは「保護されていない通信」とか「安全ではない」と表示されるようになったので、例え本物そっくりのサイト画面を作ったとしてもすぐにフィッシングサイトだと見破られてしまうためである。

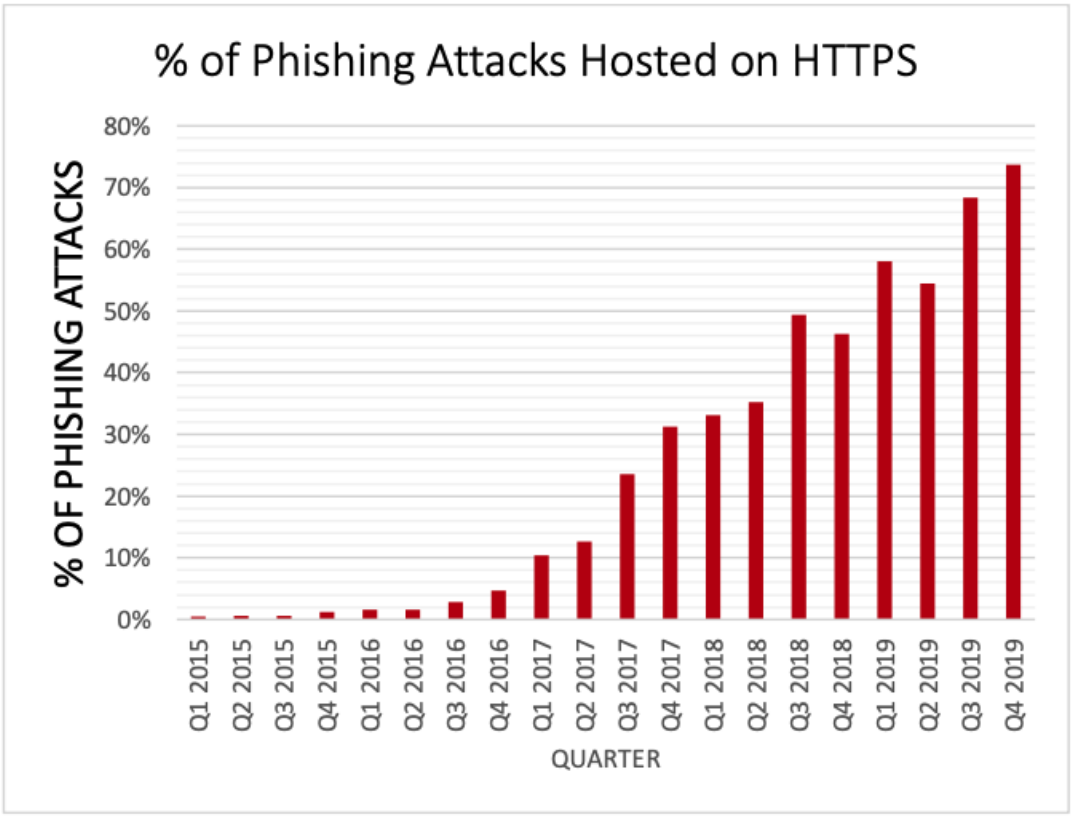


図 14 フィッシングサイトの HTTPS 利用率（[出典] The PhishLabs Blog[37]）

しかし、なぜこれほど HTTPS を使ったフィッシングサイトが作れるのかと疑問をもつ読者がいるかもしれない。そのポイントは「証明書によるサーバ認証」にかかるコストメリッ

[36] https://info.phishlabs.com/blog/more-than-half-of-phishing-sites-use-https
[37] https://info.phishlabs.com/blog/top-phishing-trends-2019

トが大きく変化したことにある。

一昔前にフィッシングサイトが HTTPS 化をしていなかった理由は、鍵マークを確認しない人が多く HTTP のままでもフィッシングサイトに誘導される可能性がそれなりにあったためである。つまり、コストをかけてサーバ証明書を購入してまでフィッシングサイトを HTTPS 化する必要性がなく、HTTPS 化することのコストメリットがほとんどなかったからにすぎない。

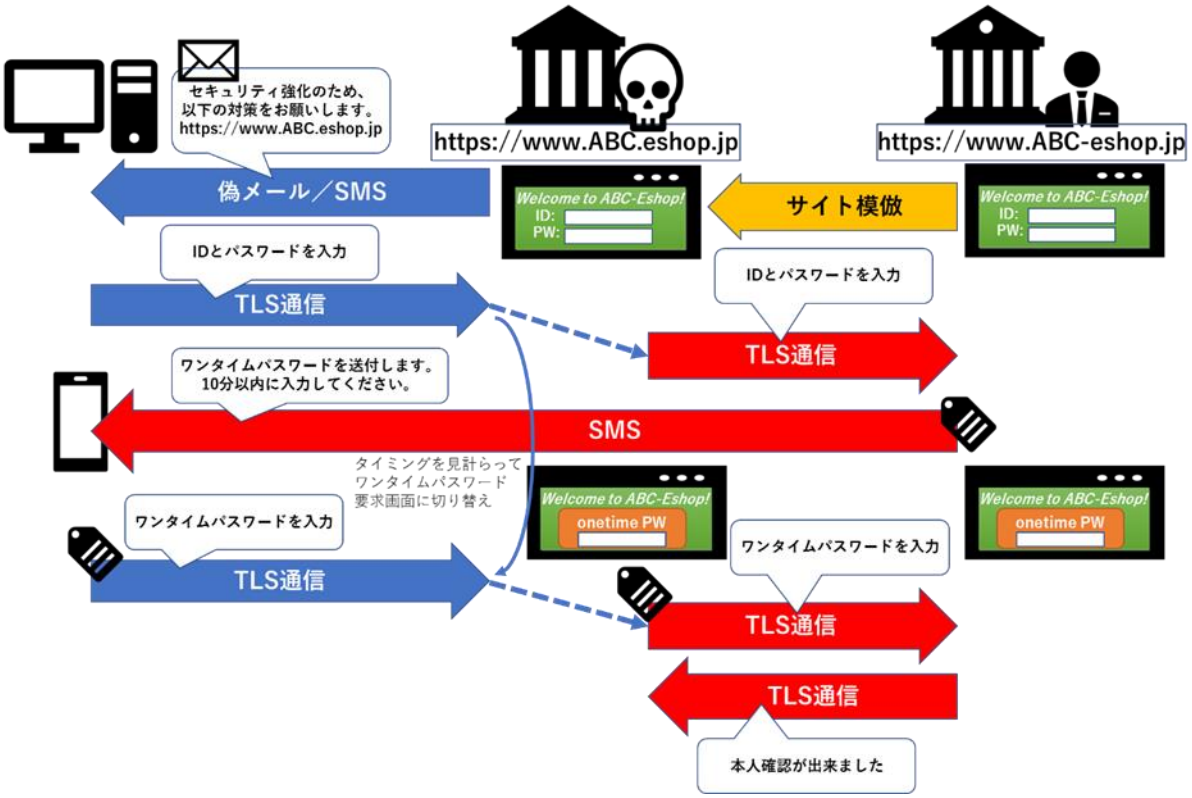


図 15　2 段階認証を突破するフィッシングの手法例

ところが、今やその状況は逆転した。HTTP のままではブラウザの警告表示によってフィッシングサイトへの誘導が難しくなった一方、常時 HTTPS 化の動きの中で Let’s Encrypt のような無料（あるいは相当安価なコスト）でサーバ証明書を準備できるようになった。しかも、心理的に「HTTPS なら安心」という警戒感が薄まる効果も期待できる。このため、攻撃者から見ると「フィッシングサイトを HTTPS 化することのコストメリットが一気に高まった」のである。

