セキュリティエンジニアリングと標準規格

セキュリティエンジニアリング特論 第 2 回

栗原 淳

兵庫県立大学大学院

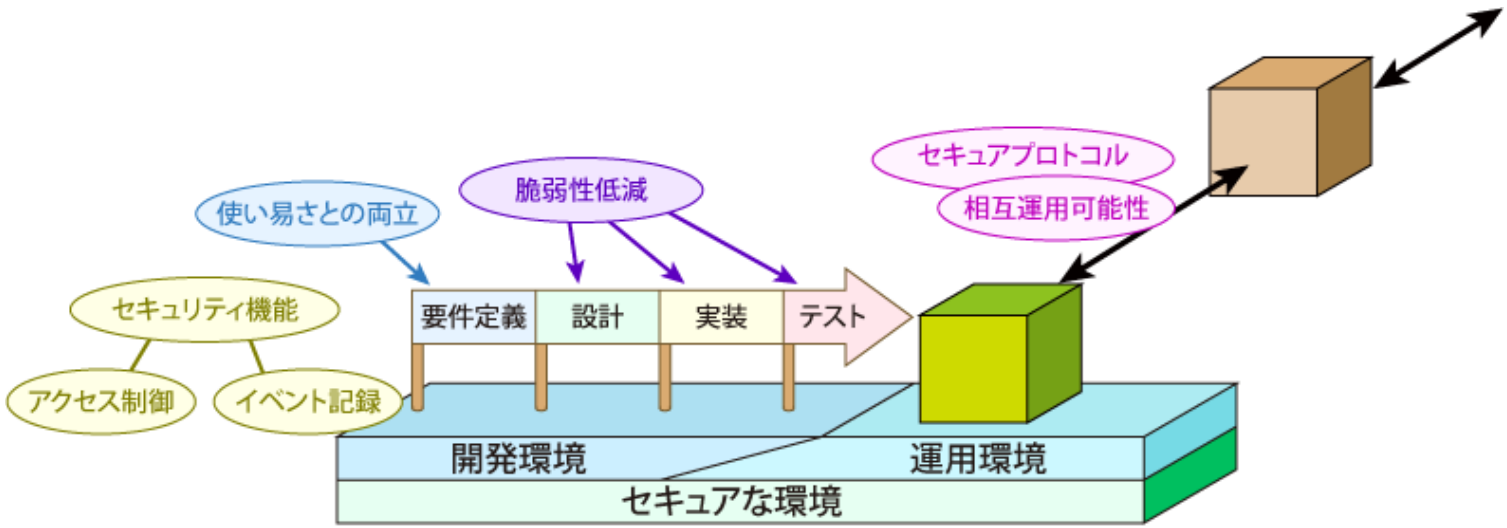
2024-10-10

はじめに

はじめに

セキュリティエンジニアリング[1]

ソフトウェア・アプリケーション開発において，セキュリティを考慮したエンジニアリング，あるいはセキュリティエンジニアリングを行うためには，要件定義・設計・実装・テストの段階ごとに種々のセキュリティ関連事項を検討する必要がある．



[1] https://www.ipa.go.jp/security/awareness/vendor/software.html (サイト刷新で消滅)

一方で，標準規格は:

標準規格となるアルゴリズム・プロトコル

国家や団体の標準文書に載せるために，安全性・相互接続性・効率性を分析し，安全性や将来性などがある程度確保されたもの，と言える．

$$\Downarrow$$

すなわち，標準規格を要件に応じて適切に選択し，設計・実装を行うことで脆弱性低減，使いやすさとの両立や相互運用可能性の担保が容易になると言える．

つまり，最新の標準規格とその推奨される利用方法[2]を把握しておくことで，効率的なセキュリティエンジニアリングが行える．

[2]なぜそのように利用するのか・載っているのか，も正しく知っておく必要がある．悪い例は，脆弱性はあるが互換性のために残っている RFC8017 の RSAES-PKCS1-v1_5．

セキュリティ関連の標準規格

PKCS (Public Key Cryptography Standards)

