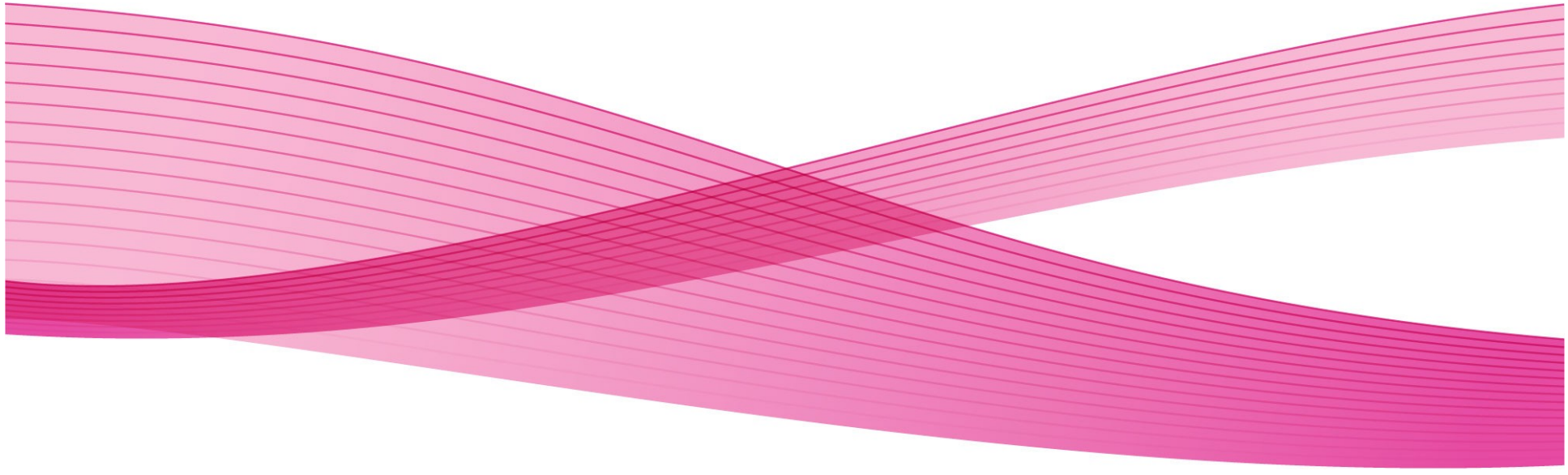
OpenSSL 1.0.0 のリリースについて

NPO日本ネットワークセキュリティ協会：JNSA PKI Day 2010
2010年6月29日(木) 12:20～13:00



富士ゼロックス株式会社
漆嶌　賢二

FUJI Xerox

今日のアジェンダ

- OpenSSL とは
- OpenSSL 1.0.0の主要な変更点
  - (1) 暗号アルゴリズム
  - (2) SSL/TSL CipherSuite
  - (3) RFC 3280証明書パス検証のフルサポート
  - (4) openssl cms – CMS署名暗号フォーマット
  - (5) opnessl ts – RFC3161 タイムスタンプ
  - (6) 公開鍵API、コマンドの共通化
  - (7) 証明書のハッシュファイル名生成ルールの変更
- その他関連情報
- まとめ

FUJI Xerox

OpenSSLとは

OpenSSLとは

- SSL/TLSや暗号のためのセキュリティライブラリ+ツール
- 歴史
  - 1995年にApacheをHTTPS対応にするためのライブラリSSLeayが前身
  - 1998年12月にOpenSSL 0.9.1cとしてリリース
  - 0.9.6あたりからバージョンが細かく更新される
  - 1.0.0 beta1, 2, 3, 4, 5でようやく1.0.0が正式リリース
- フリーでオープンソース (BSDライセンスに類似)
- 頑強
- 数多くの製品に組み込まれている
- ApacheをHTTPSで使う際のデフォルトのモジュール
- 多様な機能

OpenSSLを構成するコンポーネント

商用/非商用の製品への組み込み　RubyやPerlのライブラリ　Apache mod_ssl

暗号関連コマンドラインツール　暗号関連APIライブラリ

SSL/TLSコンポーネント

テスト用 s_server　テスト用 s_client

暗号データフォーマット コンポーネント

RFC 3161 タイムスタンプ

S/MIME 署名暗号メール形式

CMS/PKCS#7 署名暗号データ形式

鍵・証明書フォーマット
PKCS#5(1)
PKCS#7
PKCS#8
PKCS#12
PVK

証明書関連コンポーネント

パス検証

OCSP

証明書発行要求(PKCS#10)

認証局(CA)

