

# 標準はどのように実装されているのか？

-- OpenSSLにおけるSSL/TLSの実装に関して --

富士ゼロックス株式会社  
稲田 龍

<Ryu.Inada@fujixerox.co.jp>



- SSL/TLSのおさらい
- OpenSSLとは何か?
- OpenSSLはどのように使われているか?
- まとめ

# SSL/TLSのおさらい

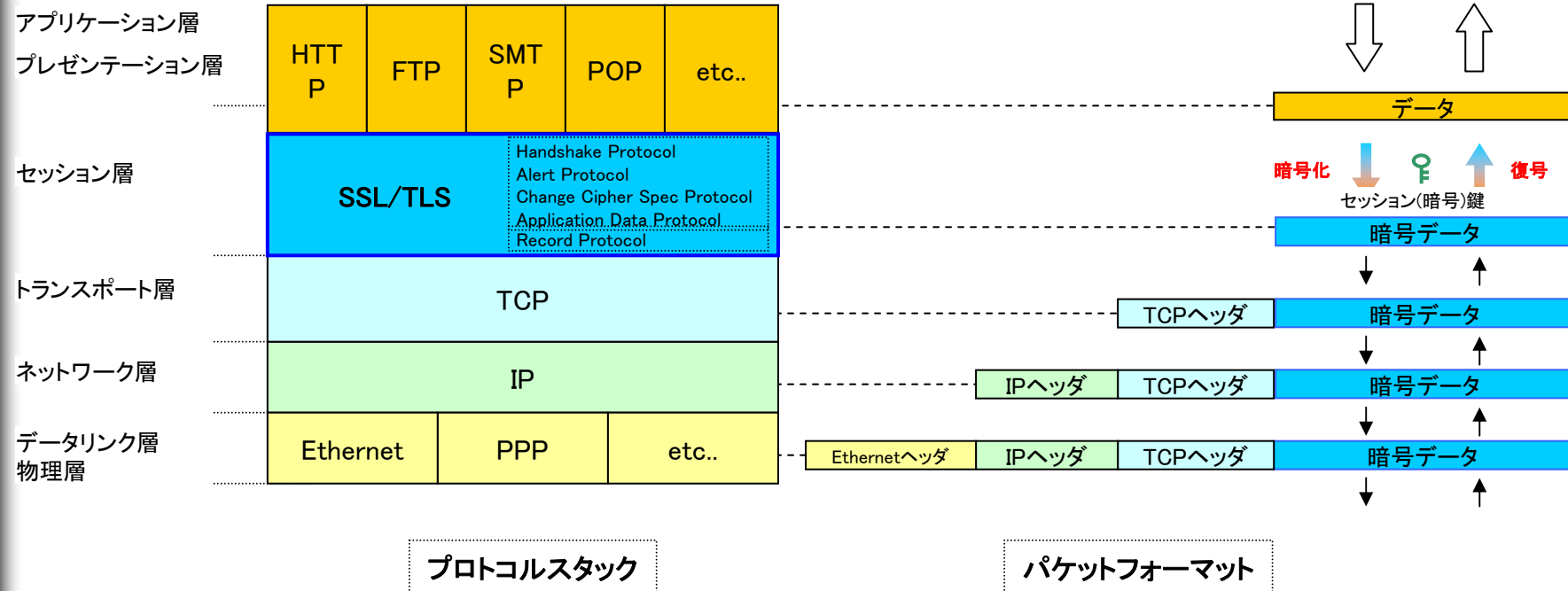


# SSL/TLSの機能

- トランスポート層でのセキュリティを保証
  - いわゆるSocket層の**Sublayer**として実装
  - 当初はHTTPのセキュリティ保証に用いられたが、**プロトコル全般**に利用可能
  - 伝送路の**暗号化**
  - 接続先の**認証**
    - サーバ側(必須)
    - クライアント側(任意)
  - **データインテグリティの保証**(HMAC)