PKCS とは？

RSA Security 社[3] の研究部門 RSA Labs が策定・公開している，公開鍵暗号を中心とする一連のセキュリティ技術標準のこと．比較的古い，枯れた標準になる．

- #1,...#15 の 15 本が存在[4]．
- 暗号化・署名生成の手続き・アルゴリズム，データフォーマットの規定など，いわゆる「ローレベル」の技術標準が中心．
- RSA 暗号標準を定める#1 を代表に，重要なものはメンテされ続けている．が，他の新標準規格で代替されるものなどは破棄・管理移譲されている．

[3] RSA 暗号を作った Rivest-Shamir-Adleman の会社．

[4] 破棄・廃盤も含む．

PKCS は，その名目上「私企業」が策定した技術標準．
しかし，採用された技術は，十分にセキュリティ評価されていると見做されるものが多く，また他の技術標準に採用・移植・継承されている．特に多くは IETF RFC の管理下へ継承されているようだ．

PKCS 文書一覧 (1/2)

	Ver.	名称	内容
PKCS #1	v2.2	RSA Cryptography Specifications[5]	RSA 鍵ペアの構造，暗号化手法，署名手法を策定．
PKCS #3	v1.4	Diffie – Hellman Key Agreement Standard	Diffie-Hellman 鍵交換の仕様を策定．RFC では Internet Key Exchange (IKE) へ継承 (?)．
PKCS #5	v2.1	Password-Based Cryptography Specification[6]	パスワードからの鍵導出手法，暗号化手法 (PBKDF1/2, PBES1/2) の策定．
PKCS #6	廃止	Extended-Certificate Syntax Standard	X.509v1 証明書の拡張．X.509v3 へ統合されて廃止．
PKCS #7	廃止 (?)	Cryptographic Message Syntax Standard[7]	暗号メッセージ構文を策定．S/MIME に利用．より新しい仕様 (RFC5652) により廃止 (?)．
PKCS #8	廃止 (?)	Private-Key Information Syntax Specification[8]	秘密鍵フォーマットを策定．より新しい仕様 (RFC5968) により v1.2 で廃止 (?)．
PKCS #9	v2.0	Selected Object Classes and Attribute Types[9]	各種フォーマットにおける「属性」タイプを策定．

[5] https://tools.ietf.org/html/rfc8017
[6] https://tools.ietf.org/html/rfc8018
[7] https://tools.ietf.org/html/rfc2315
[8] https://tools.ietf.org/html/rfc5208
[9] https://tools.ietf.org/html/rfc2985

PKCS 文書一覧 (2/2)

	Ver.	名称	内容
PKCS #10	v1.7	Certification Request Syntax Specification[10]	証明書リクエスト構文を策定．元は PKCS のみで策定されていたが，利用されるメディアタイプを RFC5967 で拡張．
PKCS #11	v2.40	Cryptographic Token Interface	Cryptoki としても知られる，暗号トークン (H/W セキュリティモジュール) インターフェースの仕様を策定．OASIS PKCS 11 Technical Committee へ継承．
PKCS #12	v1.1	Personal Information Exchange Syntax Standard[11]	パスワード暗号化された秘密鍵，公開鍵証明書の構文を策定．IETF IESG 管理下へ継承．
PKCS #15	v1.1	Cryptographic Token Information Format Standard	暗号トークン向け，ユーザ特定標準仕様の策定．IC カード部分は ISO/IEC 7816-15 へ移譲．

策定中のまま立ち消えたものなどは削除．
IETF RFC などへ Republication，あるいは継承されて新しい標準になっている．

[10] https://tools.ietf.org/html/rfc2986 + https://tools.ietf.org/html/rfc5967
[11] https://tools.ietf.org/html/rfc7292

NIST FIPS および SP800[12]

NIST FIPS/SP800 とは？

米国国立標準技術研究所 (NIST; National Institute of Standards and Technology) の発行する文書のこと．

- FIPS; Federal Information Processing Standards: 米国商務長官の承認の下，NIST が公布した情報セキュリティ関連の米国の標準規格文書．詳細な基準や要求事項，ガイドラインが記載されている．
- SP800; Special Publication: 米国政府がセキュリティ対策を実施する際に参考とすることを前提とした，コンピュータセキュリティ関係のレポート．