X.509 証明書・CRL

SSL/TLS

CipherSuites

認証・鍵交換　暗号化　MAC

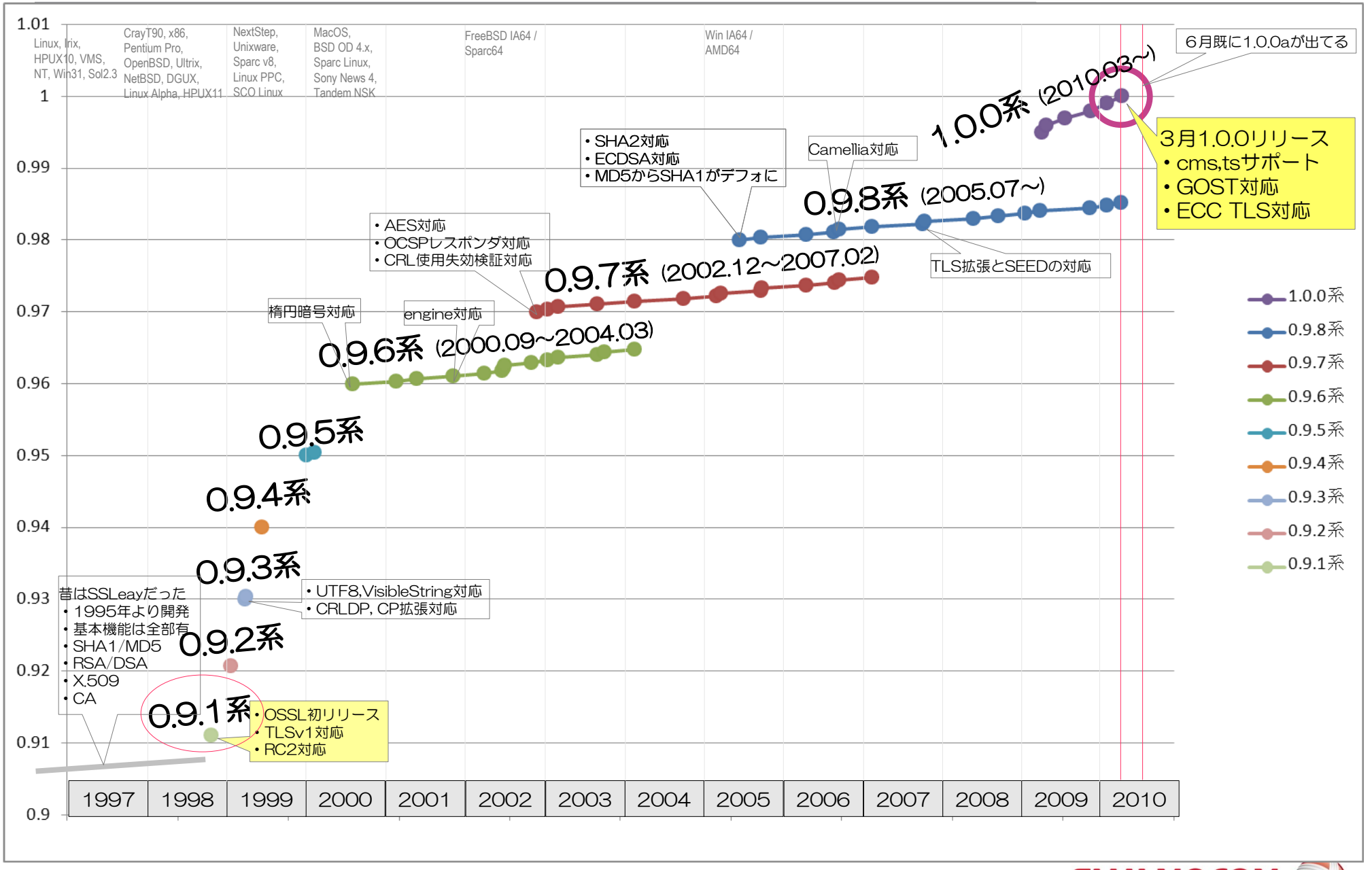
ASN.1　公開鍵暗号　ハッシュ関数　共通鍵暗号　セキュア乱数

基本コンポーネント

Engine (PKCS#11,MS CAPI SmartCardやHSMとのI/F)

FUJI Xerox

OpenSSL 1.0.0 リリースまでの軌跡



FUJI Xerox

主要な変更点 (1/7)

暗号アルゴリズムの対応について

暗号アルゴリズムのデフォルト設定の変更点

	0.9.8nまで	共通	1.0.0より
公開鍵暗号と応用		DH, RSA, DSA, ECDH, ECDSA	GOST(※1)
共通鍵暗号	なし	AES, Blowfish, CAST, DES, 3DES, DESX, IDEA, RC2, RC4	Camellia, GOST, SEED, GOST(※2)
メッセージ認証		HMAC	(1.1.0より CMAC) GOST-MAC
ハッシュ関数	MD2	MD4, MD5, MDC2, RIPEMD160, SHA, SHA1, SHA{224, 256, 384, 512}	Whirlpool GOST94

デフォルトで利用可能に

デフォルトで利用不可に

※青字はengineの設定が必要
※1：公開鍵(GOST94, GOST94CP) 公開鍵楕円(GOST2001, GOST2001CP)
※2：共通鍵(GOST89)

FUJI Xerox

主要な変更点 (2/7)

SSL/TLS機能について

OpenSSLのSSL/TLS機能

■代表的な利用方法

- Apacheのmod_sslモジュール
- テスト用としてs_server、s_clientコマンドにより

■サポート状況（ニュースリリース、ML等により）

- SSLv2（脆弱性が知られており多くのケースでデフォルトOFF）
- SSLv3をサポート
- TLSv1.0をサポート
- TLSv1.1（AES,KRB）はOpenSSL v1.1.0で最初の試験的リリース
- TLSv1.2（AES-GCM,SHA2）については現状、その予定が無い
- mod_ssl はapache 2.3.5 alphaでOpenSSL 1.0.0サポート
- mod_ssl でGOSTを使うには非公式パッチが必要

FUJI Xerox

SSL/TLSのCipherSuiteとは？

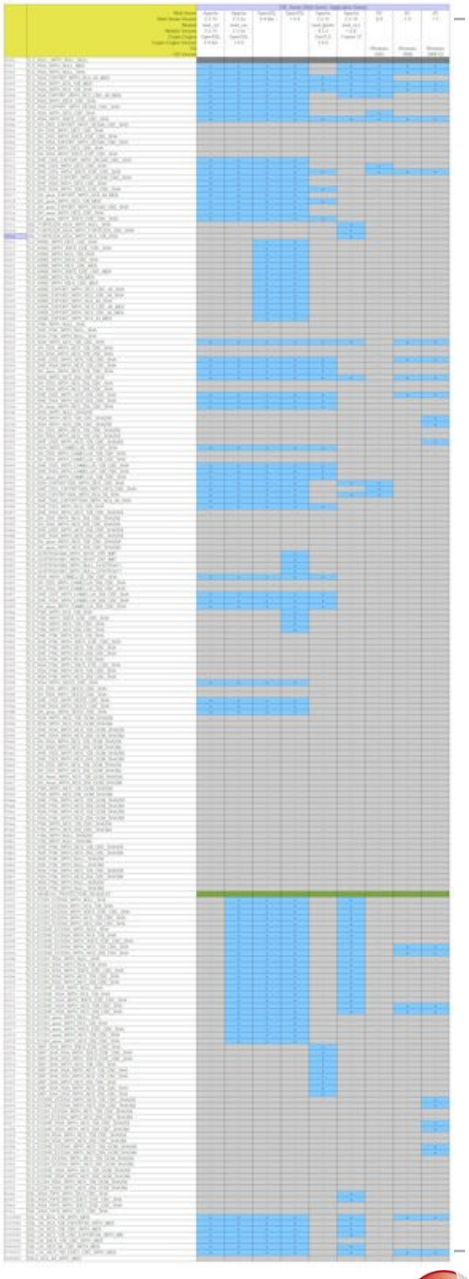
- SSL/TLSは4つのアルゴリズムを組み合わせて機密通信
  - 鍵交換：DH, DHE, ECDH等
  - 認証：RSA, DSS, PSK等　　鍵交換と認証を一緒にすることもあります
  - 共通鍵暗号化：DES, 3DES, AES, Camellia等
  - メッセージ認証(MAC)：MD5, SHA1, SHA2等
- これらのこれらの組み合わせに名前とIDをつけたものを CipherSuite(一式の暗号)と呼んでいます。
  - (例)
  - TLS_[鍵交換]_[認証]_WITH_[暗号化]_[MAC]
  - TLS_DHE_RSA_WITH_DES_CBC_SHA
- 暗号通信を始める前にサーバーとクライアントはこれを交換
  - ブラウザ側：僕はこんな８つのCipherSuiteなら大丈夫だよ
  - サーバー側：それじゃ、その中のコレ使うよ

OpenSSL 1.0.0関連のCipherSuiteの調査

■以下のサーバーを比較調査

- Apache+mod_ssl (OpenSSL1.0.0)
- Apache+mod_ssl (OpenSSL0.9.8m)
- Apache+mod_gnutls (GnuTLS)
- Apache+mod_nss (NSS)
- OpenSSL 1.0.0 s_serverコマンド
- OpenSSL 0.9.8m s_serverコマンド
- IIS(Win2003/2008/8R2)