# SSL/TLS – プロトコルスタック

OSI7階層モデル

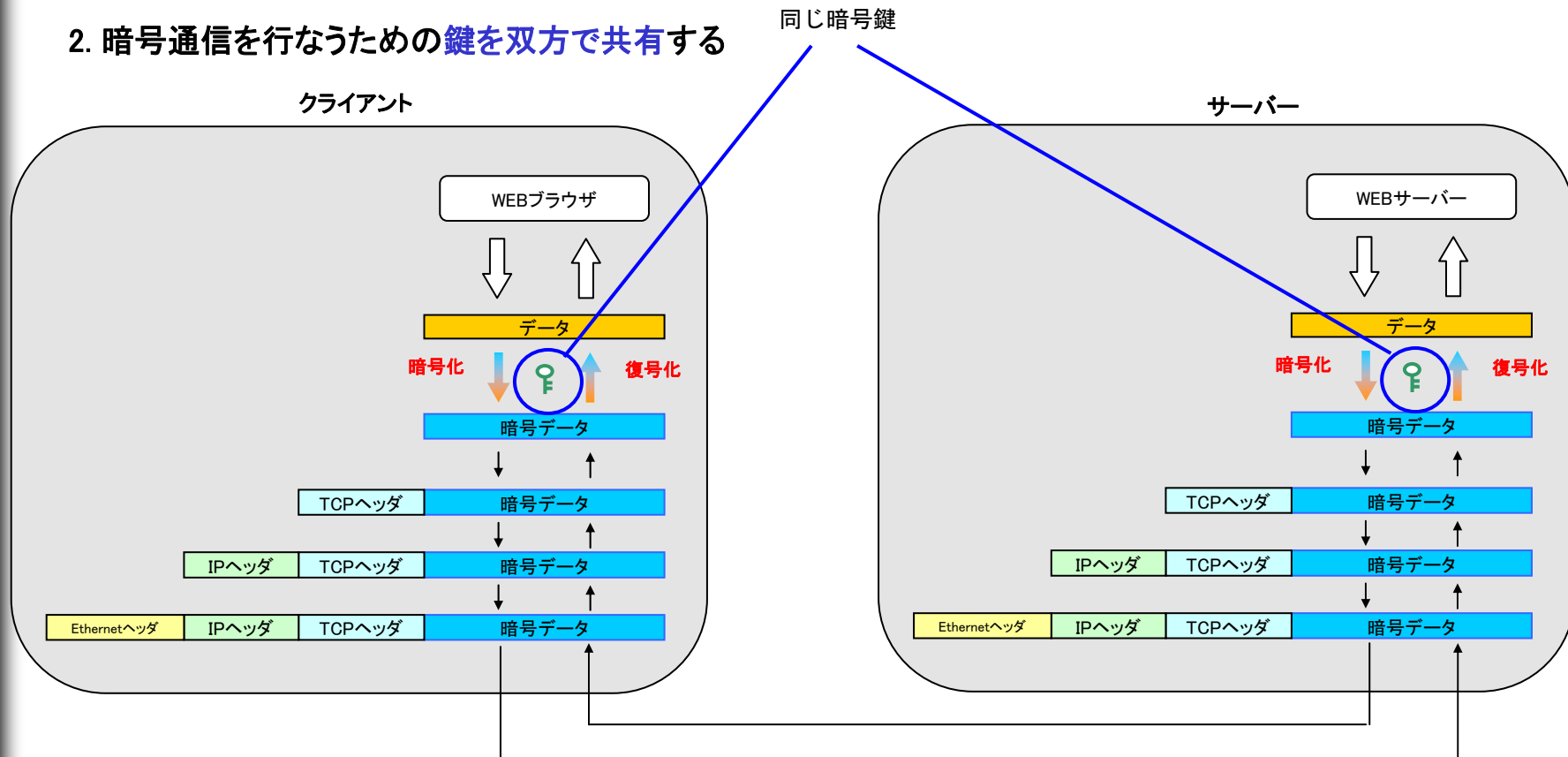


SSL/TLSはアプリケーションとトランスポートの間に位置するため、  
利用するアプリケーションに依存せずにセキュアな通信が行なうことができる。

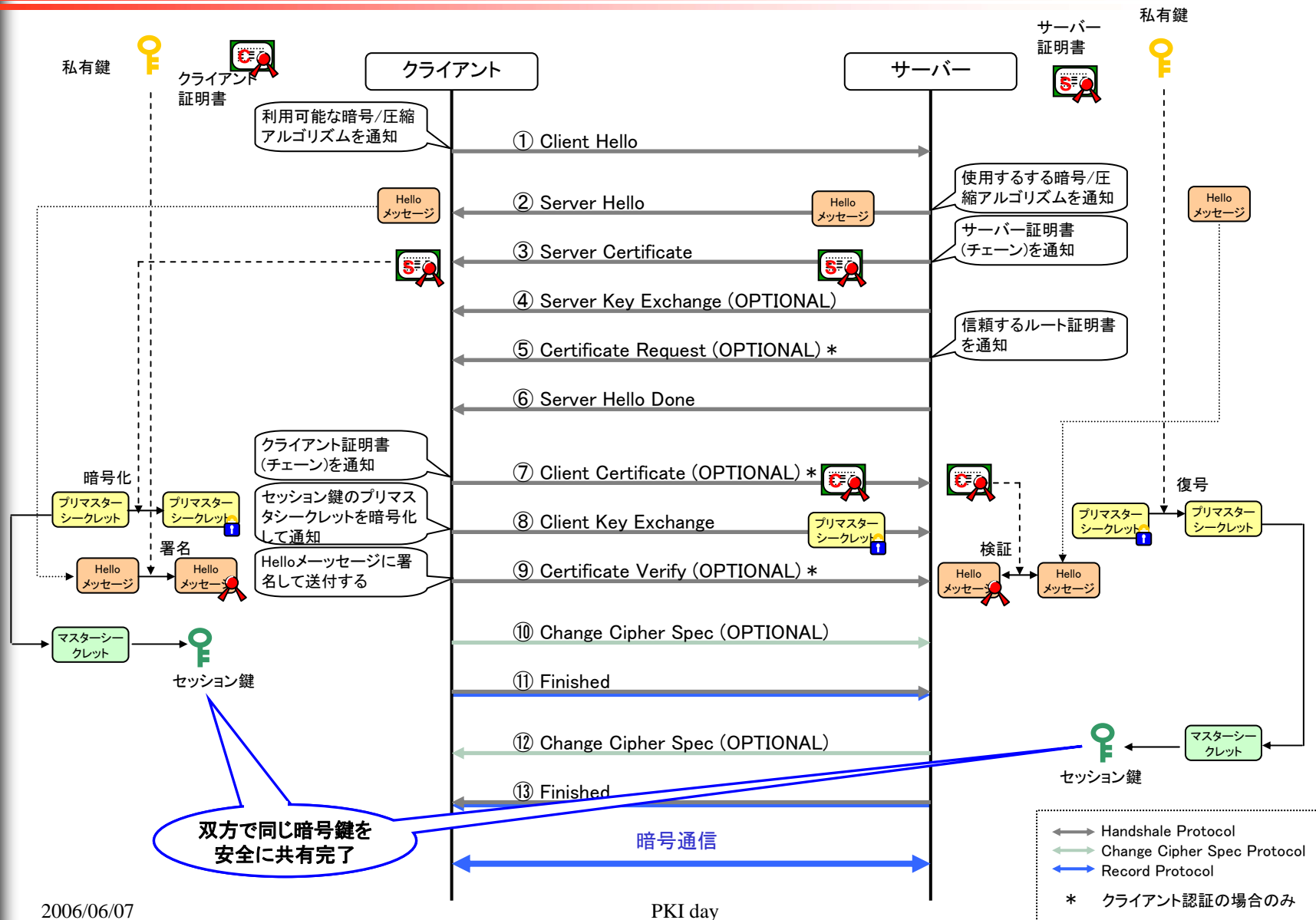
# SSL/TLS – 鍵共有

暗号通信を行なうためには...

1. 通信相手が正しい通信相手なのか認証する
2. 暗号通信を行なうための鍵を双方で共有する



# SSL/TLS - ハンドシェイク



# SSLの歴史

- SSL (Secure Socket Layer)
  - 1990年代前半にNetscape Communication社により開発されたもの
  - 1.0
    - 公開されなかった
  - 2.0
    - **Netscape Communicator 2.0**に実装
      - [http://wp.netscape.com/eng/security/SSL\\_2.html](http://wp.netscape.com/eng/security/SSL_2.html)
    - のちに**Internet Explorer 3.0**に採用
  - 3.0
    - 現在、広く使われているバージョン
      - <http://wp.netscape.com/eng/ssl3/>