[12]参考: https://www.ipa.go.jp/security/publications/nist/

すなわち NIST FIPS は，米国ローカルの標準規格と言える．

- 多くは，他の国別標準規格同様に ANSI/ISO/IEEE 等で広く使われていた既存規格を引き継ぐ．
- 一部は NIST FIPS 独自に公募・評価・策定した独自規格．代表的なものは例えば以下の通り:
  - 公募されてきた ‘Rijndael’ という新暗号アルゴリズムを採用した FIPS 197; Advanced Encryption Standard (AES)．
  - 2024 年に公募から選定された耐量子公開鍵暗号の一連のアルゴリズム: FIPS 203, 204, 205[13]

[13] FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard (CRYSTALS-Dilithium), FIPS 204: Module-Lattice-Based Digital Signature Standard (CRYSTALS-KYBER), FIPS 205: Stateless Hash-Based Digital Signature Standard (SPHINCS$^+$)

IETF RFC

RFC (Request for Comments) とは？

「インターネット技術」全般の国際標準を議論策定するグループ IETF (Internet Engineering Task Force) で議論策定された，「インターネット技術標準」および「その他」[14] の広範な内容を扱う文書 (群) のこと．

詳細仕様を策定する ITU-T や ISO と異なり，「まずは動作させる」ことを目的として実験的な「Rough」な仕様をまず策定することが特徴．

[14] https://www.nic.ad.jp/ja/rfc-jp/RFC-Category.html

RFC は 5 つのカテゴリに分類される:

- Standards Track: Proposed Standard → Internet Standard という策定過程を経る「インターネット標準技術仕様」の文書.
- Informational: すでにデファクト標準であったり，インターネット標準の議論・策定において有益として公開されたもの. 例えば，RSA セキュリティ社の PKCS#1 v2.1 = RFC8017.
- Experimental: デファクト標準を狙うような，研究等の目的で公開される技術仕様文書.
- Historical: 過去の記録として残す情報としての文書.
- Best Current Practice: 現状のベストプラクティスをまとめた仕様文書.

特に Standard Track, Informational, Experimental に関して，PKCS 等の他標準を引き継いだり，新たな技術標準を定めた文書が策定される.

セキュリティ関係の RFC 化の事例：

- 事例 1: OpenID Connect によって利用される鍵や署名，暗号化の仕様: JWS[15], JWE[16], JWK[17], JWT[18] について，OpenID Foundation のメンバにより，RFC Standards Track として国際標準化．
- 事例 2: PKCS#1, #5, #9 等の RSA セキュリティ社の独自標準は，Informational として RFC 化．
- 事例 3: HTTPS を支える TLS v1.3[19] は，Standards Track として RFC 化．
- 事例 4: ハイブリッド暗号化を「API を共通化し，パラメータを限定し，最新の暗号仕様を反映する」ことで相互運用性を高める規定，“Hybrid Public Key Encryption” は Informational として RFC 化．[20]

[15] JSON Web Signature https://tools.ietf.org/html/rfc7515
[16] JSON Web Encryption https://tools.ietf.org/html/rfc7516
[17] JSON Web Key https://tools.ietf.org/html/rfc7517
[18] JSON Web Token https://tools.ietf.org/html/rfc7519
[19] https://tools.ietf.org/html/rfc8446
[20] https://www.rfc-editor.org/rfc/rfc9180.html

ISO/IEC JTC 1

ISO[21]/IEC[22] JTC (Joint Technical Committee) 1 とは

情報技術の分野で国際標準化を行うための，ISO と IEC の第一合同技術委員会のこと．セキュリティ技術は，第 27 subcommittee (SC27)．

ISO/IEC JTC 1 は各国が提案を持ち寄り議論が進められる国際標準化団体．他の標準規格からの引継ぎだけではなく，国際会議等で発表された新しいメカニズムやアルゴリズムも標準規格として提案される．