実際、日本サイバー犯罪対策センターから注意喚起[38]が出ているように、最近のフィッシングでは偽メールだけでなく SMS（ショートメッセージサービス）も多用されており、その中には書かれる URL が「https://」から始まるなど手口が巧妙になっているものもある。例え

[38] 日本サイバー犯罪対策センター, https://www.jc3.or.jp/topics/banking/phishing.html

ば、フィッシング対策協議会の「フィッシングに関するニュース[39]」の各種金融機関をかたるフィッシング情報、IPA 安心相談窓口が公表している「偽 SMS を使ったフィッシング[40]」の事例が該当する。

技術的には、フィッシングサイト運営者は“フィッシングサイト”の“ドメインに対するサーバ証明書（DV 証明書)”を使う。DV 証明書では、ドメインが実在するかどうかしか検証せず、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しないので、(たとえフィッシングサイトであったとしても）ドメインが実在さえすれば DV 証明書の発行は可能である。また、ブラウザは、DV 証明書が有効であり、かつその証明書に記載のドメインがありさえれば TLS 接続を自動的に行う。

この時、ブラウザはそのサイトがフィッシングサイトであるかどうかの判断はしないため、利用者がサーバ証明書の中身を自ら確認して、接続したい本物のサイトのサーバ証明書でなければ接続しない（接続を閉じる）という行為をしない限り、TLS 接続されたからといってフィッシングを防ぐ手段にはならない。具体的には、偽メールや SMS に書かれているフィッシングサイト誘導用のリンクをそのまま使うと**フィッシングサイトが終端となる TLS 通信路が出来上がる**結果となる（図 15 の青矢印)。そうなると、**たとえ SMS を使った 2 段階認証を採用していたとしても、ワンタイムパスワードすらフィッシングサイト運営者に筒抜けになり、2 段階認証を突破**されるリスクがある（図 15 の赤矢印)。

このように、フィッシング対策としては「錠前マーク」や「https://になっているか」の確認だけではもはや不十分であり、「事前に正しいウェブサイトの URL をブックマークに登録してブックマークからアクセスする」等、新しい対策を取ることが必要になっていることに留意されたい。特に SMS は、便利である反面、送信元が偽装され本物の企業から届いている SMS のスレッドに送信元が偽装された SMS が紛れこむ事例が報道[41]されており、十分な注意が必要である。

また、もし利用者が本物の企業から SMS でリンク先を周知されるようなサービスを普段から日常的に受けていれば、偽装された SMS に記載されたフィッシングサイト誘導用のリンク先でもいつもと同じように疑わずに接続してしまう可能性が非常に高いと考えられる。このことから、サイト運営者のほうも、SMS を使ってリンク先を利用者に周知するといった方法を安易に使うべきではないといえるだろう。

[39] フィッシング対策協議会, https://www.antiphishing.jp/news/alert/
[40] IPA 安心相談窓口だより、「宅配便業者をかたる偽ショートメッセージに引き続き注意！」, https://www.ipa.go.jp/security/anshin/mgdayori20200220.html
[41] ケータイ Watch, 「送信元を偽装する SMS――注意すべき点とキャリアの対策は？」, https://k-tai.watch.impress.co.jp/docs/review/1220916.html

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 TLS 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 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。

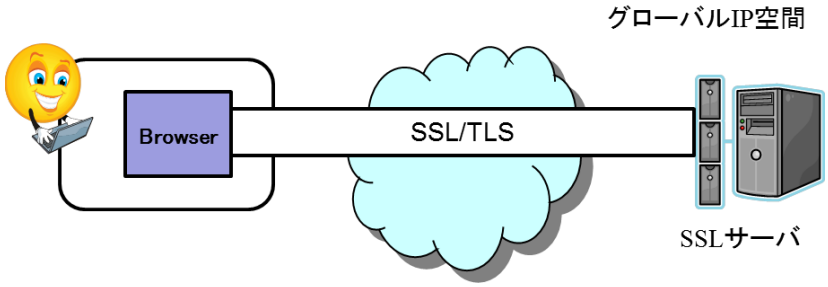
クライアントレス型は、ブラウザさえあればどの端末からでもアクセス可能であり、利便性に優れる一方、TLS との最大の差はグローバル IP をインターネットに公開しているか否か程度の違いといえる。結果として、最初のクライアント認証を TLS サーバが受け持つか、SSL-VPN 装置が受け持つか程度の差でしかなく、VPN というよりも、本質的には TLS と同じものとみるべきである。

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。

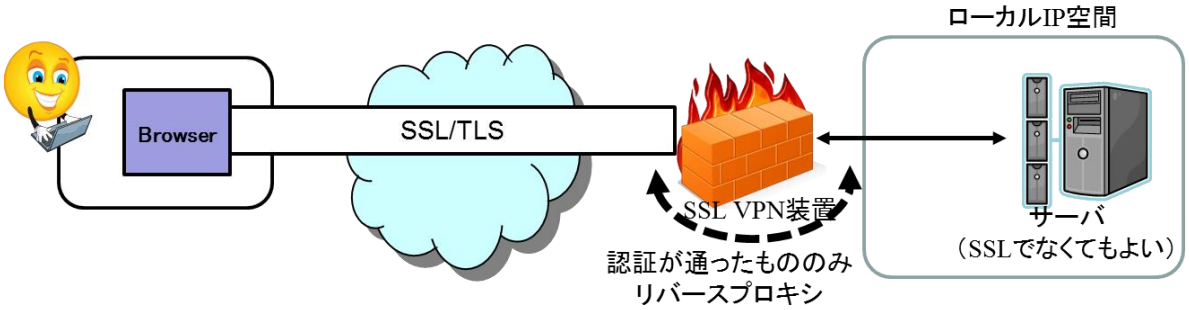
一方、クライアント型は上記の 2 つのタイプとは明らかに異なり、専用のクライアントソフトがインストールされた端末との間でのみアクセスする。つまり、誤って偽サーバに接続することがなく、また内部サーバにアクセスできる端末も厳格に制限できるため、端末に IPsec-VPN ソフトをインストールして構成するモバイル型の IPsec-VPN に近い形での運用形態となる。