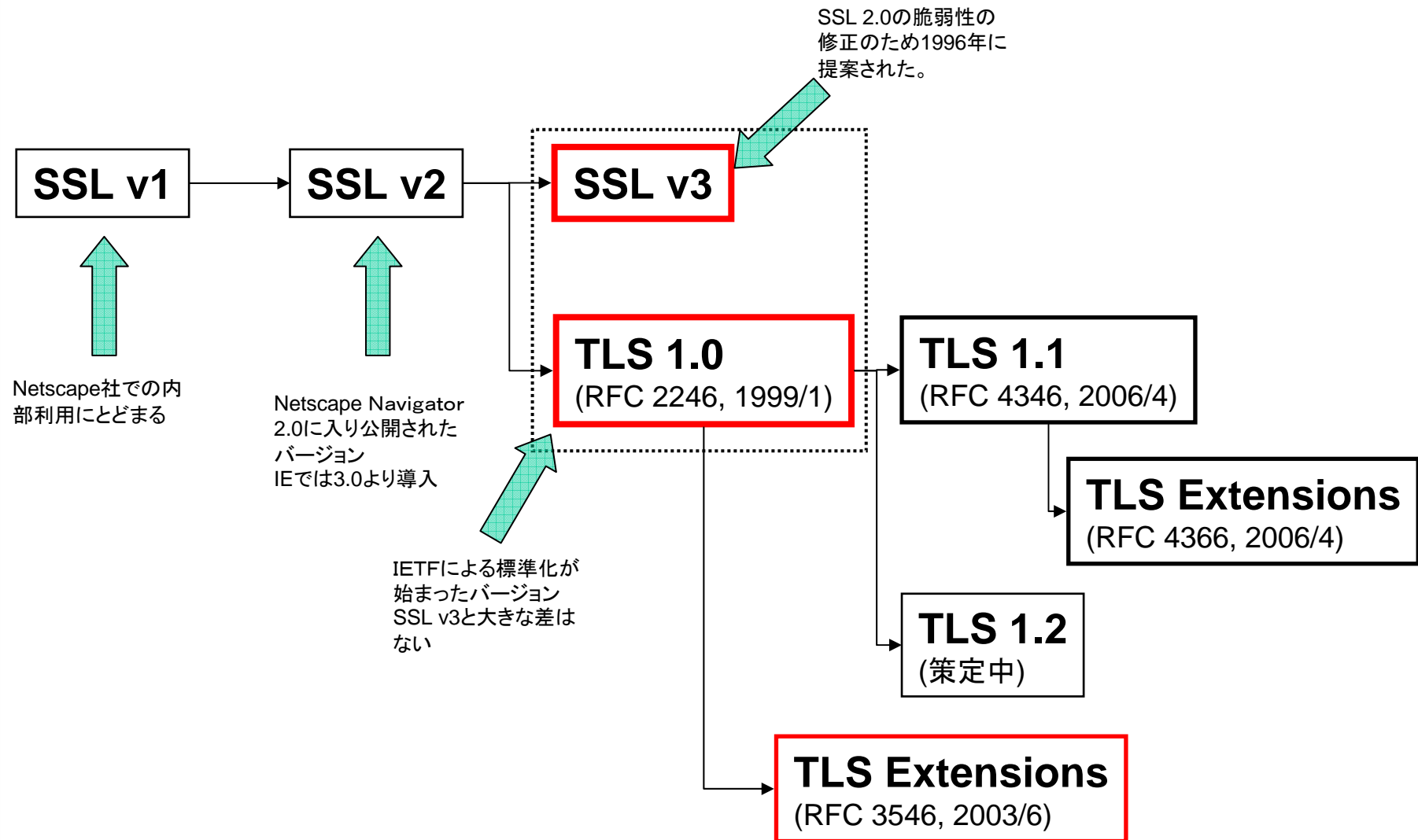
# TLSの歴史(その1)

- IETF TLS-WGにて標準化
  - <http://www.ietf.org/html.charters/tls-charter.html>
- 1.0
  - **RFC 2246**として1999年1月に制定
  - SSL v3とほぼ同等
- TLS Extensions
  - **RFC 3546**として2003年6月に制定
    - Virtual Server Extensions
    - Maximum fragmentation size negotiation
    - Client Certificate URL negotiation
    - Client indicate root certificate want to use
    - Truncated HMAC
    - Client Certificate status information method

# TLSの歴史(その2)

- 1.1
  - **RFC 4346**として2006年4月に制定
- 1.1 Extentions
  - **RFC 4366**として2006年4月に制定
  - RFC 3546のTLS 1.1対応版+  $\alpha$
- 1.2
  - I-D 4346bisとして策定中
  - アルゴリズム独立を目指している
  - 下位互換性に関して考慮が入っている
    - 古いプロトコルとは通信できないようにする場合あり

# SSL/TLS関連の標準の流れ



# OpenSSLとは何か？



# OpenSSLとは？

- もとはオーストラリアのEric A. Youngと友人のTim J. Hudsonが作ったSSLeayをベース
  - 少人数の天才が勢いに任せて書き進んだコードの典型？
  - Eric A. Youngが、1998に開発をやめ(RSA オーストラリアに移ったため)、引き取られ OpenSSLとなった。
  - <http://www.openssl.org/>

# OpenSSL Projectのゴール

- **The goal of the project**
  - The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library managed by a worldwide community of volunteers that use the Internet to communicate, plan, and develop the OpenSSL toolkit and its related documentation.
  - <http://www.openssl.org/about/> より引用

# OpenSSLの機能

- 3つの側面を持つ.....
  - SSL/TLSのデファクトスタック
  - 暗号エンジン
  - PKIアプリケーション開発ツールキット

# SSL/TLSのデファクトスタック

- サポートするプロトコル
  - SSL v2
    - 利用しないことが推奨されている
  - SSL v3
  - TLS 1.0
    - TLS Extensionsは、一部実装されている
  - DTLS (Datagram Transport Layer Security)



# 暗号エンジン

- サポートする暗号系
  - 公開鍵暗号
    - DH, RSA, DSA, 楕円関数
  - 共通鍵暗号
    - DES, Triple DES, RC2, RC4, AES, CAST, IDEA,
  - ハッシュ関数
    - MD2, MD4, MD5, MDC2, RIPEMD, SHA-1, SHA-256, SHA-512
  - HMAC
  