[21] 国際標準化機構; International Organization for Standardization

[22] 国際電気標準会議; International Electrotechnical Commission

ISO/IEC JTC1 SC27 は 5 つの Working Group で構成される [23]．特に WG2 で扱われるものが，セキュリティ「技術」や「メカニズム」の標準規格となる．

- WG 1 情報セキュリティマネジメントシステム (いわゆる ISMS)
- WG 2 暗号とセキュリティメカニズム: 暗号アルゴリズム・プロトコルの標準規格．他の標準規格からの引継ぎだけでなく，各国から新たに提案されたものを議論し，標準規格を決定している．
- WG3 セキュリティの評価・試験・仕様
- WG4 セキュリティコントロールとサービス
- WG5 アイデンティティ管理とプライバシー技術

[23] https://itscj.ipsj.or.jp/committee-activities/committee-list.html

W3C (World Wide Web Consortium)

W3C とは？

WWW で用いられる Web 技術の標準化，相互運用性の確保を目的とする団体．ブラウザの API や HTML, XML, DOM 等の標準規格を「勧告 (Recommendation)」として策定する．

W3C のセキュリティ関連 WG (Working Group) で有名な活動として，以下のような国際標準化策定が上げられる．

- WebCrypto WG[24]: WebCrypto API を策定，勧告として国際標準化．
- WebAuthn WG[25]: FIDO アライアンスの技術仕様を勧告として国際標準化 [26]
- WebAssembly WG[27]: Web ブラウザ内の OS から隔離された「サンドボックス」の上で，安全にバイナリを実行するための標準規格。2019 年 12 月に勧告として国際標準化．

---

[24] https://www.w3.org/2012/webcrypto/ 現状は Close.

[25] https://www.w3.org/blog/webauthn/

[26] 対象はブラウザ・端末・認証サーバの連携プロトコルである FIDO2 WebAuthn https://www.w3.org/2019/03/pressrelease-webauthn-rec.html.ja．デバイス連携プロトコルである FIDO2 CTAP は ITU-T で国際標準化．

[27] https://webassembly.org/

ITU-T SG17

ITU-T (International Telecommunication Union Telecommunication Standardization Sector) SG17 (Study Group 17) とは？ [28]

- ITU-T: ITU (International Telecom. Union; 国際電気通信連合) における通信分野の標準技術を策定する「電気通信標準化部門」．策定された標準は「勧告」として発行される．
- Study Group 17: ITU-T においてセキュリティ関連勧告作成の中心となるグループ．

[28] https://www.ituaj.jp/wp-content/uploads/2016/07/2016_08-06-spotITU-T.pdf

ITU-T SG17 で取り扱う技術は，SDN・IoT・ITS・クラウドのセキュリティ技術，SPAM 対策，ID 管理技術，認証技術，テレバイオメトリクスなど，ソフトウェア実装のためのアルゴリズムというより「通信事業者」や「通信端末」を対象とした分野の技術.

昨今だと，「FIDO アライアンスの技術仕様を勧告として国際標準化[29]」している.

[29]対象はデバイス連携プロトコルである FIDO UAF 1.1 および CTAP https://fidoalliance.org/fido-alliance-specifications-now-adopted-as-itu-international-standards/. Web 関連プロトコルである FIDO2 WebAuthn は W3C で国際標準化.

3GPP

3GPP(Third Generation Partnership Project) とは

- 携帯電話システムの仕様策定を行う標準化「プロジェクト」. 各国の放送・通信の標準化団体 (ARIB@日本, ETSI@欧州, ATIS@アメリカ) が協議するプロジェクト.
- ここで決まったものを各国向けにローカライズして実装していく.
- 3GPP の「SA WG3」にて，セキュリティアーキテクチャやプロトコルの決定を行う．例えば，LTE の通信路暗号化向けアルゴリズムの標準化などを行った.