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。

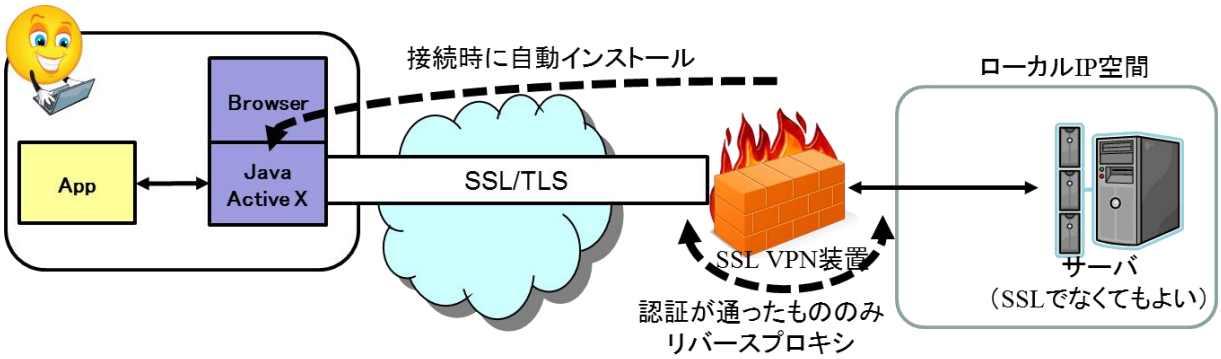
【参考：通常の TLS】



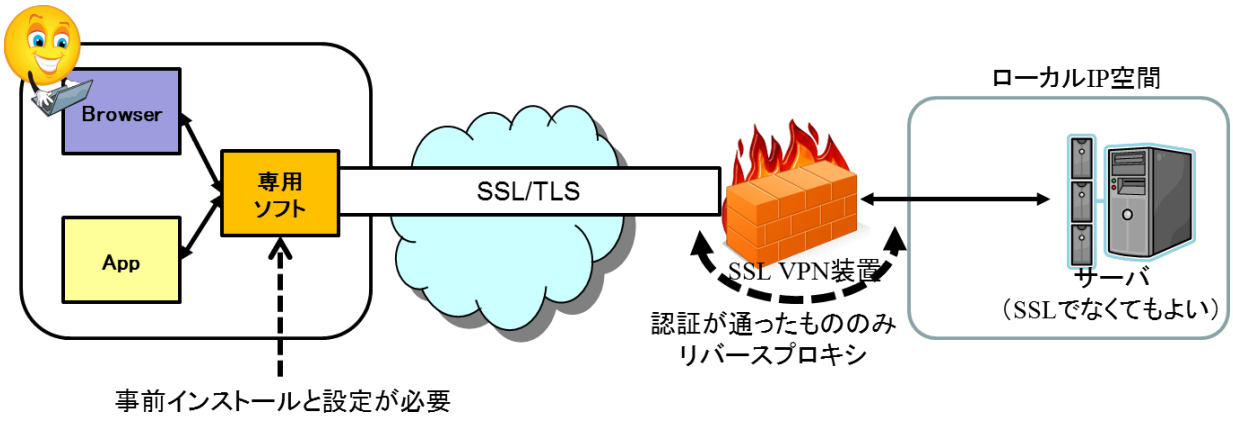
【クライアントレス型（ブラウザベース）】



【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）】



【コラム⑤】 ローカルネットワークでの HTTPS 通信問題

IoT の普及が進むとともに、ローカルネットワークに接続されたデバイス（プリンタ、テレビ等）に対して、同じローカルネットワーク上の（例えばスマホなどの）Web ブラウザからアクセスするユースケースが増えてきた。一方、8.3 節に示すように Web ブラウザのなかには HTTP 通信に警告が表示されるものも出てきているため、こうしたローカルネットワーク内の通信においても HTTPS 通信が必要となってくる。

しかし、パブリック認証局では CA/Browser Forum の規準により、パブリック DNS で一意性を確認できないサーバ名、いわゆる Internal Name を含む証明書を発行することが原則として禁じられており、前述のデバイスにおける HTTPS 通信は実現が難しい状況にある。

このような問題を解決するため、W3C（World Wide Web Consortium）の httpslocal CG（HTTPS in Local Network Community Group）において、ユースケースが整理され、これを踏まえてローカルネットワーク上の HTTPS 通信としてサーバ証明書を用いない通信方式、およびサーバ証明書を用いる通信方式の両面で検討が行われてきた。

ただ、いずれの方式もステークホルダに偏りがあるため、問題の整理や解決策の検討において俯瞰的な議論が十分にできてないのが現状であり、ベストプラクティスが出てくるまでにはまだまだ時間がかかると思われる。現時点では利用環境等に応じて個別に適切な技術を組み合わせて解決する必要がある一方で、安全性に配慮していく必要があることに留意されたい。

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-3001-3.0-checklists.pdf

[Excel 版] https://www.ipa.go.jp/security/ipg/documents/ipa-cryptrec-gl-3001-3.0-checklists.xlsx

A.1.　チェックリストの利用方法

【チェックリストの使い方】
本チェックリストは、選択した設定基準に対応した要求設定を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のチェックシートを下部の「タグ」から選択する
○ 当該チェックシートに記載のチェック項目全てについて参照章の記載を参考に設定内容を確認する
○ 「要求設定確認」は選択したチェックシートを利用してよいかの確認項目であり、「該当」にチェックが入る場合に限り、当該チェックシートを利用してよい
○ 「遵守項目」については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必要である
○ 「遵守項目」以外については記載内容を確認し、設定の実態に即して適切なほうのチェックボックスにチェックを入れる

＜チェックリストの例＞

【高セキュリティ型チェックリスト】

選択した設定基準に対応したチェックリストのチェックシートを用いる

要求設定が満たされていることを確認したら「済」にチェックを入れる

要求設定の詳細な内容が記載されている章番号

選択したチェックシートを利用してよいかの確認項目であり、「該当」にチェックが入る場合に限り利用してよい

「遵守項目」は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必要である

「遵守項目」以外については記載内容を確認し、設定の実態に即して適切なほうのチェックボックスにチェックを入れる

チェック項目		参照章		
①要求設定確認	①-1)【遵守項目】高セキュリティ型の設定基準であるか	3.1節	□該	
②プロトコルバージョン設定	[illegible]か	5.1節	□済	
	[illegible]か。ただし、TLS1.2を明確に利用をチェックする	5.1節	□済	□設定せず
	②-3)【遵守項目】SSL2.0からTLS1.1までを設定無効（利用不可）にしたか	5.1節	□済	
③サーバ証明書設定	③-1)【遵守項目】サーバの公開鍵情報（Subject Public Key Info）のSubject Public Key [illegible]gorithm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 ・ 楕円曲線暗	5.2節	□済	
	③-2)【遵守項[illegible]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以上の組合せで鍵長2048ビット以上 ・ ECDSAとSHA-256以上の組合せで鍵長256ビット（NIST P-256）以上	5.2節	□済	
	③-3)【遵守項目】サーバ証明書の発行・更新を行う際に、自ら公開鍵と秘密鍵の鍵ペアを生成する場合には、新たな公開鍵と秘密鍵の鍵ペアを生成しているか	5.2節	□済	
	③-4)【遵守項目】上記③-3)についての指示を仕様書や運用手順書等に明記したか	5.2節	□済	
	③-5)【遵守項目】 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	5.2節	□済	
④暗号スイート設定	④-1)【遵守項目】表21記載の暗号アルゴリズムを全てを設定無効（利用不可）にしたか	5.3節	□済	
	④-2) ECDHEを利用する暗号スイートを設定するか。 設定しない場合は「設定せず」をチェックする（④-2-1のチェック不要）		□設定する	□設定せず
	④-2-1)【遵守項目】ECDHEの鍵長を256[illegible]		□済	
	④-3) DHEを利用する暗号スイートを設定す 設定しない場合は「設定せず」をチェック[illegible]		□設定する	□設定せず
	④-3-1)【遵守項目】DHEの鍵長を2048ビット以上に設定したか	[illegible]節	□済	
	④-4)【推奨項目】表22記載の暗号アルゴリズムを組み合わせた暗号スイートのみで設定しているか。 設定しない／できない場合は「設定せず」をチェックする	5.3節	□済	□設定せず
	④-5) 暗号スイートの優先順位が設定できるか。 設定できない場合は「設定不可」をチェックする（④-5-1のチェック不要）		□設定可	□設定不可
	④-5-1)【推奨項目】表24記載の暗号スイートの優先順位で設定したか。優先順位どおりに設定できない／しない場合は「設定せず」をチェックする	5.3節	□済	□設定せず
⑤附録	⑤) Appendix C：暗号スイートの設定例の最新版のドキュメントを参照して設定したか。参照していない場合には「参照せず」をチェックする	Appendix C	□参照済	□参照せず