CipherSuiteはRFCやInternet Draftにあるだけで、大体 220 ぐらいあります



FUJI Xerox

OpenSSL 1.0.0他のCipherSuiteのサポート状況

機能	TLSサーバー	Apache 2.2.15 mod_ssl 2.2.15 OpenSSL 0.9.8m	Apache 2.3.5alpha mod_ssl 2.3.5alpha OpenSSL 1.0.0	OpenSSL 0.9.8m s_server	OpenSSL 1.0.0 s_server	Apache 2.2.15 mod_gnutls 0.5.5 GnuTLS 2.8.6	Apache 2.2.14 mod_nss 1.0.8 Mozilla NSS	IIS 6.0 Windows 2003	IIS 7.0 Windows 2008	IIS 7.5 Windows 2008 R2
SSL/TLS	SSL 2.0	◎	◎	◎	◎	×	○	△	△	△
	SSL 3.0	◎	◎	◎	◎	◎	◎	◎	◎	◎
	TLS 1.0	◎	◎	◎	◎	◎	◎	◎	◎	◎
	TLS 1.1	×	×	×	×	◎	×	×	◎	◎
	TLS 1.2	×	×	×	×	×	×	×	×	◎
鍵交換	RSA	◎	◎	◎	◎	◎	◎	◎	◎	◎
	RSA_EXPORT	◎	◎	◎	◎	◎	◎	○	×	×
	RSA_EXOPRT1024	◎	◎	◎	◎	×	△	○	×	×
	RSA_FIPS	×	×	×	×	×	○	×	×	×
	DH	△	△	△	△	△	△	×	×	×
	DHE	○	○	○	○	○	○	△	△	△
	FORTEZZA	×	×	×	×	×	◎	×	×	×
	Kerberos5	×	×	◎	◎	×	×	×	×	×
	GOST	×	× (※1)	×	○	×	×	×	×	×
	PSK	×	×	×	×	×	×	×	×	×
	ECDH/ECDSA	×	◎	◎	◎	×	○	×	○	○
	SRP SHA1	×	×	×	×	◎	×	×	×	×
認証	RSA	◎	◎	◎	◎	◎	◎	◎	◎	◎
	RSA_EXPORT	◎	◎	◎	◎	◎	◎	○	×	×
	RSA_EXOPRT1024	◎	◎	◎	◎	×	△	○	×	×
	RSA_FIPS	×	×	×	×	×	○	×	×	×
	FORTEZZA-KEA	×	×	×	×	×	◎	×	×	×
	Kerberos5	×	×	◎	◎	×	×	×	×	×
	GOST	×	× (※1)	×	○	×	×	×	×	×
	PSK	×	×	×	×	×	×	×	×	×
	ECDH/ECDSA	×	◎	◎	◎	×	○	×	○	○
	None	○	○	○	○	×	×	×	×	×
暗号化	RC2 40bit	○	○	○	○	×	○	○	×	×
	RC4 40bit	○	○	○	○	○	○	○	×	×
	RC4 128bit	○	○	○	○	○	○	○	○	○
	IDEA 128bit	○	○	○	○	×	×	×	×	×
	DES 40bit	○	○	○	○	×	×	○	×	×
	DES 56bit	○	○	○	○	×	×	○	×	×
	3DES 168bit	○	○	○	○	×	×	○	○	○
	FORTEZZA	×	×	×	×	×	◎	×	×	×
	AES 128bit	△	△	△	△	△	△	×	△	△
	AES 256bit	△	△	△	△	△	△	×	△	△
	AES 128bit GCM	×	×	×	×	×	×	×	×	△
	AES 256bit GCM	×	×	×	×	×	×	×	×	△
	Camellia 128bit	○	○	○	○	○	×	×	×	×
	Camellia 256bit	○	○	○	○	○	×	×	×	×
	GOST	×	× (※1)	×	○	×	×	×	×	×
	SEED 128bit	○	○	○	○	×	×	×	×	×
	None	○	○	○	○	×	○	○	○	○
MAC	MD5	○	○	○	○	○	△	○	○	○
	SHA1	○	○	○	○	○	○	○	○	○
	SHA256	×	×	×	×	×	×	×	×	△
	SHA384	×	×	×	×	×	×	×	×	△
	GOST	×	× (※1)	×	○	×	×	×	×	×

IISはTLS 1.0>1.1>1.2 のサポート最速

楕円

Camellia

脆弱なアルゴリズムを排除

EC,AES-GCM,SHA2による強固なSuite-B対応

※1：標準リリースにはGOSTの対応は無いが
2.2.6への非公式のOpenSSL 1.0.0 GOST対応パッチがある

※注意：×印であるからといって、機能が劣るわけではない。
セキュリティ上の理由からサポートを外している場合もある。

凡例

◎	このアルゴリズムを含むCipherSuitesを全てサポート
○	このアルゴリズムを含むCipherSuitesをほぼサポート
△	このアルゴリズムを含むCipherSuiitesを一部サポート
×	このアルゴリズムを含むCipherSuitesをサポートしない

FUJI Xerox

(参考)お時間あるときご覧下さい

SSLサーバーのCipherSuiteのサポート状況調査結果詳細まとめ

■ かなり広まりつつある

- ECDH/ECDSA (mod_ssl/nss,OpenSSL,IIS7x)
- Camellia (mod_ssl/gnutls, OpenSSL)

■ ユニークなアルゴリズムのサポート

- mod_ssl, OpenSSL → IDEA, SEED
- OpenSSL → Kerberos5, GOST
- mod_nss → RSA_FIPS, FORTEZZA
- mod_gnutls → SRP_SHA
- IIS 7.5 → AES-GCM, SHA256/384

■ オープンソース系では全て非サポート

- AES-GCM, SHA256/384

■ OpenSSLに関連したノート

- OpenSSLが出てmod_sslに反映されるにはタイムラグ有
- OpenSSLでは国家暗号(CAST/カナダ,SEED/韓国,GOST/ロシア,Camellia/日本(国家?))が多く取り込まれている。
- OpenSSLは汎用ライブラリなので暗号の無いもの、脆弱なものまで含んでいるので、CipherSuite設定には最新の注意が必要。デフォルトではやや脆弱

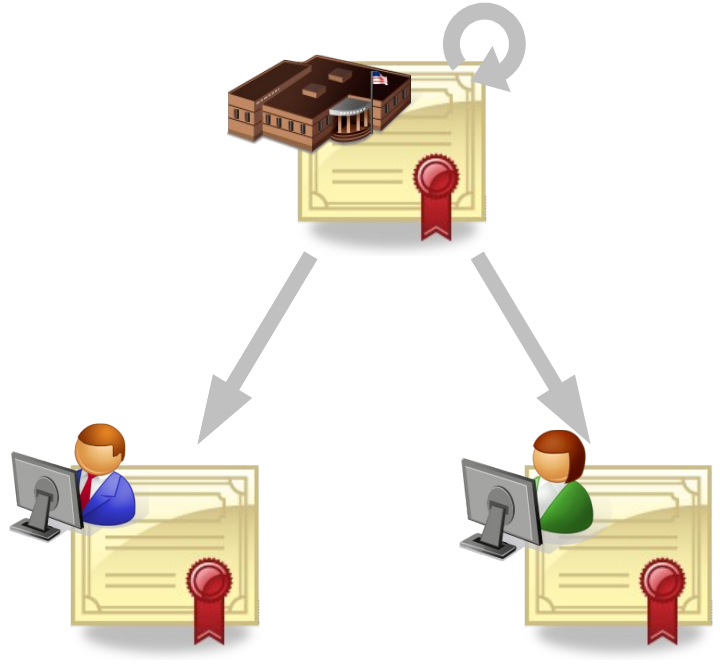
■ 他のウェブサーバーのノート

- mod_gnutlsは脆弱なSSLv2を非サポート
- mod_nss/gnutlsは暗号や認証無などの設定の抜け穴となるCipherSutieをサポートせず、弱い暗号をサポートしない。そのためデフォルト設定でも安心感がある。
- IISはサポートするアルゴリズム総数が限定されているが、6.0、7.0、7.5とバージョンアップでRC2/RC4/DES/RSA_EXPORT廃止、AES/SHA2/EC/AES-GCM追加など順調に次世代アルゴリズムへの移行が進んでいる。それなのに、SSLv2があるのは惜しい。

FUJI Xerox

主要な変更点 (3/7)

