エンジニア向け
はじめてのwolfSSL集中講座

wolfSSL Japan合同会社

2024年11月7日

本日の内容

- wolfSSL 製品のご紹介
- wolfSSLライブラリの概要
- wolfSSLパッケージの構成
- デモ1
  - まずは標準構成でビルド
  - Wiresharkを使ってTLS通信をのぞいてみる
  - wolfCrypt のテストとベンチマークプログラム
  - 他の構成を試してみる
- wolfSSLのポーティング
  - プラットフォーム依存

本日の内容

- wolfCrypt の利用方法・最適化
- デモ 2
  - RSAによる署名検証
- 知っていると差がでるノウハウ/TIPS
  - Troubleshooting 1 – また出たよ。よく見るエラーコードとその対処
  - Troubleshooting 2 – デバックログON！.. デバックログの勘所
  - Troubleshooting 3 – メモリリークや消費メモリを調べたい
- まとめ
- Q&A

エンジニア向け、はじめての wolfSSL集中講座

ねらいと目標

- SSL/TLS の機能についての基本的な理解
- wolfSSLのパッケージ構成と設計
- wolfSSLの移植（ポーティング）で必要になる作業
- 暗号ライブラリ wolfCrypt の使用方法や最適化
- 知っておくと便利な機能やデバック方法の共有

- **前提条件**

- 自社製品のソフトウェア、ファームウェアを開発するエンジニアの方
- インターネットプロトコルに関する基礎的な知識のある方

開発環境のセットアップ

インストールしておくと役立つもの

- OS - Linux/Unix を推奨
- C コンパイラ (gcc または clang)
- Autotools (autoconf、libtool、make)
- テキスト エディターまたは IDE （デモでは VSCodeを使用します。）
- Wireshark
- git (GitHub および wolfSSL リポジトリ)
- wolfSSL 製品 (wolfSSL、wolfSSH、wolfMQTT、wolfTPM)
- OpenSSL コマンド ライン ユーティリティ (openssl)