A.2. 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チェックリスト】

チェック項目		参照章		
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)【遵守項目】TLS1.2を設定有効としたか	4.1節	□済	
	②-2)【遵守項目】SSL2.0からTLS1.1までを設定無効（利用不可）にしたか	4.1節	□済	
	②-3) TLS1.3が実装されているか。 実装されていない場合は「未実装」をチェックする（②-3-1のチェック不要）		□実装済	□未実装
	②-3-1)【推奨項目】TLS1.3について設定を有効にしたか。ただし、TLS1.3を明確に利用しない場合は「設定せず」をチェックする	4.1節	□済	□設定せず
③サーバ証明書設定	③-1)【遵守項目】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	4.2節	□済	
	③-2)【遵守項目】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	4.2節	□済	
	③-3)【遵守項目】サーバ証明書の発行・更新を行う際に、自ら公開鍵と秘密鍵の鍵ペアを生成する場合には、新たな公開鍵と秘密鍵の鍵ペアを生成しているか	4.2節	□済	
	③-4)【遵守項目】上記③-3)についての指示を仕様書や運用手順書等に明記したか	4.2節	□済	
	③-5)【遵守項目】 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	4.2節	□済	
④暗号スイート設定	④-1)【遵守項目】表15記載の暗号アルゴリズムを全てを設定無効（利用不可）にしたか	4.3節	□済	
	④-2) ECDHEを利用する暗号スイートを設定するか。 設定しない場合は「設定せず」をチェックする（④-2-1のチェック不要）		□設定する	□設定せず
	④-2-1)【遵守項目】ECDHEの鍵長を256ビット以上に設定したか	4.3節	□済	
	④-3) DHEを利用する暗号スイートを設定するか。 設定しない場合は「設定せず」をチェックする（④-3-1のチェック不要）		□設定する	□設定せず
	④-3-1)【遵守項目】DHEの鍵長を2048ビット以上に設定したか	4.3節	□済	
	④-4)【推奨項目】表16記載の暗号アルゴリズムを組み合わせた暗号スイートのみで設定しているか。 設定しない／できない場合は「設定せず」をチェックする	4.3節	□済	□設定せず
	④-5) 暗号スイートの優先順位が設定できるか。 設定できない場合は「設定不可」をチェックする（④-5-1のチェック不要）		□設定可	□設定不可
	④-5-1)【推奨項目】表18記載の暗号スイートの優先順位で設定したか。優先順位どおりに設定できない／しない場合は「設定せず」をチェックする	4.3節	□済	□設定せず
⑤附録	⑤) Appendix C：暗号スイートの設定例の最新版のドキュメントを参照して設定したか。参照していない場合には「参照せず」をチェックする	Appendix C	□参照済	□参照せず

【表15】（TLS暗号設定ガイドライン 4.3節）
※下記の暗号アルゴリズムのいずれかを含む暗号スイートは
「全種類」設定無効化（利用不可）とすること

【遵守項目】利用禁止暗号アルゴリズム一覧（2020年1月7日時点）		
鍵交換		DH
		ECDH
署名		GOST R 34.10-2012
暗号化	ブロック暗号	RC2
		EXPORT-RC2
		IDEA
		DES
		EXPORT-DES
		GOST 28147-89
		Magma
		3-key Triple DES
		Kuznyechik
		ARIA
		SEED
	暗号利用モード	CTR_OMAC
	ストリーム暗号	RC4
		EXPORT-RC4
ハッシュ関数		MD5
		GOST R 34.11-2012

【表16】（TLS暗号設定ガイドライン 4.3節）
※下記の暗号アルゴリズムだけを組み合わせた暗号スイートのみで
設定（利用可）されている

【推奨項目】利用推奨暗号アルゴリズム一覧		
鍵交換		ECDHE
		DHE
署名		ECDSA
		RSASSA PKCS#1 v1.5 (RSA)
		RSASSA-PSS (TLS1.3のみ)
暗号化	ブロック暗号	AES
		Camellia (TLS1.2のみ)
	暗号利用モード	GCM
		CCM
		CCM_8
		CBC
	ストリーム暗号	ChaCha20-Poly1305
ハッシュ関数		SHA-256
		SHA-384
		SHA-1

【表18】（TLS暗号設定ガイドライン 4.3節）

【推奨項目】TLS1.2を利用する場合の優先順位		
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_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_CCM	(0xC0,0xAC)
	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8	(0xC0,0xAE)
	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_CCM	(0xC0,0xAD)
	TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8	(0xC0,0xAF)
	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	(0xCC,0xA9)
	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	(0xCC,0xA8)
グループB	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_CCM	(0xC0,0x9E)
	TLS_DHE_RSA_WITH_AES_128_CCM_8	(0xC0,0xA2)
	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_CCM	(0xC0,0x9F)
	TLS_DHE_RSA_WITH_AES_256_CCM_8	(0xC0,0xA3)
	TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	(0xCC,0xAA)
グループC	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_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)
グループD	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	(0xC0,0x09)
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	(0xC0,0x13)
	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_DHE_RSA_WITH_AES_128_CBC_SHA256	(0x00,0x67)
	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256	(0x00,0xBE)
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	(0x00,0x6B)
	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256	(0x00,0xC4)
グループF	TLS_DHE_RSA_WITH_AES_128_CBC_SHA	(0x00,0x33)
	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA	(0x00,0x45)
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA	(0x00,0x39)
	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA	(0x00,0x88)

【推奨項目】TLS1.3を利用する場合の優先順位		
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_AES_128_GCM_SHA256	(0x13,0x01)
	TLS_AES_128_CCM_SHA256	(0x13,0x04)
	TLS_AES_128_CCM_8_SHA256	(0x13,0x05)
	TLS_AES_256_GCM_SHA384	(0x13,0x02)
	TLS_CHACHA20_POLY1305_SHA256	(0x13,0x03)
鍵交換	ECDHE（優先）	RFC8446に規定
	DHE	RFC8446に規定
署名	ECDSA	RFC8446に規定
	RSA-PSS	RFC8446に規定
	RSASSA-PKCS1-v1_5	RFC8446に規定

A.3. 高セキュリティ型のチェックリスト

【高セキュリティ型チェックリスト】

チェック項目		参照章		
①要求設定確認	①-1)【遵守項目】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	□該当	
②プロトコルバージョン設定	②-1)【遵守項目】TLS1.3を設定有効としたか	5.1節	□済	
	②-2)【遵守項目】TLS1.2を設定有効としたか。ただし、TLS1.2を明確に利用しないと判明している場合は「設定せず」をチェックする	5.1節	□済	□設定せず
	②-3)【遵守項目】SSL2.0からTLS1.1までを設定無効（利用不可）にしたか	5.1節	□済	
③サーバ証明書設定	③-1)【遵守項目】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	5.2節	□済	
	③-2)【遵守項目】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以上の組合せで鍵長2048ビット以上 ・ ECDSAとSHA-256以上の組合せで鍵長256ビット（NIST P-256）以上	5.2節	□済	
	③-3)【遵守項目】サーバ証明書の発行・更新を行う際に、自ら公開鍵と秘密鍵の鍵ペアを生成する場合には、新たな公開鍵と秘密鍵の鍵ペアを生成しているか	5.2節	□済	
	③-4)【遵守項目】上記③-3)についての指示を仕様書や運用手順書等に明記したか	5.2節	□済	
	③-5)【遵守項目】 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	5.2節	□済	
④暗号スイート設定	④-1)【遵守項目】表21記載の暗号アルゴリズムを全てを設定無効（利用不可）にしたか	5.3節	□済	
	④-2) ECDHEを利用する暗号スイートを設定するか。 設定しない場合は「設定せず」をチェックする（④-2-1のチェック不要）		□設定する	□設定せず
	④-2-1)【遵守項目】ECDHEの鍵長を256ビット以上に設定したか	5.3節	□済	
	④-3) DHEを利用する暗号スイートを設定するか。 設定しない場合は「設定せず」をチェックする（④-3-1のチェック不要）		□設定する	□設定せず
	④-3-1)【遵守項目】DHEの鍵長を2048ビット以上に設定したか	5.3節	□済	
	④-4)【推奨項目】表22記載の暗号アルゴリズムを組み合わせた暗号スイートのみで設定しているか。 設定しない／できない場合は「設定せず」をチェックする	5.3節	□済	□設定せず
	④-5) 暗号スイートの優先順位が設定できるか。 設定できない場合は「設定不可」をチェックする（④-5-1のチェック不要）		□設定可	□設定不可
	④-5-1)【推奨項目】表24記載の暗号スイートの優先順位で設定したか。優先順位どおりに設定できない／しない場合は「設定せず」をチェックする	5.3節	□済	□設定せず
⑤附録	⑤) Appendix C：暗号スイートの設定例の最新版のドキュメントを参照して設定したか。参照していない場合には「参照せず」をチェックする	Appendix C	□参照済	□参照せず