  - DSA
  - ECDSA
  - ECDH
  
  - Hardware Accelerator のサポートあり

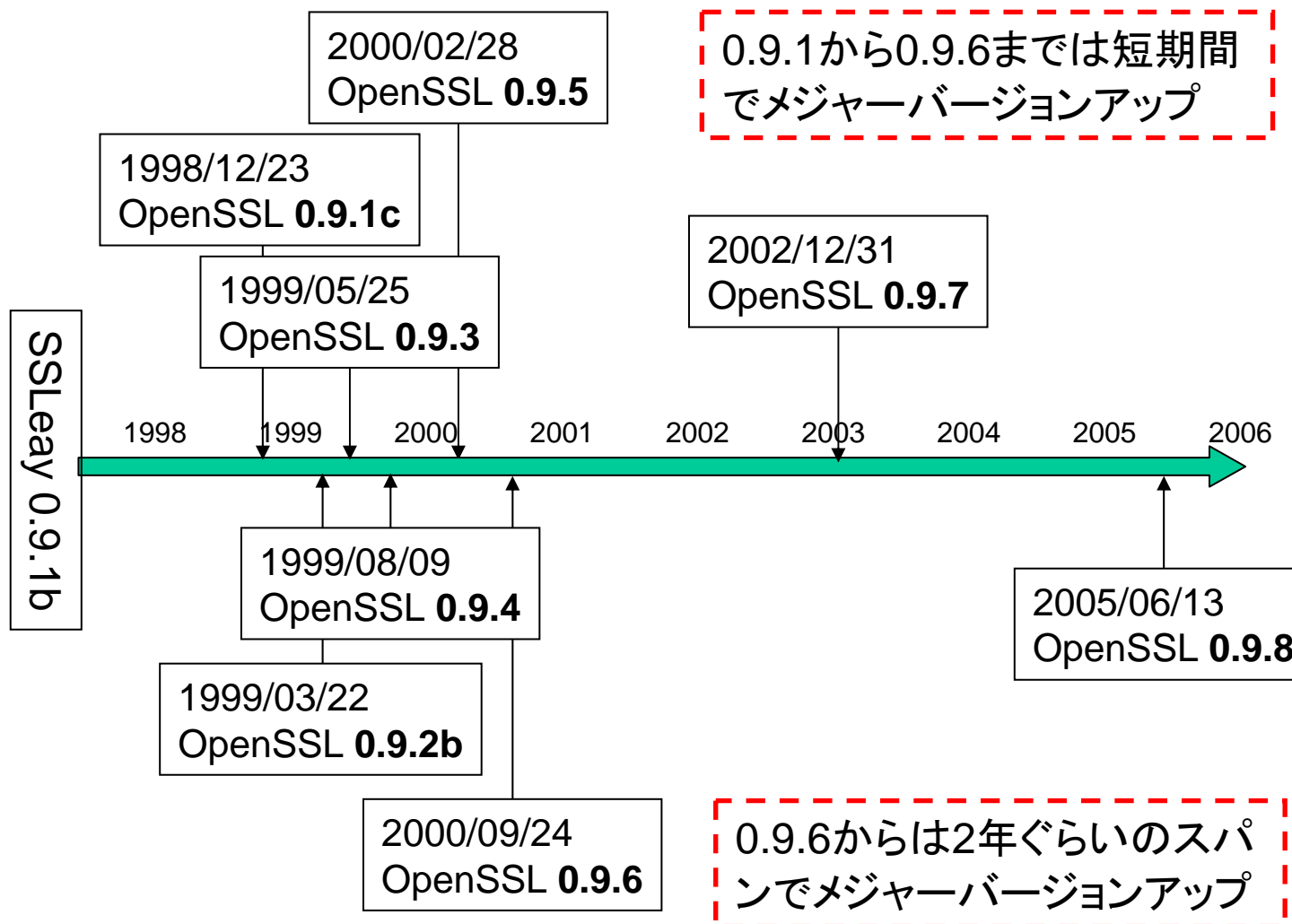
# ツールキット

- PKI関連アプリケーション/プロトコルの開発環境
  - PKI関連の基本機能
  - SSL/TLSスタックの基本機能
  - ASN.1テンプレートよりソースコードの生成

# OpenSSLの特徴

- 複数のプラットフォームをサポート
  - UNIX系
    - Linux, \*BSD\*, MacOS
  - Windows系
    - Windows **CE**/3.1/95/NT/2000/XP ...
  - VMS
  - Netware

# OpenSSL年表



# OpenSSLのソース構造(概略)

- ソースコードツリーの概略
  - プラットフォームごとのディレクトリ
    - ms/MacOS/Netware/VMS/os2
  - テスト関連のディレクトリ
    - test/demos/times
  - 機能単位のディレクトリ
    - ssl/crypto/apps/demos/include/bugs/doc/tool/certs/shlib/util

# プラットフォームごとのディレクトリ

- プラットフォームごとの環境の差異を吸収
  - 作成用のスクリプト
  - 差異を吸収するためのコード
  - プラットフォームごとのテストスクリプト

# テスト関連

- Test
  - テスト用のIV
  - テスト用の証明書
  - テストスクリプト
- Demos
  - デモ用の環境
    - Tiny Client
    - Tiny Server
- Times
  - 処理速度計測用のプログラム
  - 処理速度計測用のスクリプト
  - 処理速度計測用のデータ

# 機能単位のディレクトリ

- ssl
  - SSL/TLSのプロトコルスタックの実装
- crypto
  - 暗号関連の実装
  - BIO
  - EVP
  - 乱数発生器を含む
- apps
  - OpenSSLのアプリケーション(openssl)の実装
- engines
  - Hardware Accelerator 系のコード
- include
  - include
- bugs
  - バグ
  - プラットフォームによるバグの修正プログラム



# 機能単位のディレクトリ

- doc
  - ドキュメンテーション
- tools
  - (主に)証明書関連の処理を助けるためのプログラム群
- certs
  - 著名なCAの証明書
- shlib
  - Shared Libraryのための処理プログラム
- util
  - OpenSSLの作成に必要な処理プログラム群

# sslディレクトリ

- SSLv2 / v3 / TLS 1.0 / DTLSを実装



# SSLの実装状況(その1)

- DTLSの実装
  - Datagram TLSをUDP上で実装
- Kerberos5認証(RFC 2712)
- 圧縮(RFC 3749)

# SSLの実装状況(その2)

- 利用可能暗号
  - 公開鍵系
    - RSA
    - DSA/DSS
    - ECDSA/ECDH(RFC 4492 ただしI-D 12ベースの模様)
  - 共通鍵系
    - DES/Triple DES
    - IDEA
    - RC2/RC4
    - AES (128/192/256) (RFC 3268)
  - 残念ながら Camelliaのサポートはない

# SSLの実装状況(その3)

- 利用可能ハッシュ
  - MDC2
  - MD2/MD5
  - SHA-1/SHA-2
- Pre-Shared Secret(RFC 4492)