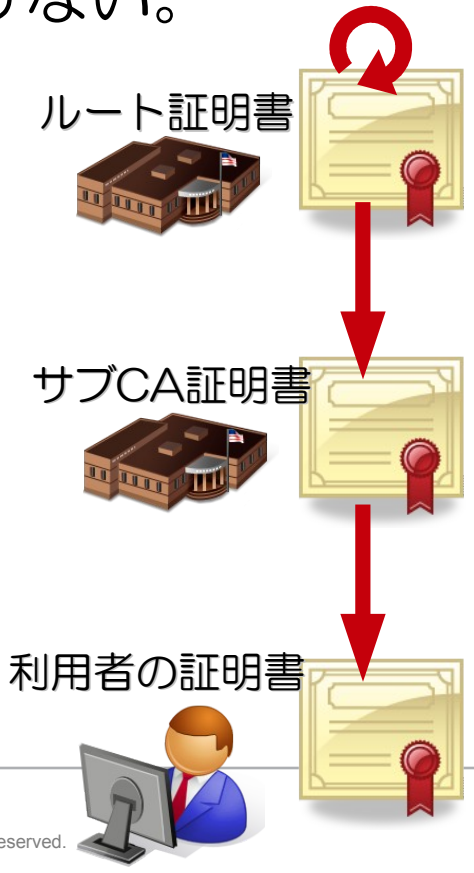
NIST PKITSに対応した RFC 3280証明書パス検証のフルサポート



証明書のパス検証とは？

■信頼するルート証明書から辿って、対象の証明書が有効かどうかを判断すること

■RFC 3280の6章にパス検証アルゴリズムの例が示されている。その通りに実装する必要は無いが、検証結果は同じにならなければいけない。



証明書のパス検証内容

- 署名が繋がってる？
- 名前が繋がっている？
- 有効期限内？
- 失効してない？

- 鍵使用目的は大丈夫？
- それ本当にCA証明書？(基本制約)
- 証明書の段数が範囲内？(パス長制約)
- 名前が許された範囲内？(名前制約)
- 証明書ポリシが繋がる？(ポリシ制約)

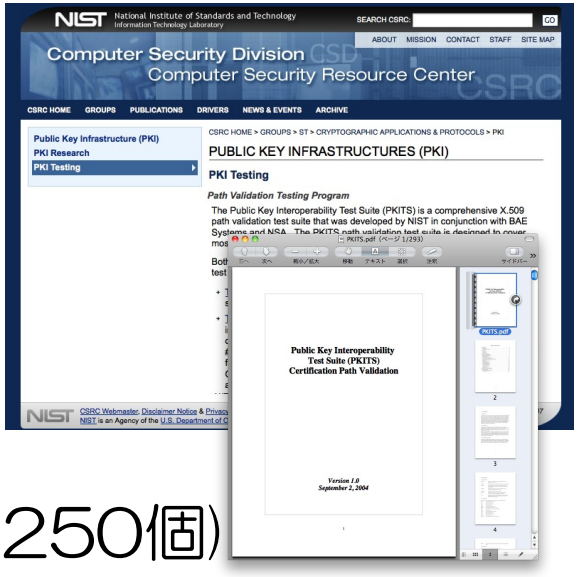
、、、、とか

FUJI Xerox

NIST PKITS(PKIテストスイート)とは？

- 米国 国立標準技術研究所(NIST)でNISTで2004年に開発されたRFC 3280に準拠した証明書パス検証のテストケース集
  - テストケース仕様書
  - データ：証明書, CRL, 署名メール, LDAP
  - テストはオンラインでもオフラインでも可能
  - オンライン失効検証はLDAPのみ

http://csrc.nist.gov/groups/ST/crypto_apps_infra/pki/pkitesting.html



- テストケース (節数は224だが細かく分けると250個)
  - 署名, 有効期限, 名前チェーン, 失効検証, 基本制約, パス制約, ポリシ制約, ポリシマッピング, CDP, 分割/Indirect/Delta CRL等
- 2005年にパス構築のテストスイートも追加された

テストデータに使われている証明書の
有効期限は2011年4月19日までなんですが、みんな大丈夫？

FUJI Xerox

では、OpenSSLのPKITS対応パス検証とは？

どうテストしたの？

1.0.0リリースノートにこんな風に書いてある
「RFC3280 path validation: sufficient to process PKITS tests.」

- テスト用Perlスクリプト“test/pkits-test.pl”
- NISTからPKITS_data.zipをダウンロードしtest/pkitsで解凍
- pkits-test.pl を実行

```
% cd test; perl pkits-test.pl
Running PKITS tests:
4.1 Signature Verification
4.2 Validity Periods
4.3 Verifying Name Chaining
4.4 Basic Certificate Revocation Tests
4.5 Verifying Paths with Self-Issued Certificates
4.6 Verifying Basic Constraints
4.7 Key Usage
4.8 Certificate Policies
4.9 Require Explicit Policy
4.10 Policy Mappings
4.11 Inhibit Policy Mapping
4.12 Inhibit Any Policy
4.13 Name Constraints
4.14 Distribution Points
4.15 Delta-CRLs
4.16 Private Certificate Extensions
All Tests Successful.
```

テスト成功と書いてあるけど、、、

FUJI Xerox

PKITS検証結果の比較
(OpenSSL 1.0.0/0.9.8n smime/cms, Sun JCE)

1.0.0はフルサポート

	Sun Java JCE 6	OpenSSL smime 0.9.8n	OpenSSL smime 1.0.0	OpenSSL cms 1.0.0	PKITS章
発行者と主体者の繋がりを正しく検証できる	○	△	◎	◎	
発行者RSA鍵による署名値を検証できる	◎	◎	◎	◎	4.1
発行者DSA鍵による署名値を検証できる	◎	◎	◎	◎	4.1
発行者/主体者の名前の繋がりを検証できる	◎	○	◎	◎	4.3
鍵更新があった場合でも正しく検証できる	×	×	◎	◎	4.5
有効期限を正しく検証できる	◎	◎	◎	◎	4.2
基本制約を全て正しく処理できる	◎	○	◎	◎	
基本制約(cAフラグ)を正しく処理できる	◎	◎	◎	◎	4.6
基本的なパス長制約を正しく処理できる	◎	◎	◎	◎	4.6
自己署名証明書が含まれた場合のパス長制約を正しく処理できる	◎	×	◎	◎	4.6
CAの証明書やCRL発行のための鍵使用目的拡張を正しく処理できる	◎	◎	◎	◎	4.7
ポリシ制約を全て正しく処理できる	◎	△	◎	◎	
ポリシ制約について基本的な検証ができる	◎	◎	◎	◎	4.8
ポリシ制約のrequireExplicitPolicyを正しく処理できる	◎	×	◎	◎	4.9
ポリシ制約においてポリシマッピングを正しく処理できる	◎	×	◎	◎	4.10
ポリシ制約においてinhibitPolicyMappingを正しく処理できる	◎	×	◎	◎	4.11
ポリシ制約においてinhibitAnyPolicyを正しく処理できる	◎	×	◎	◎	4.12
名前制約を正しく処理できる	◎	×	◎	◎	4.13
どのようなCRL発行モデルでも正しく失効検証できる	△	△	◎	◎	
CRLによる基本的な失効検証ができる	◎	◎	◎	◎	4.4
DNは同じだが証明書とCRLの発行鍵が異なる時でも失効検証できる	×	×	◎	◎	4.4
CRL発行モデルにおいて基本的モデルの場合に正しく処理できる	◎	△	◎	◎	4.14
CRL発行モデルにおいてARL/CRLモデルの場合に正しく処理できる	◎	×	◎	◎	4.14
CRL発行モデルにおいてIndirect CRLを正しく処理できる	×	×	◎	◎	4.14
CRL発行モデルにおいてDelta CRLを正しく処理できる	×	×	◎	◎	4.15
未知のプライベートクリティカル拡張を正しく処理できる	◎	◎	◎	◎	4.16