【表21】（TLS暗号設定ガイドライン 5.3節）
※下記の暗号アルゴリズムのいずれかを含む暗号スイートは
「全種類」設定無効化（利用不可）とすること

【遵守項目】利用禁止暗号アルゴリズム一覧（2020年1月7日時点）		
鍵交換		DH
		ECDH
		RSAES PKCS#1 v1.5 (RSA)
署名		GOST R 34.10-2012
暗号化	ブロック暗号	RC2
		EXPORT-RC2
		IDEA
		DES
		EXPORT-DES
		GOST 28147-89
		Magma
		3-key Triple DES
		Kuznyechik
		ARIA
		SEED
	暗号利用モード	CBC
		CTR_OMAC
	ストリーム暗号	RC4
		EXPORT-RC4
ハッシュ関数		MD5
		SHA-1
		GOST R 34.11-2012

【表22】（TLS暗号設定ガイドライン 5.3節）
※下記の暗号アルゴリズムだけを組み合わせた暗号スイートのみで
設定（利用可）されている

【推奨項目】利用推奨暗号アルゴリズム一覧		
鍵交換		ECDHE
		DHE
署名		ECDSA
		RSASSA PKCS#1 v1.5 (RSA)
		RSASSA-PSS（TLS1.3のみ）
暗号化	ブロック暗号	AES
		Camellia（TLS1.2のみ）
	暗号利用モード	GCM
		CCM
		CCM_8
	ストリーム暗号	ChaCha20-Poly1305
ハッシュ関数		SHA-256
		SHA-384

【表24】（TLS暗号設定ガイドライン 5.3節）

【推奨項目】TLS1.3を利用する場合の優先順位		
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_AES_256_GCM_SHA384	(0x13,0x02)
	TLS_CHACHA20_POLY1305_SHA256	(0x13,0x03)
	TLS_AES_128_GCM_SHA256	(0x13,0x01)
	TLS_AES_128_CCM_SHA256	(0x13,0x04)
	TLS_AES_128_CCM_8_SHA256	(0x13,0x05)
鍵交換	ECDHE（優先）	RFC8446に規定
	DHE	RFC8446に規定
署名	ECDSA	RFC8446に規定
	RSA-PSS	RFC8446に規定
	RSASSA-PKCS1-v1_5	RFC8446に規定

【推奨項目】TLS1.2を利用する場合の優先順位		
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_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_CCM	(0xC0,0xAD)
	TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8	(0xC0,0xAF)
	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	(0xCC,0xA9)
	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	(0xCC,0xA8)
	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_CCM	(0xC0,0xAC)
	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8	(0xC0,0xAE)
グループB	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_CCM	(0xC0,0x9F)
	TLS_DHE_RSA_WITH_AES_256_CCM_8	(0xC0,0xA3)
	TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	(0xCC,0xAA)
	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_CCM	(0xC0,0x9E)
	TLS_DHE_RSA_WITH_AES_128_CCM_8	(0xC0,0xA2)

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チェックリスト】

	チェック項目	参照章		
①要求設定確認	①-1) 【遵守項目】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	□該当	
	①-2) 【遵守項目】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	□済	
②プロトコルバージョン設定	②-1) 【遵守項目】SSL3.0及びSSL2.0を設定無効（利用不可）にしたか	6.1節	□済	
	②-2) TLS1.2が実装されているか。 実装されていない場合は「未実装」をチェックする（②-2-1のチェック不要）		□実装済	□未実装
	②-2-1) 【遵守項目】TLS1.2について設定を有効にしたか	6.1節	□済	
	②-3) TLS1.1とTLS1.0のいずれか、または両方の設定を有効にするか。 両方とも有効にしない場合は「設定せず」をチェックする（②-3-1のチェック不要）		□設定する	□設定せず
	②-3-1) 【遵守項目】TLS1.1とTLS1.0のいずれか、または両方を設定有効とする必要性を確認したか	6.1節	□済	
	②-4) TLS1.3が実装されているか。 実装されていない場合は未実装にチェックする（②-4-1のチェック不要）		□実装済	□未実装
	②-4-1) 【推奨項目】TLS1.3について設定を有効にしたか。 ただし、TLS1.3を明確に利用しないと判断している場合には「設定せず」をチェックする	6.1節	□済	□設定せず
	②-5) プロトコルバージョンの優先順位が設定できるか。 設定できない場合は「設定不可」にチェックする（②-5-1のチェック不要）		□設定可	□設定不可
	②-5-1) 【推奨項目】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バージョンで接続するように設定したか。設定しない場合は「設定せず」をチェックする	6.1節	□済	□設定せず
③サーバ証明書設定	③-1) 【遵守項目】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	6.2節	□済	
	③-2) 【遵守項目】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	6.2節	□済	
	③-3) 【遵守項目】サーバ証明書の発行・更新を行う際に、自ら公開鍵と秘密鍵の鍵ペアを生成する場合には、新たな公開鍵と秘密鍵の鍵ペアを生成しているか	6.2節	□済	
	③-4) 【遵守項目】上記③-3)についての指示を仕様書や運用手順書等に明記したか	6.2節	□済	
	③-5) 【遵守項目】 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	6.2節	□済	
④暗号スイート設定	④-1) 【遵守項目】表27記載の暗号アルゴリズムを全てを設定無効（利用不可）にしたか	6.3節	□済	
	④-2) ECDHE／ECDHを利用する暗号スイートを設定するか。 設定しない場合は「設定せず」をチェックする（④-2-1のチェック不要）		□設定する	□設定せず
	④-2-1) 【遵守項目】ECDHE/ECDHの鍵長を256ビット以上に設定したか	6.3節	□済	
	④-3) DHE/DHを利用する暗号スイートを設定するか。 設定しない場合は「設定せず」をチェックする（④-3-1のチェック不要）		□設定する	□設定せず
	④-3-1) 【遵守項目】DHE/DHの鍵長を1024ビット以上に設定したか	6.3節	□済	
	④-4) 【推奨項目】表28記載の暗号アルゴリズムを組み合わせた暗号スイートのみで設定しているか。 設定しない／できない場合には「設定せず」をチェックする	6.3節	□済	□設定せず
	④-5) 暗号スイートの優先順位が設定できるか。 設定できない場合は「設定不可」をチェックする（④-5-1のチェック不要）		□設定可	□設定不可
	④-5-1) 【推奨項目】表30記載の暗号スイートの優先順位で設定したか。 優先順位どおりに設定できない／しない場合には「設定せず」をチェックする	6.3節	□済	□設定せず
⑤附録	⑤) Appendix C：暗号スイートの設定例の最新版のドキュメントを参照して設定したか。参照していない場合には「参照せず」をチェックする	Appendix C	□参照済	□参照せず