第 3 世代携帯電話 (3G) の仕様策定から始まったから 3GPP

その他; 各国の推奨技術リストとしての標準規格

- CRYPTOREC[30]
  電子政府推奨暗号リストを作り，その実装や運用方法も含めて安全性を調査・評価・監視・検討するプロジェクト (2000 年〜)
- NESSIE[31]
  EU の制定した暗号標準リストを策定するプロジェクト (2000 年〜)

基本的には，「評価検討した結果，既存のアルゴリズム・プロトコルのどれそれを標準として採用する」という「推奨技術リスト」の策定プロジェクトだと思って差し支えない．

[30] Cryptography Research and Evaluation Committee

[31] New European Schemes for Signature, Integrity, and Encryption

「各国独自」という意味

推奨技術リストへ採用されたアルゴリズム・プロトコルは，「その国において正しく評価された比較的安全なもの」というお墨付きを得る．
セキュリティ技術は国防上重要な意味を持つため，このお墨付きは，その技術を自国で利用して良いものかどうかを判定するもの，と言える．

仕様の詳細は IETF (RFC), ISO, NIST 公募など国際的に比較的オープンな場でまず評価・採用・策定される [32]．

その後，各国が独自に調査検討して推奨技術リストとして採用する，というケースが多い．

[32] 例外は存在する．元々 PKCS は RSA Labs. の独自標準を公開したものだったが，IETF の公開の場で Internet Draft の形で標準化されてきている．

その他; 諸々

- FIDO Alliance: フォーラム標準を定める業界団体．国際標準ではない．
- OpenID Foundation: 上記同様．
- Ecma International: JavaScript のメイン機能の標準規格 (ECMAScript) を策定する業界団体．JS 以外にも多種の規格策定，および Ecma 規格の国際標準化を行っている．[33]

[33] 例えば: Ecma 規格 ECMA-334 “C#” は，国際規格 ISO/IEC 23270 として標準化．

公開鍵・秘密鍵等の標準規格での表現形式

公開鍵・秘密鍵・証明書等の形式

公開鍵・秘密鍵・証明書などを表すための代表的な表現形式:

- DER (Distinguished Encoding Rules) 形式: ISO/ITU-T で定められた ANS.1 で記述されるエンコード方法．RSA 公開鍵・秘密鍵，楕円曲線暗号の公開鍵・秘密鍵など，それぞれエンコード方法を規定．
- PEM (Privacy Enhanced Mail[34]): DER の Base64 テキスト．
- SECG SEC1 形式: (楕円曲線暗号の鍵のみ) 業界団体 SECG で定められた公開鍵・秘密鍵のバイナリ表現方法．[35]
- JWK/JWE/JWS (JSON Web Key/Encryption/Signature) 形式: IETF で定められた JSON 形式での表現方法．[36]

OpenSSL 等で最も良く利用されるのは DER・PEM 形式の鍵．
JavaScript では鍵を JWK として使うことが多い．

[34] 元々メールを暗号化したデータのエンコード方法だったので．

[35] `http://www.secg.org/sec1-v2.pdf`

[36] `https://tools.ietf.org/html/rfc7517`, `https://tools.ietf.org/html/rfc7516`, `https://tools.ietf.org/html/rfc7515`

Table: 各データの表現形式の規定文書一覧

	DER/PEM	SECG SEC1	JWK/E/S
RSA 公開鍵	1) PKCS#1 2)RFC5280[37]	N/A	RFC7517
RSA 秘密鍵	RFC5958[38], PKCS#1[39]	N/A	RFC7517
ECC 公開鍵	RFC5480[40]	SECG SEC1 v2	RFC7517
ECC 秘密鍵	RFC5958[41], RFC5915[42]	SECG SEC1 v2	RFC7517
(AES) 共通鍵	N/A	N/A	RFC7517
公開鍵証明書	RFC5280 (他)	N/A	N/A [43]
RSA 暗号文	N/A	N/A	RFC7516
ECDH+AES 暗号文	N/A	N/A	RFC7516
AES 暗号文	N/A	N/A	RFC7516
HMAC	N/A	N/A	RFC7515
RSA 署名	N/A	N/A	RFC7515
ECDSA 署名	RFC5759	N/A	RFC7515