この辺りほぼGoogleの貢献

1.0.0ではRFC3280パス検証の完全サポート

Sun Java JCEのパス検証機能を抜いた

ブリッジ、区分CRLなど複雑なPKIモデルにも対応できる。

※注意：OpenSSL 1.0.0 smimeでは、-verify_retcodeが無くそのままテストは通らない。適切にコマンド引数を変更し、テスト期待値を修正したテストスクリプトを作ってやる必要がある。
cms, smime コマンド用に修正した検証詳細を表示できるスクリプトは後日公開します。

FUJI Xerox

(参考)お時間あるときご覧下さい

PKITS検証結果比較のまとめ(詳細)

- OpenSSL 0.9.8nまでは比較的、仕様に厳密ではあるが、サポートする範囲がWindowsのCryptoAPIと程度のカバー範囲の小さいパス検証であった。(多くのケースでこれで十分)
- Googleが寄付したポリシ検証、失効検証のコードによりOpenSSL 1.0.0ではすべてのテストケースでうまくいくようになった。

- ただ、これが100%成功であることで何か大きな影響があるとは思わない。
  - CRLオンライン配布方法はLDAP URIによるもの
  - DeltaCRLもIndirect CRLもあまり使われているとは言えない
  - 100%サポートした検証はCMSやPKCS#7のためのものであり、SSL/TLSで使えるわけではない。

- 米国の政府、航空機、製薬、金融などの分野ではツールとして役立つかも？
- Chromeで使われているわけでもなさそうで、どんな意図でGoogleが貢献したのかはわからない。

FUJI Xerox

主要な変更点 (4/7)

CMS署名・暗号フォーマットの対応
(openssl cmsコマンド)

CMS/PKCS#7 とは？　＝　署名や暗号データの入れ物

S/MIME署名暗号メールや
PDF署名、汎用の
署名暗号データとして
使われている

CMS署名・暗号フォーマットの対応

- RFC 3852 CMS (Cryptographic Message Syntax)とは？
  - PKCS#7を元にした署名・暗号データのフォーマット
  - PKCS#7とCMSはほとんど同じ（・・・だが、微妙に違う）
  - S/MIME署名・暗号メール、汎用的な署名、暗号データとして用いられている。
  - RFC 5126 CAdES長期署名データの元になっている。

- OpenSSLでは、S/MIME用途が主となっている
  - “openssl cms” コマンドが1.0.0より提供されている。
  - “openssl smime” - PKCS#7ベースの署名・暗号データ
  - “openssl cms” - CMSベースの署名・暗号データ
  - 1.0.0よりpkcs7/cmsで(コッソリ) “-stream” オプションがサポートされ、大容量データに対しても可能になった(はず)
  - “openssl cms” のヘルプ中 ’-skeyid’ は ’-keyid’ の間違い

CMS(RFC 3852)とPKCS#7の署名データの微妙な違い

PKCS#7はCMSと互換性を持つと言われているが、、

CMS(RFC 3852)署名	PKCS #7
証明書や含まれる要素の種類によってバージョンが変わる。	SignedDataのバージョンは常に1
内包署名で含まれるcontentがdata型以外の場合はカプセル化はされない	常にcontentはOCTET STRINGでカプセル化される
署名者識別情報にIssuerAndSerialとSubjectKeyIdentifierのいずれかが使える。	署名者識別情報はIssuerSerialのみ。
SigningTime属性ではGeneralizedTimeも使える	PKCS#9のSigningTimeではUTCTimeしか使えない
汎用性という意味ではこちらが本流のはず。CAdES長期署名、RFC 3161タイムスタンプ、他の多くのRFCではこちらが使われる。	PDF署名では常にこれが使われる。S/MIMEではこちらが使われるケースが比較的多い。

openssl cms/smime -verifyのオプションの違い

オプション	0.9.8n smime	1.0.0 smime	1.0.0 cms
-verify_retcode：エラー理由を終了コードで細分化	―	―	✓*
-CAfile：信頼するルート証明書を指定する	✓*	✓*	✓*
-x509_strict：壊れた証明書の回避策を無効に	✓	✓	✓
-crl_check_all：CRLの検証を全て行う	✓	✓	✓
-extended_crl ：indirectCRLやCRL署名が別鍵	―	✓	✓
-use_deltas：デルタCRLを検証する	―	✓	✓
-policy_check：ポリシ検証を行う	✓	✓	✓
-policy_print：ポリシ検証の結果を表示する	✓	✓	✓
-policy OID1…：受理ポリシを指定する	✓	✓	✓
-explicit_policy：明示的なポリシ必要とする	✓	✓	✓
-inhibit_map：ポリシマッピングを禁止する	―	✓	✓
-inhibit_any：anyPolicyを禁止する	―	✓	✓

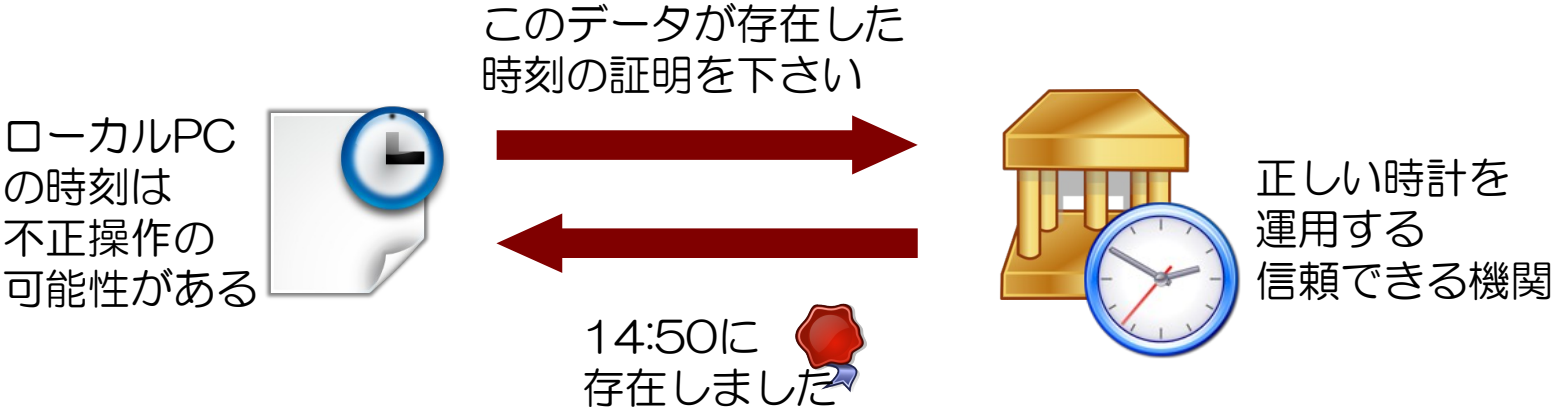
凡例：*：cms, smimeでのみ使われるオプション　無印：appsのアプリ共通で使用可能なオプション

主要な変更点 (5/7)

デジタルタイムスタンプ関連の機能

(openssl ts コマンド)