【表27】（TLS暗号設定ガイドライン 6.3節）
※下記の暗号アルゴリズムのいずれかを含む暗号スイートは
「全種類」設定無効化（利用不可）とすること

【遵守項目】利用禁止暗号アルゴリズム一覧（2020年1月7日時点）		
署名		GOST R 34.10-2012
暗号化	ブロック暗号	RC2
		EXPORT-RC2
		IDEA
		DES
		EXPORT-DES
		GOST 28147-89
		Magma
		3-key Triple DES
		Kuznyechik
		ARIA
		SEED
	暗号利用モード	CTR_OMAC
	ストリーム暗号	RC4
		EXPORT-RC4
ハッシュ関数		MD5
		GOST R 34.11-2012

【表28】（TLS暗号設定ガイドライン 6.3節）
※下記の暗号アルゴリズムだけを組み合わせた暗号スイートのみで
設定（利用可）されている

【推奨項目】利用推奨暗号アルゴリズム一覧		
鍵交換		DHE
		ECDHE
		RSAES PKCS#1 v1.5 (RSA)
		DH
		ECDH
署名		RSASSA PKCS#1 v1.5 (RSA)
		ECDSA
		RSASSA-PSS (TLS1.3のみ)
暗号化	ブロック暗号	AES
		Camellia (TLS1.2まで)
	暗号利用モード	GCM
		CCM
		CCM_8
		CBC
	ストリーム暗号	ChaCha20-Poly1305
ハッシュ関数		SHA-256
		SHA-384
		SHA-1

【表30】（TLS暗号設定ガイドライン 6.3節）

【推奨項目】TLS1.2を利用する場合の優先順位		
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X	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	{0x00,0x9E}
	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	{0xC0,0x2B}
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	{0xC0,0x2F}
	TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256	{0xC0,0x7C}
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256	{0xC0,0x86}
	TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256	{0xC0,0x8A}
	TLS_DHE_RSA_WITH_AES_128_CCM	{0xC0,0x9E}
	TLS_ECDHE_ECDSA_WITH_AES_128_CCM	{0xC0,0xAC}
	TLS_DHE_RSA_WITH_AES_128_CCM_8	{0xC0,0xA2}
	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8	{0xC0,0xAE}
	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	{0x00,0x9F}
	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	{0xC0,0x2C}
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	{0xC0,0x30}
	TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384	{0xC0,0x7D}
	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384	{0xC0,0x87}
	TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384	{0xC0,0x8B}
	TLS_DHE_RSA_WITH_AES_256_CCM	{0xC0,0x9F}
	TLS_ECDHE_ECDSA_WITH_AES_256_CCM	{0xC0,0xAD}
	TLS_DHE_RSA_WITH_AES_256_CCM_8	{0xC0,0xA3}
	TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8	{0xC0,0xAF}
	TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	{0xCC,0xAA}
	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	{0xCC,0xA9}
	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	{0xCC,0xA8}
グループY	TLS_RSA_WITH_AES_128_GCM_SHA256	{0x00,0x9C}
	TLS_DH_RSA_WITH_AES_128_GCM_SHA256	{0x00,0xA0}
	TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256	{0xC0,0x2D}
	TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256	{0xC0,0x31}
	TLS_RSA_WITH_CAMELLIA_128_GCM_SHA256	{0xC0,0x7A}
	TLS_DH_RSA_WITH_CAMELLIA_128_GCM_SHA256	{0xC0,0x7E}
	TLS_ECDH_ECDSA_WITH_CAMELLIA_128_GCM_SHA256	{0xC0,0x88}
	TLS_ECDH_RSA_WITH_CAMELLIA_128_GCM_SHA256	{0xC0,0x8C}
	TLS_RSA_WITH_AES_128_CCM	{0xC0,0x9C}
	TLS_RSA_WITH_AES_128_CCM_8	{0xC0,0xA0}
	TLS_RSA_WITH_AES_256_GCM_SHA384	{0x00,0x9D}
	TLS_DH_RSA_WITH_AES_256_GCM_SHA384	{0x00,0xA1}
	TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	{0xC0,0x2E}
	TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	{0xC0,0x32}
	TLS_RSA_WITH_CAMELLIA_256_GCM_SHA384	{0xC0,0x7B}
	TLS_DH_RSA_WITH_CAMELLIA_256_GCM_SHA384	{0xC0,0x7F}
	TLS_ECDH_ECDSA_WITH_CAMELLIA_256_GCM_SHA384	{0xC0,0x89}
	TLS_ECDH_RSA_WITH_CAMELLIA_256_GCM_SHA384	{0xC0,0x8D}
	TLS_RSA_WITH_AES_256_CCM	{0xC0,0x9D}
	TLS_RSA_WITH_AES_256_CCM_8	{0xC0,0xA1}

（グループZに続く）

【表30（続）】（TLS暗号設定ガイドライン 6.3節）

【推奨項目】TLS1.2を利用する場合の優先順位（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Z	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_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)
	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_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)
	TLS_RSA_WITH_AES_128_CBC_SHA256	(0x00,0x3C)
	TLS_DH_RSA_WITH_AES_128_CBC_SHA256	(0x00,0x3F)
	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	(0xC0,0x25)
	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	(0xC0,0x29)
	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256	(0x00,0xBA)
	TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA256	(0x00,0xBC)
	TLS_ECDH_ECDSA_WITH_CAMELLIA_128_CBC_SHA256	(0xC0,0x74)
	TLS_ECDH_RSA_WITH_CAMELLIA_128_CBC_SHA256	(0xC0,0x78)
	TLS_RSA_WITH_AES_128_CBC_SHA	(0x00,0x2F)
	TLS_DH_RSA_WITH_AES_128_CBC_SHA	(0x00,0x31)
	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	(0xC0,0x04)
	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	(0xC0,0x0E)
	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA	(0x00,0x41)
	TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA	(0x00,0x43)
	TLS_RSA_WITH_AES_256_CBC_SHA256	(0x00,0x3D)
	TLS_DH_RSA_WITH_AES_256_CBC_SHA256	(0x00,0x69)
	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	(0xC0,0x26)
	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	(0xC0,0x2A)
	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256	(0x00,0xC0)
	TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA256	(0x00,0xC2)
	TLS_ECDH_ECDSA_WITH_CAMELLIA_256_CBC_SHA384	(0xC0,0x75)
	TLS_ECDH_RSA_WITH_CAMELLIA_256_CBC_SHA384	(0xC0,0x79)
	TLS_RSA_WITH_AES_256_CBC_SHA	(0x00,0x35)
	TLS_DH_RSA_WITH_AES_256_CBC_SHA	(0x00,0x37)
	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	(0xC0,0x05)
	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	(0xC0,0x0F)
	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA	(0x00,0x84)
	TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA	(0x00,0x86)
【推奨項目】TLS1.3を利用する場合の優先順位		
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_AES_128_GCM_SHA256	(0x13,0x01)
	TLS_AES_128_CCM_SHA256	(0x13,0x04)
	TLS_AES_128_CCM_8_SHA256	(0x13,0x05)
	TLS_AES_256_GCM_SHA384	(0x13,0x02)
	TLS_CHACHA20_POLY1305_SHA256	(0x13,0x03)
鍵交換	ECDHE（優先）	RFC8446に規定
	DHE	RFC8446に規定
署名	ECDSA	RFC8446に規定
	RSA-PSS	RFC8446に規定
	RSASSA-PKCS1-v1_5	RFC8446に規定