# cryptoディレクトリ

- 暗号関連の実装
  - 各種暗号の実装
  - BIO
  - EVP
  - 乱数発生器を含む



# 類似ソフトウェア(その1)

- GNU TLS
  - TLS 1.1/1.0 SSL v3.0をサポート
    - SSL v2は安全でないためサポート対象からはずしている
  - 最新バージョン: 1.4.0 (2006/05/15リリース)
  - Gnu PGにて利用されている
  - <http://www.gnu.org/software/gnutls/>
- NSS (Network Security Services)
  - SSL v2.0/v3.0 TLS 1.0をサポート
    - PKCS#5,#7,#11,#12もサポート
  - 最新バージョン: 3.10 (2005/5/19 リリース)
  - Mozilla (Firefox / Thunderbird)にて利用されている
  - <http://www.mozilla.org/projects/security/pki/nss/>

## 類似ソフトウェア(その2)

- JSSE (Java Secure Socket Extension)
  - SSL v2/V3 TLS 1.0をサポート
  - <http://java.sun.com/j2se/1.5.0/docs/guide/security/jsse/JSSERefGuide.html>
- Microsoft Crypto API



# OpenSSLはどう使われているか？



# OpenSSLはどう使われているのか？

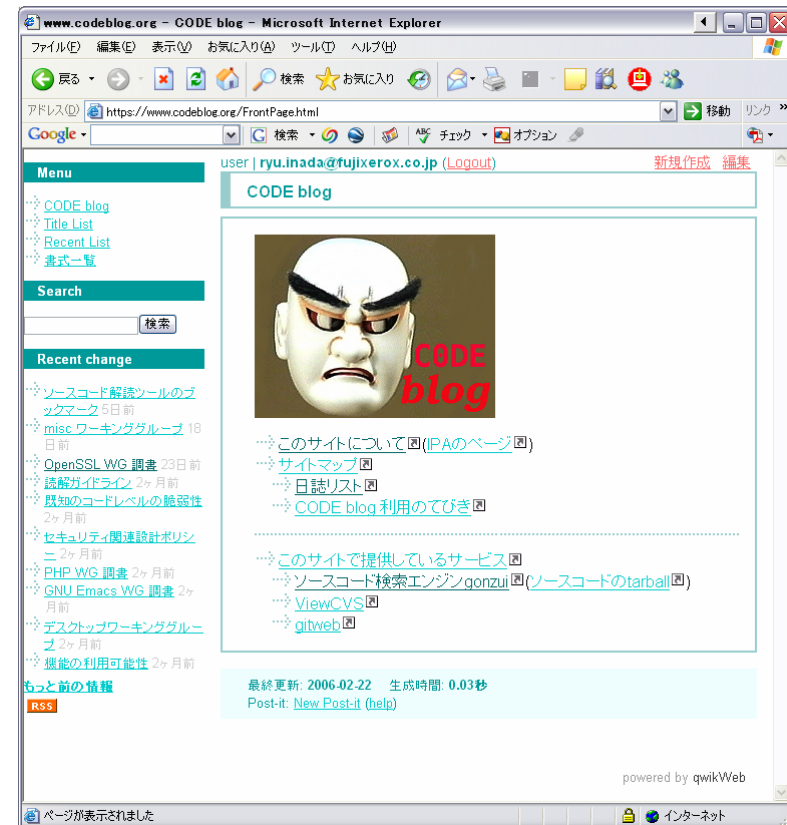
- SSL/TLSのプロトコルスタックとして
  - いまどき、SSLを使わないプロトコルなんて  
.....
  - HTTP
    - Apache, IBM, Oracle .....
  - SMTP
    - Sendmail, Postfix, qmail .....
  - POP/IMAP
    - Courier-imap, cygnus .....
  - Telnet
    - stelnet
  - 汎用
    - Stunnel

# OpenSSLはどう使われているのか？

- S/MIMEのエンジンとしても.....
  - OpenSSLのPKCS#7スタックを利用
- 暗号エンジンとして.....
  - Libcrypto
    - Linuxのデフォルトの暗号エンジン
    - FreeBSD/OpenBSD/NetBSD
  - OpenSSH
- EAP-TLS (IEEE 802.1Xなど)
  - Radiusサーバの実装(FreeRadiusなど)

# Codeblogの紹介

- 独立行政法人 情報処理推進機構 セキュリティセンター 情報セキュリティ技術ラボラトリー のプロジェクト
  - <https://www.codeblog.org/about.html>
- OSSの品質保証の一観点として
  - ソースコードを読むスキルの整理と発見