タイムスタンプとは？＝データが何時あったか第三者証明する仕組み



タイムスタンプ機能

- OpenTSAプロジェクトからの一部ソースコード寄付
  - ハンガリーのテスト用タイムスタンプ局プロジェクト
  - 2003年～2005年ぐらいにアクティブに活動していた
  - Apache + mod_tsa + (OpenSSLのRFC 3161拡張)で実装
- OpenSSL 1.0.0 では
  - OpenSSLに加えたRFC 3161拡張を標準で反映させた。
  - コマンド “openssl ts ・・・” で利用できる。
  - 3つの機能
    - タイムスタンプ要求の生成・表示
    - タイムスタンプトークン・応答の生成・表示
    - タイムスタンプトークンの検証
  - Apacheのmod_tsaモジュールは無い
- タイムスタンプサービスではない。静的なデータの生成・表示

タイムスタンプ応答・トークンの検証

openssl ts -verify -data 対象ファイル -token_in -in トークン
-CAfile ルート証明書 [-untrusted TSA証明書]

残念ながら現状では、日本国内の商用RFC3161タイムスタンプ局のタイムスタンプトークンを検証するとエラーになってしまう。

扱えない原因

- 時刻監査証(X.509 v2属性証明書)がトークンに含まれる場合、opensslではこれをパーズできない。
- reqCertフラグをfalseにしてTSA証明書、時刻監査証を含めないようにしても証明書置換防止のためのSigningCertificate属性に参照情報が含まれる。
- Openssl ts -verifyではSigningCertificate属性の検証も行うが、SigningCertificate属性に時刻監査証のハッシュがあり、時刻監査証をトークンに含めなくとも参照情報の検証ができずにエラーとなる。
- 時刻監査証やその参照情報を扱わないTSPならばOK

RFC 3161準拠の日本の認定タイムスタンプ局のトークンを扱えるようにOpenSSL ts コマンドの修正が望まれる

FUJI Xerox

タイムスタンプ応答・トークンの検証(2)

- それでも、openssl tsの実装はとても「まとも」でサポート範囲が広い
- MessageImprintのハッシュアルゴリズム (MD2/4/5,SHA1/2,RIPEMD160,MDC2)
- 秒より小さい単位(ミリ秒、マイクロ秒をサポート)
- 精度範囲(Accuracy)も秒、ミリ秒、マイクロ秒をサポート

主要な変更点 (6/7)

公開鍵暗号の秘密鍵フォーマットの一般化・共通化

秘密鍵フォーマットの一般化・共通化

OpenSSL 0.9.xまで (PKCS#1or5)

今までは鍵フォーマット、鍵生成、鍵の利用方法、APIはRSA, DSA, ECDSAと別々だった

素のままの鍵

RSA
genrsa
-----BEGIN RSA PRIVATE KEY-----
MF8CAQEEGK49cEnm7Tkki9nm7Tkk

DSA
dsaparam
gendsa
-----BEGIN DSA PRIVATE KEY-----
MF8CAQEEGK49cEnm7Tkki9nm7Tkk

ECDSA
ecparam
ecparam -genkey
-----BEGIN EC PRIVATE KEY-----
MF8CAQEEGK49cEnm7Tkki9nm7Tkk

OpenSSL 1.0.0から (PKCS#8)

1.0.0から秘密鍵は共通の箱(PKCS#8)に入れることになった。生成・利用のコマンドやAPIも共通化された

RSA
genpkey
-----BEGIN PRIVATE KEY-----
MF8CAQEEGK49cEnm7Tkki9nm7Tkk

DSA
genpkey -genparam
genpkey

ECDSA
genpkey -genparam
genpkey

アルゴリズムID
素のままの鍵

共通の箱
PKCS#8

※ crypto/{rsa,dsa,ec...}_ameth.c で共通化

FUJI Xerox

主要な変更点 (7/7)

CAのハッシュファイル名の変更

CApathオプションなどで使う 証明書,CRLのハッシュファイル名の変更(1)

簡易認証局(caコマンド)機能や-CApathオプションを指定した証明書の検証機能では、証明書やCRLの参照に特別なファイル名「ハッシュファイル名」を使いますが0.9.xと1.x.xでは互換性がありません。

OpenSSL 0.9.8系

証明書の主体者名の取り出し

OpenSSL 1.0.0系

正規化とは？

そのままバイナリ

名前を正規化しバイナリへ

非互換性

MD5を計算

SHA1を計算

上位4バイトからからLong値を作る

a2	b3	01	23	a2	・	・

b3	a2	23	01

4ビットulong値

注意

従って1.0より前のOpenSSLで運用していたCAを1.0で運用するように変更する場合、証明書、CRLのハッシュファイル名の再計算が必要(tools/c_rehashコマンド)

FUJI Xerox

OpenSSL 1.0.0以降における

証明書の識別名の正規化

■名前比較やハッシュファイル名の生成の際に正規化を使う

- 構造体メンバ x->cert_info->subject->canon_{enc, enclen}

このバイト列と長さを設定

SEQUENCE

属性タイプ：国(C)	
値	PrintableString
	JP
属性タイプ：組織(O)	
値	TeletexString
	Fuji Xerox
属性タイプ：部門(OU)	
値	UTF8String
	Marketing

中身を取り出す

属性タイプ：国(C)	
値	PrintableString
	JP
属性タイプ：組織(O)	
値	TeletexString
	Fuji Xerox
属性タイプ：部門(OU)	
値	UTF8String
	Marketing

ASN.1 SETの並びとなる

UTF8で小文字に

属性タイプ：国(C)	
値	UTF8String
	jp
属性タイプ：組織(O)	
値	UTF8String
	fuji xerox
属性タイプ：部門(OU)	
値	UTF8String
	marketing

FUJI Xerox

その他の関連情報

その他、細かい1.0.0での変更点

- マニュアルやコマンドヘルプに書かれない細かい変更がある
- 証明書有効期間の2038年問題(符号付きINTEGERの桁あふれ)
  - 1.0.0よりnotAfterが2038年以降の証明書も出せる
    - 0.9.8: -days 10585 → “GEN:19030516020507Z”
    - 1.0.0: -days 10585 → “UTC:390621082510Z”
    - 0.9.8: -days 18250 → “GEN:19240510014540Z”
    - 1.0.0: -days 18250 → “GEN:20600615080536Z”
- openssl ocspコマンド
  - OCSP要求にカスタムHTTPヘッダを付与可能(-header タイプ 値)
  - OCSP要求のハッシュアルゴリズムを指定可能になった(-{md5,sha1,sha256,ripemd160...)だが、要求には反映されてない
- MSの古い鍵形式(PVK(public/private keyblob)をサポート(crypto/pem/pvkfmt.c)

次のOpenSSL 1.1.x の楽しみな機能

- RSA PSS署名のサポート
- CMACのサポート
- “-verify”コマンドの検証パラメーター指定が可能に
- TLS1.1のサポート
- TLS renegotiation

OpenSSLの

Ruby, Perlなどのスクリプト言語からの利用

Rubyからなら OpenSSL APIを 簡単に利用できます

```
% irb      文字列aaaのハッシュ値計算
>> require 'openssl'
>> md = OpenSSL::Digest.new('SHA1')
>> md.update( "aaa" )
>> md.hexdigest()
=> "7e240de74fb1ed..."
```

■ C言語版 Ruby OpenSSL (※1)