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

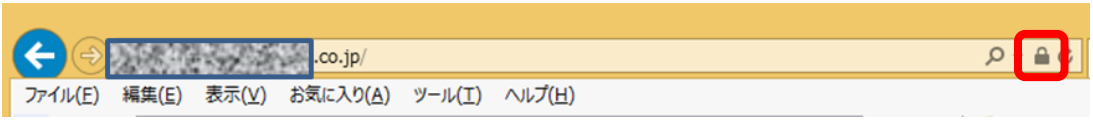
D.1.　ルート CA 証明書の暗号アルゴリズムおよび鍵長の確認方法

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 でアクセスする場合】

① EV 証明書の場合（グリーンバーになっている場合）は、錠前マーク横のサイト運営組織の表示をクリックする
EV 証明書でない場合は、錠前マークをクリックする

（EV 証明書を利用していないサイト）

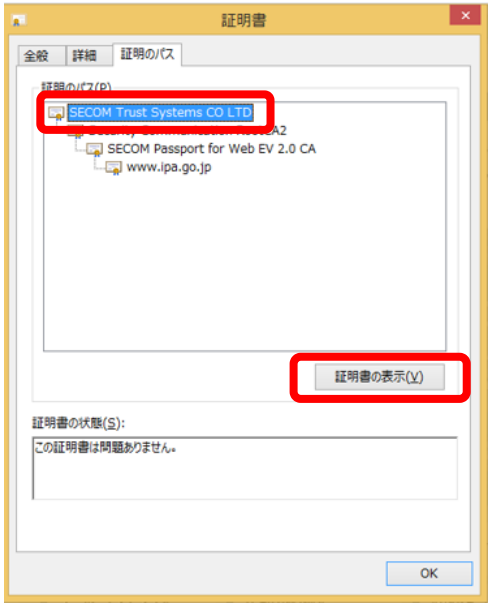


（EV 証明書を利用しているサイト）

② 「Web サイトの認証」ウィンドウの「証明書の表示」をクリックする

③　「証明書」ウィンドウの「証明のパス」タブをクリックする

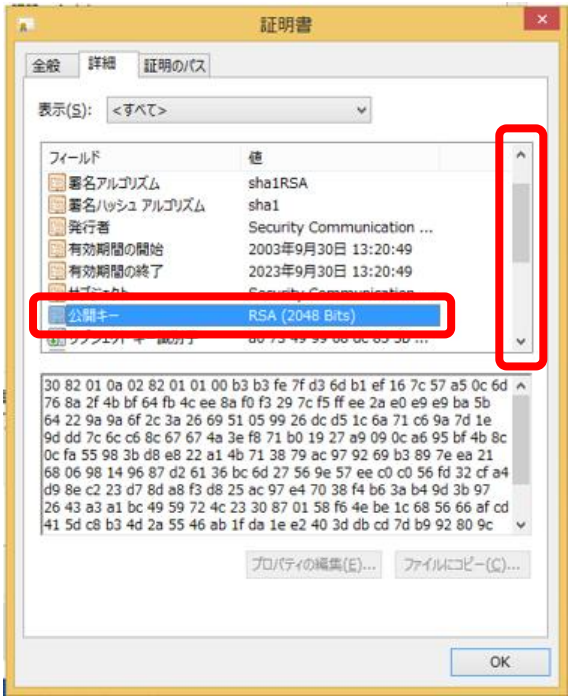
④　一番上に表示されている証明書（これがルート CA 証明書に当たる）を選択し、「証明書の表示」をクリックする