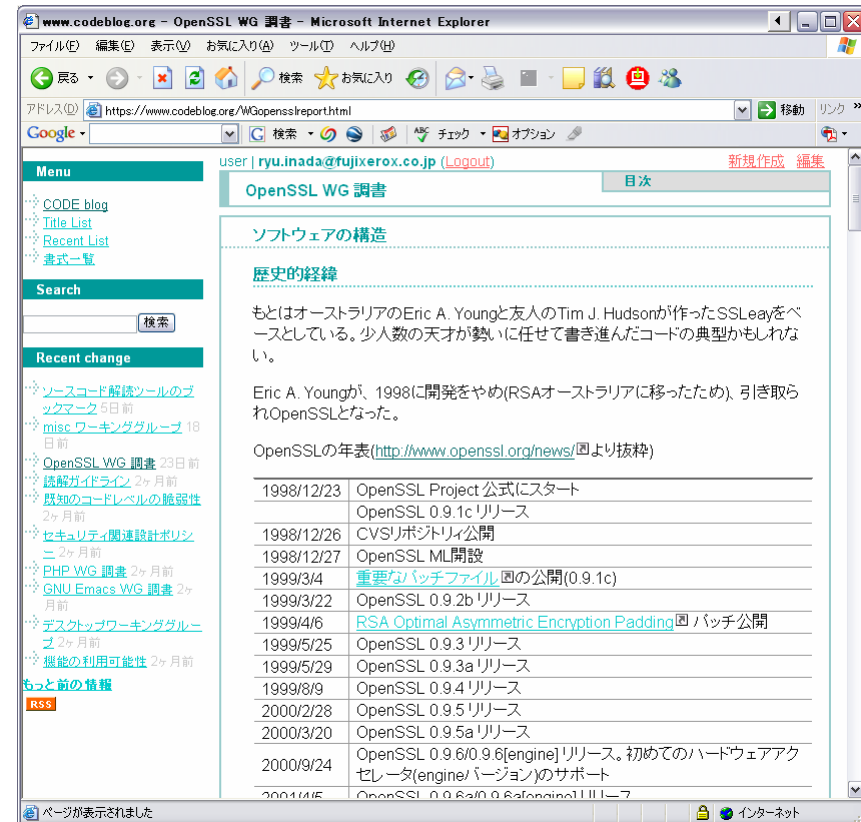


# OpenSSL WG

- 7名のメンバにより  
構成

–機能単位で分割  
–手が足りない

- 今年度もやります  
–メンバ募集中



# OpenSSLの良い点

- デファクト!
  - 実勢に合わせた実装を提供
    - ……というより事実上、広く利用されているためOpenSSLが実勢になっている(苦笑)。
- BIOの設計
  - 設計は古いが**実用的**
  - IOをうまく仮想化しSSL/TLSをストリームベースでのコード実行を可能としている
- 複数のプラットフォームでそこそこの機能の実装
  - 多くのプラットフォームでそこそこの機能を利用可能
  - Windows **CE**でも動く
- 簡易なプログラム作成支援機能
  - PKI関連アプリ/プロトコルのプログラムを容易に作成する機能
    - ASN.1 テンプレート機能の実装
      - <https://www.codeblog.org/blog/kiyoshi/20060203.html>
      - <https://www.codeblog.org/blog/kiyoshi/20060204.html>
      - <https://www.codeblog.org/blog/kiyoshi/20060205.html>
      - <https://www.codeblog.org/blog/kiyoshi/20060219.html>

# OpenSSLの悪い点

- 機能面での不足
  - CRLの扱いの不備
    - 結構、致命的
  - サンプル実装とはいえopensslコマンドの証明書発行機能はまずい
    - シリアル番号のデータベース不在
- ソースコード上の問題
  - アドホックな改良を加えられているため全体的にソースコードが汚い
    - 随所にObject Orientedに書こうとして失敗した形跡が.....
    - S/MIMEに関してはMIME Body Partの解析がよろしくない
  - 高機能なソースコード生成能力を持たせているために難解
  - メモリリークを起こしやすい
    - 正しい作法にのっとりたコーディングを要求する
  - したがって、初心者に敷居が高い

# CRLの処理の問題

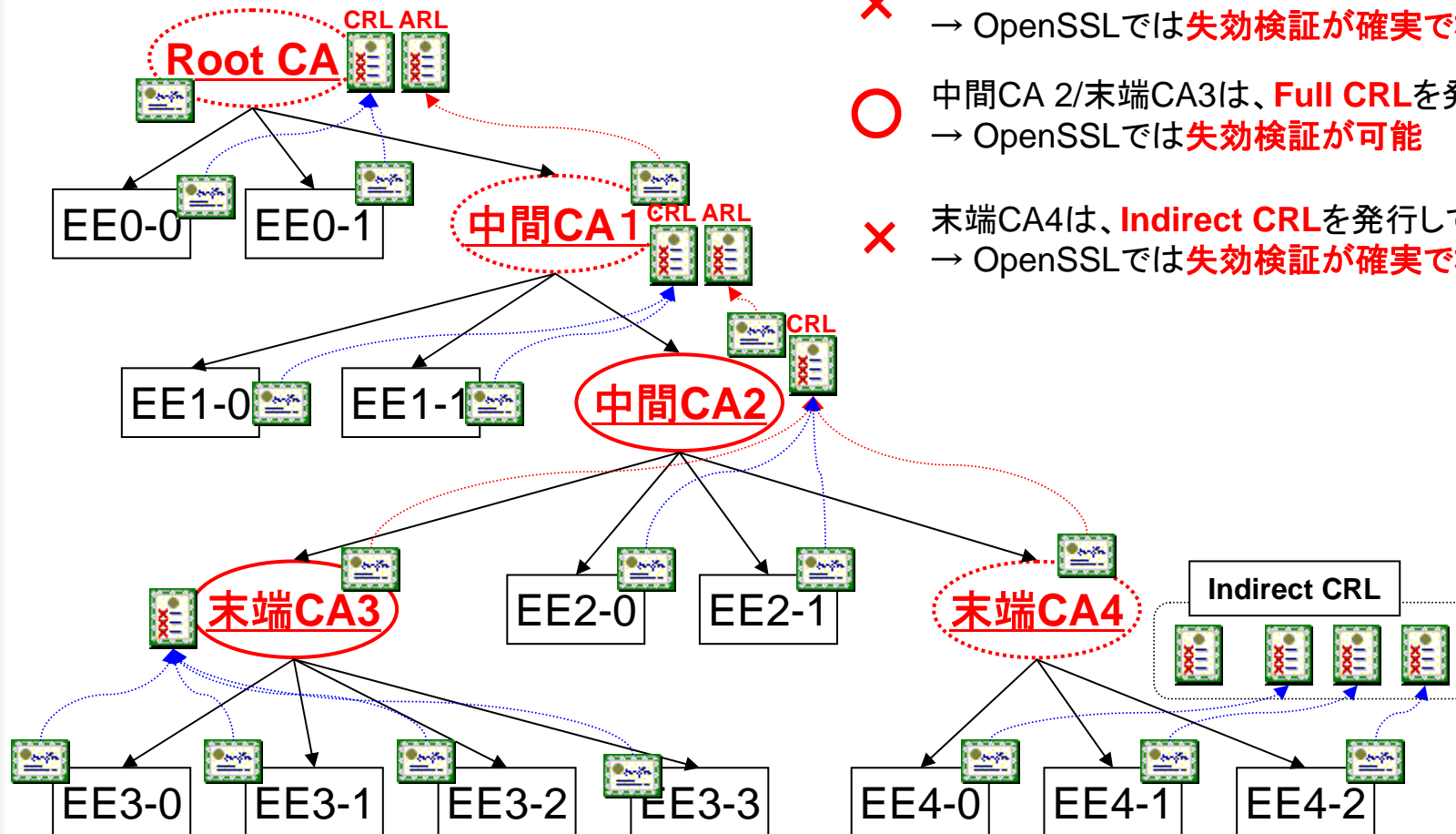
- OpenSSLの前提
  - CAはCRLを一つしか発行しない
    - Issuing Distribution Pointの処理はない
    - CRLの拡張属性に関しての処理はほとんど出来ない
  - 信頼関係は単純な木構造
    - 大半のPKIは現行ではそうだけど.....
      - 政府認証基盤(GPKI)関連で利用するときは要注意
    - トラストアンカーを全て持っていればいいのかも出来ないけど.....