[37] 証明書の中にある SubjectPublicKeyInfo フィールド

[38] 秘密鍵の暗号化サポート．RFC5958 は PKCS#8 の拡張．

[39] RSA 秘密鍵自体の構造を定義

[40] 証明書の中にある SubjectPublicKeyInfo フィールド

[41] 秘密鍵の暗号化をサポートするフォーマット

[42] ECC 秘密鍵自体の構造を定義

[43] X.509 証明書への URI 等は記述可能 `https://tools.ietf.org/html/rfc7517`

標準規格の形式であっても，環境毎に対応が異なる

同じ JavaScript を使う環境であってもブラウザ側とサーバ側で…

WebCrypto API/Node.js でサポートする公開鍵・秘密鍵フォーマット

- WebCrypto API (フロントエンド/ブラウザ):
  RFC5280 形式 (公開鍵のみ)，RFC5958 形式 (秘密鍵のみ)，
  JWK
- Node.js Crypto (バックエンド/サーバ):
  RFC5280 形式 (公開鍵のみ)，RFC5958 形式 (秘密鍵のみ)，
  PKCS#1 形式

統一が取れておらず，環境に応じて適切な表現形式の変換が必要
⇒ 環境同士を結合するのに，標準文書を読み解いて変換するライブラリを作ったり…

次週に向けた環境準備解説

必要環境

割と最近の Web サービスのフロントエンド・バックエンドのエンジニアリング環境を模擬していく．

- Git が利用可能
- Bash/Zsh あたりが利用可能 [44]
- とりあえず Node.js v22[45] がインストール済み．あとは Go とか Rust とか．
  - パッケージマネージャ「pnpm」が利用可能なこと．
    シェルから「sudo npm i -g pnpm」でインストールできる．
- Google Chrome 系ブラウザ or Firefox がインストール済み
- (optional) コードの中身を見たければ，VS Code や Intelli J など自分の使いやすい高機能エディタを使うことが望ましい

[44] Windows の場合は WSL2 および PowerShell でなんとかすること．Ubuntu 24.04 LTS Desktop を VM に入れて実行するのを勧める (Virtualbox/VMWare 等)．

[45] `https://nodejs.org/ja/` 現時点では Latest だが 10 月末に LTS．

Ubuntu Desktop での Node.js インストール方法

シェルで以下を実行

```
$ sudo apt update && apt install -y nodejs npm
$ sudo npm i -g n
$ sudo n lts // 安定版がインストールされる
$ sudo apt purge -y nodejs npm // apt で入れたやつはいらない
$ npm i -g npm pnpm
```

WSL2 は基本 Ubuntu なので同様の操作でインストールできると思われるが，未確認．

macOS X での Node.js インストール方法

シェルで以下を実行

```
$ brew install n // homebrew は入っている前提
$ sudo n lts // 安定版がインストールされる
$ sudo npm i -g npm pnpm
```

基本，プログラミング (書くこと) は要求しないので，Node.js 等に触ったことがなくても安心して欲しい．ただ，ソースコードを読んで実行したり，中のデータを解説したりするので，なんとなく読めて意味がわかる程度には慣れておくのがベター．
まずはインストールしてバージョン表示までできれば OK．

シェルにて実行

```
~$ git --version
git version 2.42.0

~$ zsh --version
zsh 5.9 (x86_64-apple-darwin22.0)

~$ node --version
v20.5.1

~$ pnpm --version
8.7.6

~$ go version
go version go1.21.1 darwin/amd64

~$ cargo --version && rustc --version
cargo 1.72.1 (103a7ff2e 2023-08-15)
rustc 1.72.1 (d5c2e9c34 2023-09-13)
```

まとめ

まとめ

- 標準規格の意味と，いくつかの団体を紹介
- 例として「鍵フォーマット」の標準を紹介

次週からは…

- 次週からは，「安全な End-to-End 暗号化を実装する」のを例題として，ステップバイステップでシステムを構成していってみる．
- その中で，システムに採用する標準技術と，それが必要な理由を解説していく．
  ※ E2E 暗号化は標準「SSL/TLS」の基礎となる考え方．