- CRubyに標準で組み込まれておりOpenSSLのAPIプリミティブが利用可能
  - SSL,X509,Cipher,Digest,HMAC,BN,PKey,PKCS7,ASN1,EC,NetScapePKI
  - http://www.ruby-lang.org/ja/man/html/openssl.html
- OpenSSL 1.0.0の対応予定
  - 現行版1.8.7-p249, 1.9.1-p378ではOSSL1.0.0とのビルドは未サポート
  - 次期版1.9系(2010.07.31予定)、1.8系(2010.06.24予定)はOSSL1.0.0とビルド可

    現行クラス(Cipher, Pkey等)ではOpenSSL 1.0.0の全暗号アルゴリズムをサポート

    OpenSSL1.0.0の新機能cms, ts等は直近のサポート予定なし

■ Perl OpenSSL (Crypt::OpenSSL::*)

- Crypt::OpenSSL::{CA,RSA,DSA,X509,PKCS12,PKCS10,Bignum,AES,Blowfish...等々} など個別にモジュールあるが特にOpenSSL 1.0.0の対応は無いようだ。

※1: Ruby OpenSSLの最新状況について中村@サリオンシステムズリサーチ様より情報提供頂きました。ありがとうございました。

FUJI Xerox

まとめ

まとめ

- 11年を経て2010年3月ようやくOpenSSL 1.0.0がリリース
  - 2010年6月にはもう1.0.0aが出てる
- 暗号アルゴリズム・CipherSuitesの修正
  - Camelliaに対応、MD2をデフォルト無効に
- 追加機能
  - パス検証はNIST PKITSフルサポート
    - » NIST PKITSは2011年4月までしか使えない！
  - CMS暗号データフォーマットの対応(openssl cms)
  - 汎用的な公開鍵フォーマットと利用(openssl genpkey他)
  - タイムスタンプ機能 (openssl ts)
- 注意点
  - CApath指定用証明書ハッシュファイル名の変更(要ファイル再生成)
- OpenSSL 1.1.0の予定
  - TLS 1.1やRSA-PSS署名のサポート

FUJI Xerox

ご静聴ありがとうございました

Kenji Urushima

電子署名者 : Kenji Urushima
DN : c=JP, l=Tokyo, o=Kenji Urushima, cn=Kenji Urushima, email=kenji.urushima@gmail.com
理由 : **この文書の著者**
場所 : **東京**
日付 : 2010.06.30 18:25:24 +09'00'

付録

本講演に関する補足情報・修正情報のページ(予定)
http://www9.atwiki.jp/kurushima/pages/101.html

参考リンク

- OpenSSL: The Open Source toolkit for SSL/TLS

  http://www.openssl.org/
- Shining Light Productions: OpenSSL for Windows (Windowsバイナリ配布)
  - http://www.slproweb.com/products/Win32OpenSSL.html

- 米国 NIST PKI Testing: Path Validation and Discovery Testing Program
  - http://csrc.nist.gov/groups/ST/crypto_apps_infra/pki/pkitesting.html

- IPA: 2002年度電子政府情報セキュリティ相互運用支援技術の開発

  http://www.ipa.go.jp/security/fy14/development/pki/interop.html パス検証の解説として
- IPA: 2003年度タイムスタンプ・プロトコルに関する技術調査

  http://www.ipa.go.jp/security/fy15/reports/tsp/documents/tsp2003.pdf p149 OpenTSA解説
- JNSA PKI Day 2006 講演資料 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 稲田龍氏
  http://www.jnsa.org/seminar/2006/20060607/inada.pdf

付録1: OpenSSL 1.0.0 CHANGES 全項目と補足説明

- ○ RFC3280 path validation: sufficient to process PKITS tests.
  - NIST PKITSパス検証テストのテストメールをcmsコマンドで検証した結果、全て期待値と一致
- ○ Integrated support for PVK files and keyblobs
  - MSの古い鍵格納フォーマットPVKへの対応
- ○ Change default private key format to PKCS#8
  - 公開鍵の扱いがアルゴリズムに依存せず一般化された。今までgenrsaだとPKCS#1で直接RSA秘密鍵が出てきたが、今後は汎用の公開鍵暗号秘密鍵を生成する。genpkeyで生成することになり、結果はアルゴリズム名と鍵本体を含む汎用の秘密鍵の入れ物PKCS#8を使うようになった。鍵長やDSA,ECDSAなどの鍵パラメータは別途コマンド(dsaparam,ecparam)で生成するのではなく、オプション-keyoptで指定する。
- ○ CMS support: able to process all examples in RFC4134
  - RFC4134で示される全てのCMSのサンプルを処理可能。
- ○ Streaming ASN1 encode support for PKCS#7 and CMS.
  - PKCS#7, CMSの暗号メッセージのフォーマットでストリーム処理が可能になった
- ○ Multiple signer and signer add support for PKCS#7 and CMS
  - CMS,PKCS#7で複数の(並列)署名者、および署名者の追加のサポート(cms -resign)
- ？ ASN1 printing support.
  - ASN.1表示サポートとのことだがソースの変更点が不明で前と変わっていない？
- ○ Whirlpool hash algorithm added.
  - Whirlpoolハッシュアルゴリズムのサポート
- ○ RFC3161 time stamp support.
  - RFC3161タイムスタンプの要求/応答データの生成/表示と検証のサポート
- ○ New generalised public key API supporting ENGINE based algorithms
  - ENGINEベースのアルゴリズムをサポートする新しい汎用化された公開鍵API
- ○ New generalised public key API utilities.
  - 汎用化された公開鍵ユーティリティ(genpkey,pkey,pkeyparam,pkeyutl)
- ○ New ENGINE supporting GOST algorithms.
  - ロシア国家暗号GOSTアルゴリズムをサポートするエンジン
- ○ SSL/TLS GOST ciphersuite support.
  - SSL/TLS GOST ciphersuiteサポート
- ○ PKCS#7 and CMS GOST support.
  - PKCS#7, CMSでのGOSTのサポート
- △ RFC4279 PSK ciphersuite support.
  - RFC4279 PSK ciphersuiteのサポート (共通鍵による認証と鍵交換)
- ○ Supported points format extension for ECC ciphersuites.
  - ECC ciphersuitesにおけるpoint format拡張のサポート。draft-ietf-tls-ecc-12.txtに基づくECC曲線拡張の実装
- ○ ecdsa-with-SHA224/256/384/512 signature types.
  - ecdsaWithSHA224/256/384/512署名タイプのサポート (dgstコマンド sha1/224/256/384/512のみ)
- ○ dsa-with-SHA224 and dsa-with-SHA256 signature types.
  - dsaWithSHA224/SHA256署名タイプのサポート (dgstコマンド sha1/224/256のみ)
- ○ Opaque PRF Input TLS extension support.
  - Internet Draft draft-rescorla-tls-opaque-prf-input-00.txt の実験的実装。ビルド時に拡張の値を指定してビルドする必要がある。これまで固定値だったものを乱数性をクライアントとサーバーの双方に基づいて提供するための米国政府アプリの要件に対応できる拡張。
- ○ Updated time routines to avoid OS limitations.
  - OSの日時関数を使わずに新たに追加されたOPENSSL_gmtime_adj()関数を使うようになりtm構造体を直接扱うようになった。これによりOSによってUnix Origin Timeのinteger桁あふれにより2038年以降の日時を正しく扱えないという問題が無くなる。