# CAはCRLを複数発行する

- 全てのCAがFull CRLを発行しているのではない
  - 複数階層の信頼構造を持つCAは
    - ARL (Authority Revocation List)
      - CAに対して発行した証明書の失効リスト
    - CRL (Certificate Revocation List)
      - EEに対して発行した証明書の失効リスト
  - を発行している場合が多い
    - CRLのサイズを小さくするための工夫
  - 今後、増えると考えざるを得ない
- Delta CRLを発行している場合もある
  - まだ数は少ないがMicrosoft Windowsではサポート済み
- 下位のインターフェイスを用いて自前で作ることも可能ではあるが  
.....
  - PKIの証明書失効検証をきちんと理解することが前提となる

# ARLとCRLとIndirect CRL



- ✗ Root CA/中間CA 1は、**CRL/ARL**を発行している  
→ OpenSSLでは**失効検証が確実でない**
- 中間CA 2/末端CA3は、**Full CRL**を発行している  
→ OpenSSLでは**失効検証が可能**
- ✗ 末端CA4は、**Indirect CRL**を発行している  
→ OpenSSLでは**失効検証が確実でない**

# Issuing Distribution Pointとは?

## •CRLの素性

- どのCAから発行しているのか?
- どこで公開されていたものなのか?

## •CRLの性質

- どういう種類の証明書の失効情報が登録されているのか?
- CRLに全ての失効情報が載っているのか?

## •Indirect CRL

- CRLの中にCRLへのポインタが埋め込まれている形式のCRL

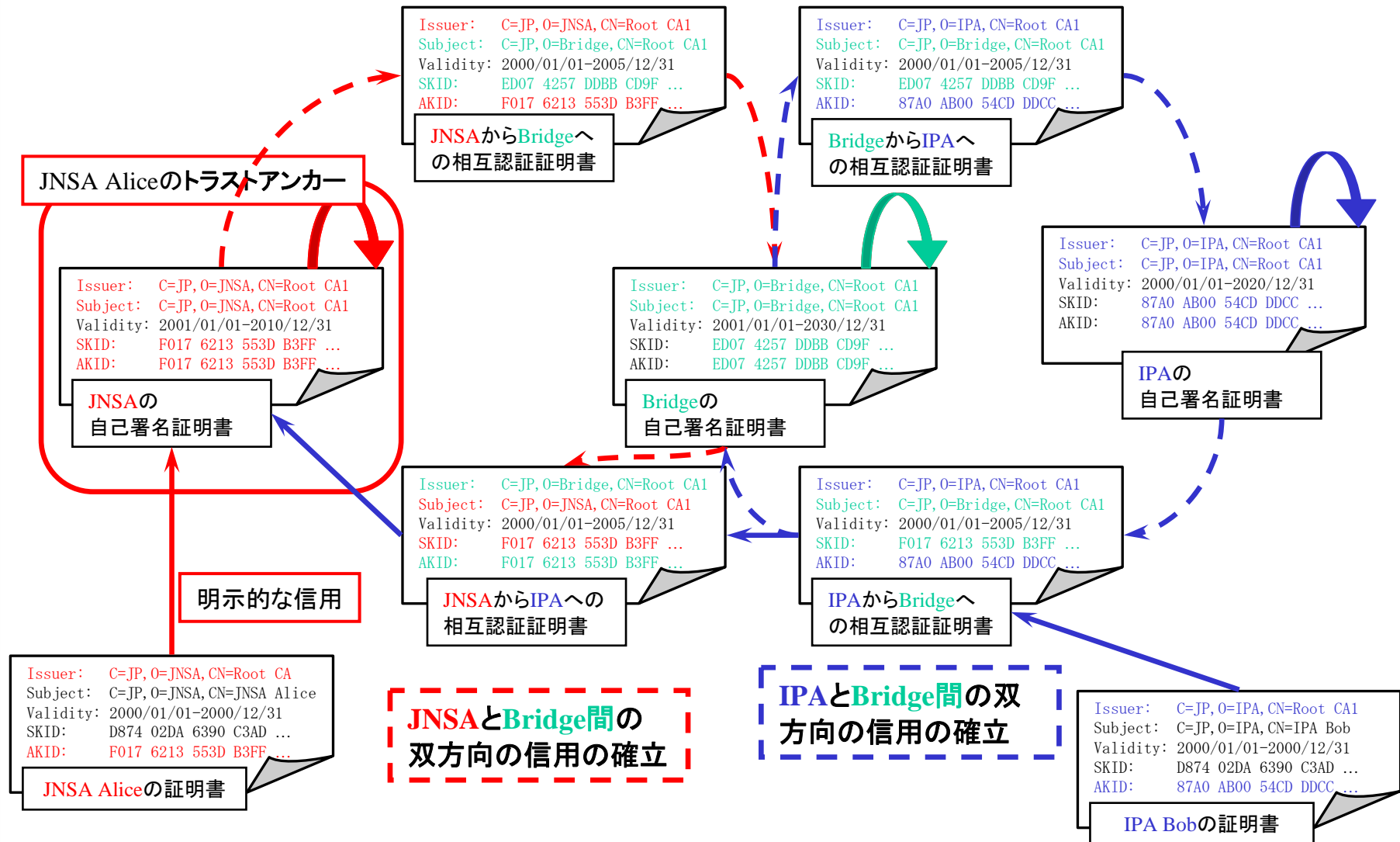
```
issuingDistributionPoint ::= SEQUENCE {  
  distributionPoint [0] DistributionPointName OPTIONAL,  
  onlyContainsUserCerts [1] BOOLEAN DEFAULT FALSE,  
  onlyContainsCACerts [2] BOOLEAN DEFAULT FALSE,  
  onlySomeReasons [3] ReasonFlags OPTIONAL,  
  indirectCRL [4] BOOLEAN DEFAULT FALSE,  
  onlyContainsAttributeCerts [5] BOOLEAN DEFAULT FALSE }
```

- IDPが解釈できないのでは分割CRL系は対処できない!
  - そもそも分割系のCRLであることがわからない
  - Delta CRLも処理は出来ない

# OpenSSLのCRLの処理の概略

1. 証明書チェーン作成
    - 深さ優先探索 & バックトラック無し
  2. ローカルにある(信用できる)証明書に到達できたか確認
  3. 拡張の検証(check\_chain\_extensions)
  4. トラストアンカーへ到達したか確認
  5. RevocationCheck
  6. 証明書チェーン検証もしくはユーザ定義関数の呼び出し
- 単純な木構造のPKIモデルを前提にしている
    - 複雑なパス構築・検証はやりようがない
      - 事前に最適化したパスを作成し、検証に必要な証明書のみを提示し検証するのが精一杯