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



【Microsoft Edge（Chromium ベース）でアクセスする場合】

① 錠前マークをクリックする

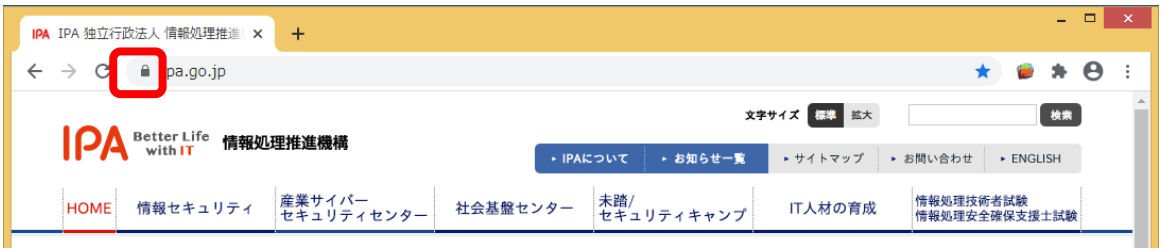
② 「証明書」をクリックする

なお、EV 証明書の場合は、発行先情報が表示される

③～⑥ 「Internet Explorer 11 でアクセスする場合」と同様の手順（③～⑥）で、「公開キー」フィールドに表示されている値を確認する。

【Google Chrome でアクセスする場合】

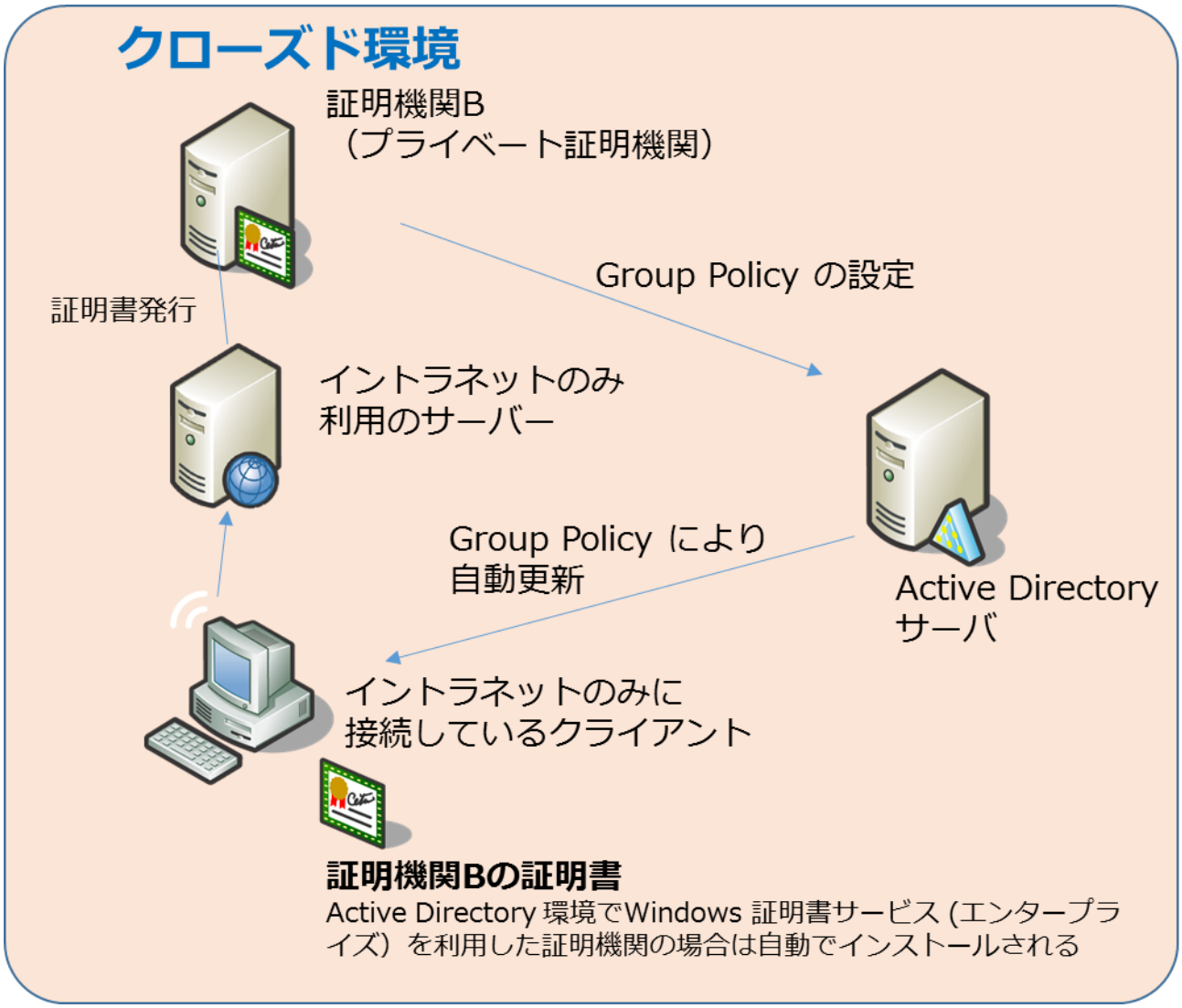
① 錠前マークをクリックする



② 「証明書」をクリックする
なお、EV 証明書の場合は、発行先情報が表示される

③～⑥ 「Internet Explorer 11 でアクセスする場合」と同様の手順（③～⑥）で、「公開キー」フィールドに表示されている値を確認する。

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

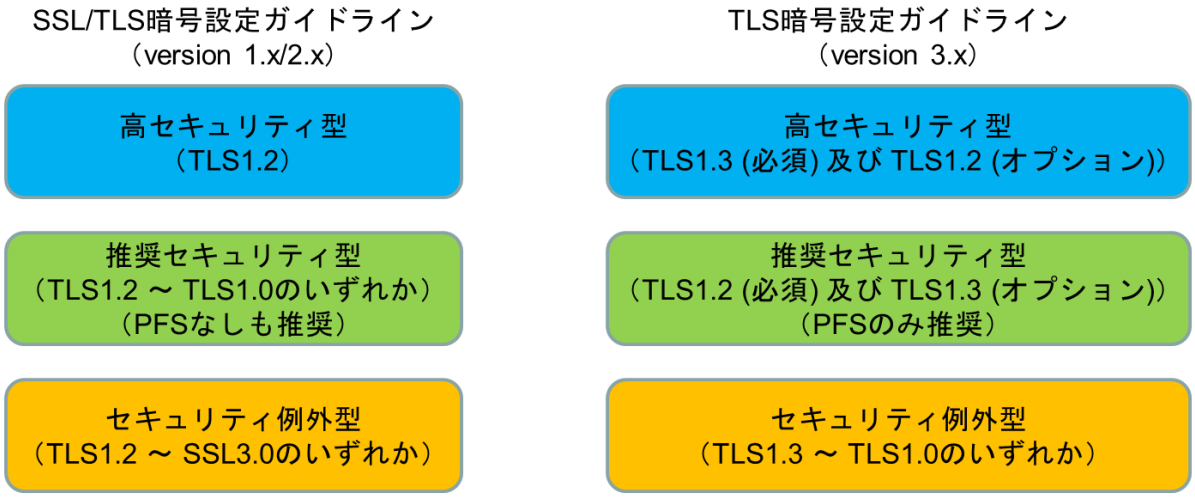


Appendix E：version 1.x/2.x と version 3.x の大きな差分

「SSL/TLS 暗号設定ガイドライン（version 1.x/2.x)」と「TLS 暗号設定ガイドライン（version 3.x)」との大きな差分は以下の通りである。

1. TLS1.3 の採用及び SSL3.0 の禁止に伴う各設定基準における要求設定の変更

TLS 暗号設定ガイドライン（version 3.x）では、プロトコルバージョンの要求設定において TLS1.3 の採用及び SSL3.0 の禁止が行われた。これに伴い、各設定基準における要求設定についても大幅な変更が行われており、SSL/TLS 暗号設定ガイドライン（version 1.x/2.x）における設定基準から一段階高い安全性を求めるようになった項目も多い。例えば、推奨セキュリティ型で利用が認められていた TLS1.0 や TLS1.1 は、本ガイドラインではセキュリティ例外型のみで利用可能となった。また、鍵交換では Perfect Forward Secrecy の特性をもつ ECDHE や DHE をさらに強く推奨するようにした。



2. 要求設定における「遵守項目」と「推奨項目」の区分けの新設

SSL/TLS 暗号設定ガイドライン（version 1.x/2.x）では全ての設定項目について一律に「要求設定」と位置付けていた。

今回は、設定項目のおける安全性への寄与度を考慮し、TLS 暗号設定ガイドライン（version 3.x）では、選択した設定基準としての最低限の安全性を確保するために必ず満たさなければならない「遵守項目」と当該設定基準としてよりよい安全性を実現するために満たすことが望ましい「推奨項目」とに分け、より現実的かつ実効性が高い要求設定とした。

3. 章構成の変更

SSL/TLS 暗号設定ガイドライン（version 1.x/2.x）では、

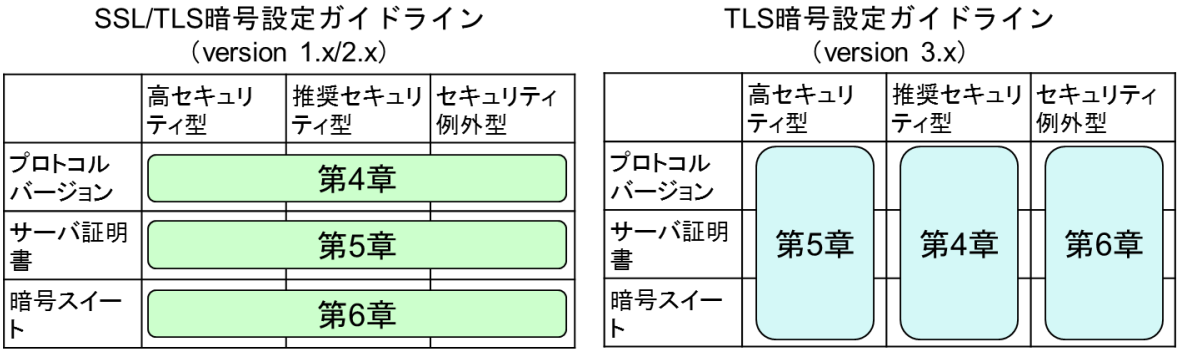
- プロトコルバージョンの設定（4 章）
- サーバ証明書の設定（5 章）
- 暗号スイートの設定（6 章）

の章構成とし、各章に「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」の設定項目を記載していた。

今回、章構成を見直し、TLS 暗号設定ガイドライン（version 3.x）では、

- 推奨セキュリティ型の要求設定（4 章）
- 高セキュリティ型の要求設定（5 章）
- セキュリティ例外型の要求設定（6 章）

の章構成とし、各章に「プロトコルバージョン」「サーバ証明書」「暗号スイート」の設定項目を記載した。これにより、選択した設定基準での該当章だけを参照すればよい構成とした。



発行日　2020 年 7 月 7 日　　第 3.0 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
（セキュリティセンター　セキュリティ技術評価部　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

TLS 暗号設定ガイドライン

～ 安全なウェブサイトのために（暗号設定対策編）～

【作成】

CRYPTREC
Cryptography Research and Evaluation Commitees

CRYPTREC とは、デジタル庁・総務省・経済産業省・NICT・IPA が実施している暗号技術評価プロジェクトのこと。暗号技術の専門家により安全性と実装性に優れると判断された CRYPTREC 暗号リストを作成・公表している。

【発行】

独立行政法人情報処理推進機構 セキュリティセンター

2015 年 5 月 12 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
2020 年 7 月 7 日　第 3.0 版