凡例：

○ 変更の内容を理解しているもの

？ 変更がどこに影響しているのか、どんな効果があるのか理解していないもの

△ 意味はわかるが実際に動作確認できていないもの。

FUJI Xerox

付録2: OpenSSLコマンド例

```
── 従来の公開鍵暗号の鍵生成 ──────────────────────
従来方法のRSA秘密鍵生成
% openssl genrsa -out aaa.prvkey 1024
従来方法のDSA秘密鍵生成
% openssl dsaparam -out aaa.prm 1024
% openssl gendsa -out aaa.prvkey aaa.prm
従来方法のECDSA秘密鍵生成
% openssl ecparam -out ec.prvkey -name prime192v1 -genkey
── genpkeyによる公開鍵暗号の鍵生成 ──────────────
RSA秘密鍵生成
% openssl genpkey -algorithm RSA -out rsa1.key -pkeyopt rsa_keygen_bits:2048
DSA秘密鍵生成
% openssl genpkey -genparam -algorithm DSA -out dsa1.prm -pkeyopt dsa_paramgen_bits:1024
% openssl genpkey -paramfile dsa1.prm -out dsa1.key
ECDSA秘密鍵
% openssl genpkey -genparam -algorithm EC -out ec1.prm -pkeyopt ec_paramgen_curve:prime192v1
% openssl genpkey -paramfile ec1.prm -out ec1.key
── SSL/TLS CipherSuitesの情報(ciphers)──────────
CipherSuites一覧の詳細表示(Kx鍵交換,Au認証,Enc暗号化,Mac別)
% openssl ciphers -V
SSLv2のCipherSuitesの詳細表示
% openssl ciphers SSLv2 -V
Apache mod_sslでのCipherSuitesの設定パターンと選択されたCipherSuites
% openssl ciphers [パターン]
── ハッシュおよび署名機能(dgst)──────────────────
■ハッシュ計算
% openssl dgst -{sha1,sha256,sha512,md_gost94,whirlpool,md5...} < aaa.txt
■署名 / dgst コマンドでテキストファイル(aaa.txt)に署名
DSA署名
% openssl dgst -sha{1,224,256} -sign dsa.key -binary -out aaa.sig aaa.txt
ECDSA署名
% openssl dgst -sha{1,224,256,384,512} -sign ec.key -binary -out aaa.sig aaa.txt
── SSL/TLSサーバー/クライアントテスト機能(s_server/s_client)──────
■SSLサーバー
% openssl s_server -key ec.key -cert ec.cer -www -accept 443
■SSLクライアント
% openssl s_client -connect localhost:4433
── PKCS#7暗号メッセージフォーマットの機能(smime)──────
■ PKCS#7 SignedData 署名(-sign)
% openssl smime -sign -in aaa.txt -inkey t1.key -signer t1.cer -nodetach -outform DER -out t1.p7s
■ PKCS#7 SignedData 署名への並列署名の追加(-resign)
% openssl smime -resign -in t1.p7s -inform DER -inkey t2.key -signer t2.cer  -nodetach -outform DER -out t1t2.p7s
── CMS暗号メッセージフォーマットの機能(cms)──────
■ CMS SignedData 署名(-sign)
% openssl cms -sign -in aaa.txt -inkey t1.key -signer t1.cer [-keyid] -nodetach -outform DER -out t1.p7s
■ CMS SignedData 署名への並列署名の追加(-resign)
% openssl cms -resign -in t1.p7s -inform DER -inkey t2.key -signer t2.cer [-keyid] -nodetach -outform DER -out
        t1t2.p7s
■ CMS EnvelopedData 共通鍵で暗号化後、共通鍵を公開鍵で暗号化(-encrypt)
% openssl cms -encrypt -in aaa.txt -out x.p7s -outform DER -des3 test1.cer
■ CMS Signed and EnvelopedData
% openssl cms -sign -in aaa.txt -signer test1.cer -inkey test1.key -text | openssl cms -encrypt -out u.p7s -outform DER
        -des test2.cer
── タイムスタンプ機能(ts) ──────────────────────
■テキストファイルaaa.txtに対しアマノタイムビジネス社のフリータイムスタンプの取得
% export PATH=$PATH:$OPENSSLTOP/misc
% openssl ts -query -data aaa.txt -sha1 -out a.req
% tsget -h http://free-tsu.e-timing.ne.jp/TSS/HttpTspServer -o a.tsr a.req
% openssl ts -reply -in a.tsr -token_out -out a.tst
※ ts -queryで-certオプションを付けるとts -replyで時刻監査証のパースに失敗するので注意
```

```
■タイムスタンプの検証(ts -verify)
% openssl ts -verify -data aaa.txt -in aaa.1res.der -CAfile tsaroot.cer.pem
■タイムスタンプ応答の表示 (ts -reply)
% openssl ts -reply -in t1.res.der -text
■タイムスタンプトークンの表示 (ts -reply -token_in)
% openssl ts -reply -token_in -in t1.tst.der -text
── GOSTの利用 ──────────────────────────────
■GOST engineを使えるようにするopenssl.cnfの設定
[openssl_def]
engines = engine_section
[engine_section]
gost = gost_section
[gost_section]
engine_id = gost
dynamic_path = /usr/local/ssl/lib/engines/libgost.so
default_algorithms = ALL
CRYPT_PARAMS = id-Gost28147-89-CryptoPro-A-ParamSet
■ GOST engineの設定の表示
% openssl engine gost -t -c -vvvv
■ コマンド利用例
楕円鍵の生成
% openssl genpkey -engine gost -algorithm gost2001 -pkeyopt paramset:[A-C] -out ec.pem
公開鍵暗号秘密鍵の生成
% openssl genpkey -engine gost -algorithm gost94 -pkeyopt paramset:[A-D] -out prv.pem
証明書発行要求の生成
% openssl req -new -engine gost -key prv.pem
% openssl req -new -engine gost -key prv.pem -subj /C=JP/CN=hoge.com -out hoge.req
証明書の発行
% openssl x509 -in hoge.req -out hoge.cer -req -signkey prv.pem -engine gost -days 3650
簡易サーバー
% openssl s_server -www -tls1 -accept 9443 -key prv.pem -cert hoge.cer -engine gost
── OCSP ──────────────────────────────────
OCSP要求ファイルの内容表示
% openssl ocsp -reqin req.der -text
OCSP応答ファイルの内容表示
% openssl ocsp -respin res.der -text -noverify
URL指定されたOCSPレスポンダとの通信による証明書の状態の確認
% openssl -ocsp -issuer ca.cer -cert ee.cer -url http://ocsp.foo.com -resp_text -respout resp.der
── PKCS#12 ──────────────────────────────────
証明書(PEM)ファイルと秘密鍵(PEM)ファイルからPKCS12を生成する
% openssl pkcs12 -export -inkey 秘密鍵 -in 証明書 -out PKCS12ファイル
PKCS#12ファイルの内容を表示する
% openssl pkcs12 -in PKCS12ファイル -info
── X.509証明書 ──────────────────────────────
証明書からの情報の取得
% openssl x509 -in aaa.cer.pem -noout -text
X.509証明書のRSA/DSA公開鍵のモジュラスの取得 (*3)
% openssl x509 -in aaa.cer.pem -noout -modulus
Modulus=1F2F...
PEMテキストからDERバイナリへの変換
% openssl x509 -inform PEM -in aaa.cer.pem -outform DER -out aaa.cer.der
DERテキストからPEMバイナリへの変換
% openssl x509 -inform DER -in aaa.cer.der -outform PEM -out aaa.cer.pem
秘密鍵から簡単に証明書の生成(req)
% openssl req -new -key aaa.prvkey -x509 -subj /C=JP/O=TEST1 -set_serial 1 -days 3652 -out aaa.cer.pem
```

※青字：OpenSSL 1.0の新機能

FUJI Xerox