# こんなのは無理！



# 証明書の検証

- 本来は.....
  - RFC 3280/4158 のパス構築・検証を行うべきである.....
  - 詳細は
    - <http://www.ipa.go.jp/security/rfc/RFC3280-00EN.html>
    - <http://www.ipa.go.jp/security/rfc/RFC4158JA.html>

# まとめ



# まとめ

- OpenSSLは.....
  - 証明書の失効検証をする際には役に立たない状況があることを認識する
  - 単なる単純なSSL/TLSスタックとして利用することは問題ない



# 今後

- アプローチ 1.
  - OpenSSLの失効検証をまともにする
    - 複雑でメンテナンスしがたいコードとの格闘
    - OpenSSLチームとのネゴ
- アプローチ 2.
  - OpenSSLの失効検証部を拡張
    - サーバサイドでの失効確認サーバを使えるようにする
    - SCVPなどが候補?
- アプローチ 3.
  - OpenSSLプラグコンパチのソフトウェアの開発?
    - 新規コードできれいに作りなおせる可能性あり
    - GNU TLSにはBIO APIコードもある

THE DOCUMENT COMPANY  
**FUJI XEROX**



# 商標など

- Microsoft、MS、Windows、Windows 2000、Windows NT、Windows XP、Windowsロゴ、Internet Explorer、Outlook、Outlook Expressなどは、米国Microsoft Corporationの米国およびその他の国における登録商標または商標である。
- Sun Microsystems、Sunロゴ、Java コーヒーカップロゴ、Solaris、Java、JDKなどは、米国Sun Microsystemsの米国およびその他の国における登録商標または商標である。
- その他、本文に記載されている会社名、商品名、製品名などは、一般に各社の商標または登録商標である。
- 本書では、™、©、®などを記載しない

# 参考文献(SSL/TLS)

- SSL
  - Kipp E.B. Hickman, The SSL Protocol
    - [http://www.netscape.com/eng/security/SSL\\_2.html](http://www.netscape.com/eng/security/SSL_2.html)
  - Alan O. Freier, Philip Karlton, Paul C. Kocher, The SSL Protocol Version 3.0
    - <http://wp.netscape.com/eng/ssl3/draft302.txt>
- TLS
  - T. Dierks, C. Akken, RFC 2246: The TLS Protocol Version 1.0
  - A. Medvinsky, M. Hur, RFC 2712: Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)
  - P. Chown, RFC 3268: Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)
  - S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, T. Wright, RFC 3546: Transport Layer Security (TLS) Extensions
  - S. Hollenbeck, RFC 3749: Transport Layer Security Protocol Compression Methods
  - S. Moriai, A. Kato, M. Kanda, RFC 4132: Addition of Camellia Cipher Suites to Transport Layer Security (TLS)
  - P. Erone, Ed., H. Tschofenig, Ed. , RFC 4279: Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)
  - T. Dierks, E. Rescorla, RFC 4346: The Transport Layer Security (TLS) Protocol Version 1.1
  - S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, T. Wright, RFC 4366: Transport Layer Security (TLS) Extensions
  - S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk B. Moeller, RFC 4492: Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)

# 参考文献(PKI関連)

- ITU-T Recommendation X.509 | ISO/IEC 9594-8: Information technology - Open Systems Interconnection - The Directory: Authentication framework. , 1997
- R. Housley, W. Ford, W. Polk, and D. Solo, RFC 2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile, 1999
  - <http://www.ietf.org/rfc/rfc2459.txt>
- M. Myers, R. Ankney, A. Malpani, S. Galperin and C. Adams , RFC 2560: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, 1999
  - <http://www.ietf.org/rfc/2560.txt>
- R. Housley, W. Polk, W. Ford, D. Solo, RFC 3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile . 2002
  - <http://www.ietf.org/rfc/rfc3280.txt>
- M. Cooper, Y. Dzambasow, P. Hesse, S. Joseph and R. Nicholas, RFC 4158: Internet X.509 Public Key Infrastructure: Certification Path Building, 2005
  - <http://www.ietf.org/rfc/rfc4158.txt>

# 参考文献(報告書関連、書籍)

- 報告書関連
  - IPA, PKI 関連技術情報, 2004-2005
    - <http://www.ipa.go.jp/security/pki/pki.html>
  - IPA, インターネットセキュリティに関するRFCの日本語訳
    - <http://www.ipa.go.jp/security/rfc/RFC.html>
  - IPA/JNSA, 電子政府情報セキュリティ相互運用支援技術の開発 GPKI 相互運用フレームワーク, 2004
    - <http://www.ipa.go.jp/security/fy14/development/pki/interop.html>
- 書籍
  - John Viega, Matt Messier, Pravir Chandra, 齋藤孝道監訳, OpenSSL 暗号・PKI・SSL/TLSライブラリの詳細, ISBN 4274065731, オーム社, 2004, 436p
  - 小松 文子, PKIハンドブック, ISBN 4883732053, ソフトリサーチセンター, 2004, 255p
  - 日本ネットワークセキュリティ協会, 情報セキュリティプロフェッショナル総合教科書, ISBN 479800880X, 秀和システム, 2005, 575p
  - 青木 隆一, 稲田 龍, PKIと電子社会のセキュリティ, ISBN 4320120280, 共立出版, 2001, 233p

# URLs

- OpenSSL Project, OpenSSL.org
  - <http://www.openssl.org/>
- IETF, TLS-WG
  - <http://www.ietf.org/html.charters/tls-charter.html>
- GNU TLS
  - <http://www.gnu.org/software/gnutls/>
- NSS (Network Security Services)
  - <http://www.mozilla.org/projects/security/pki/nss/>
- JSSE (Java Secure Socket Extension)
  - <http://java.sun.com/j2se/1.5.0/docs/guide/security/jsse/JSSERefGuide.html>
- Microsoft Crypto API
  - <http://www.microsoft.com/japan/windows2000/techinfo/planning/security/pki.asp>
- Codeblogプロジェクト
  - <https://www.codeblog.org/>
- IPA セキュリティセンター
  - <http://www.ipa.go.jp/security/>

THE DOCUMENT COMPANY  
FUJI XEROX

ご清聴ありがとうございました