Open Source

ネットワークセキュリティ専門ベンダー

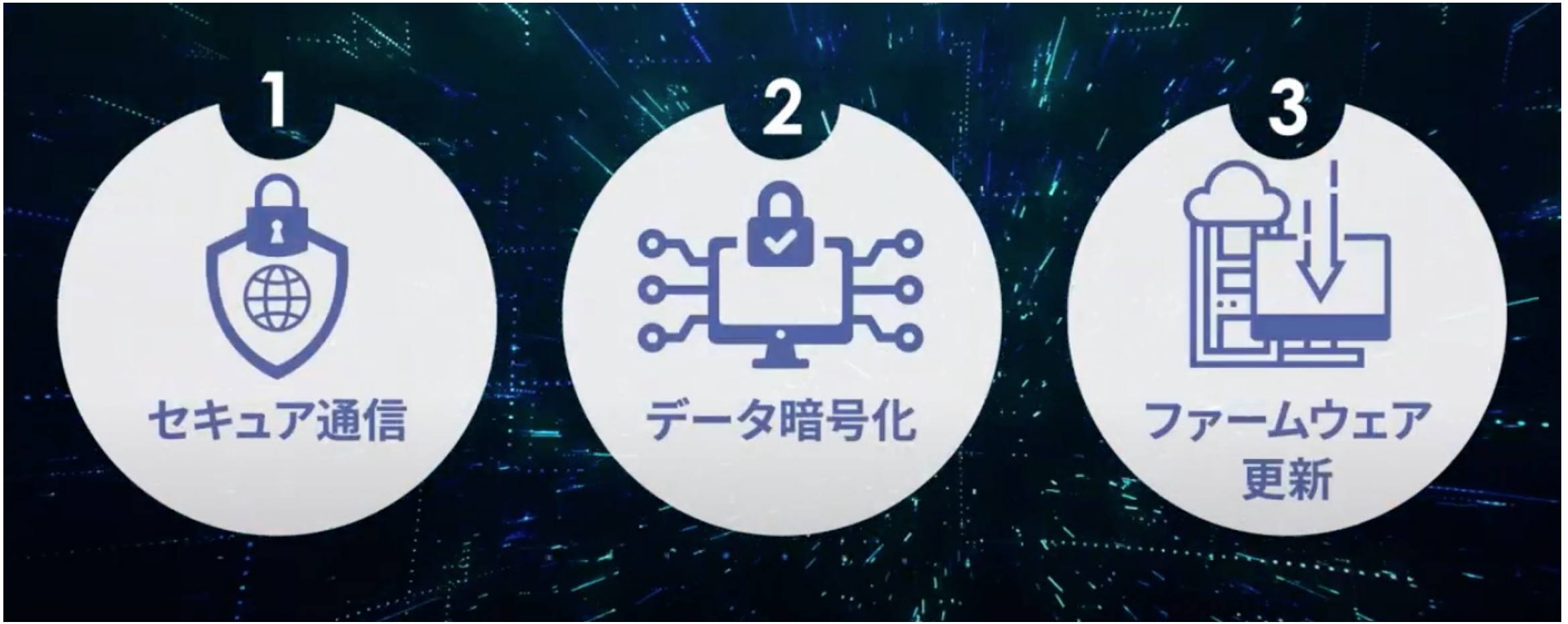
wolfSSL

組み込みシステム向け
ソフトウェアライブラリを提供

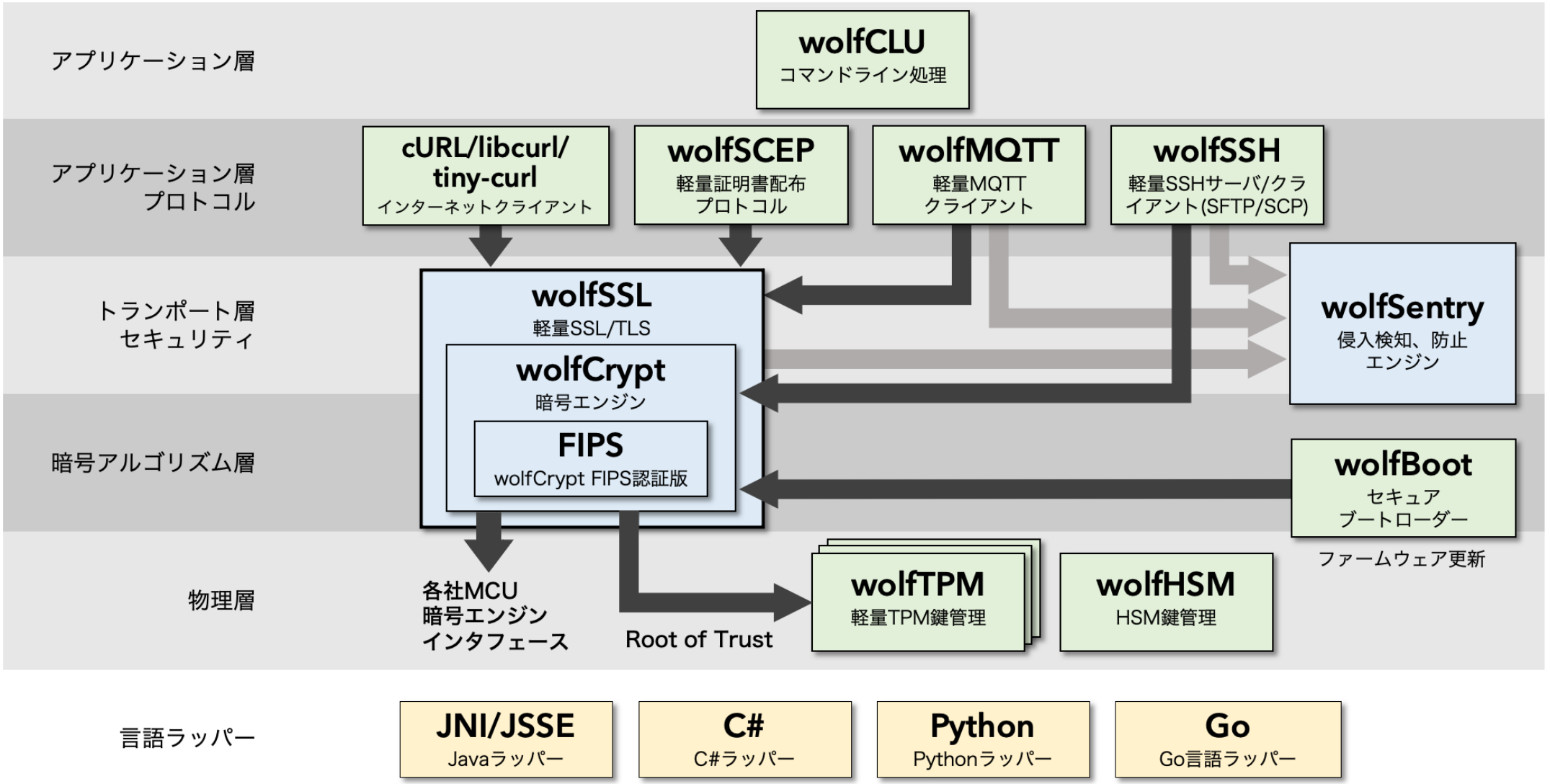
国内外の政府機関、民間企業

2,000社以上のグローバルカスタマー

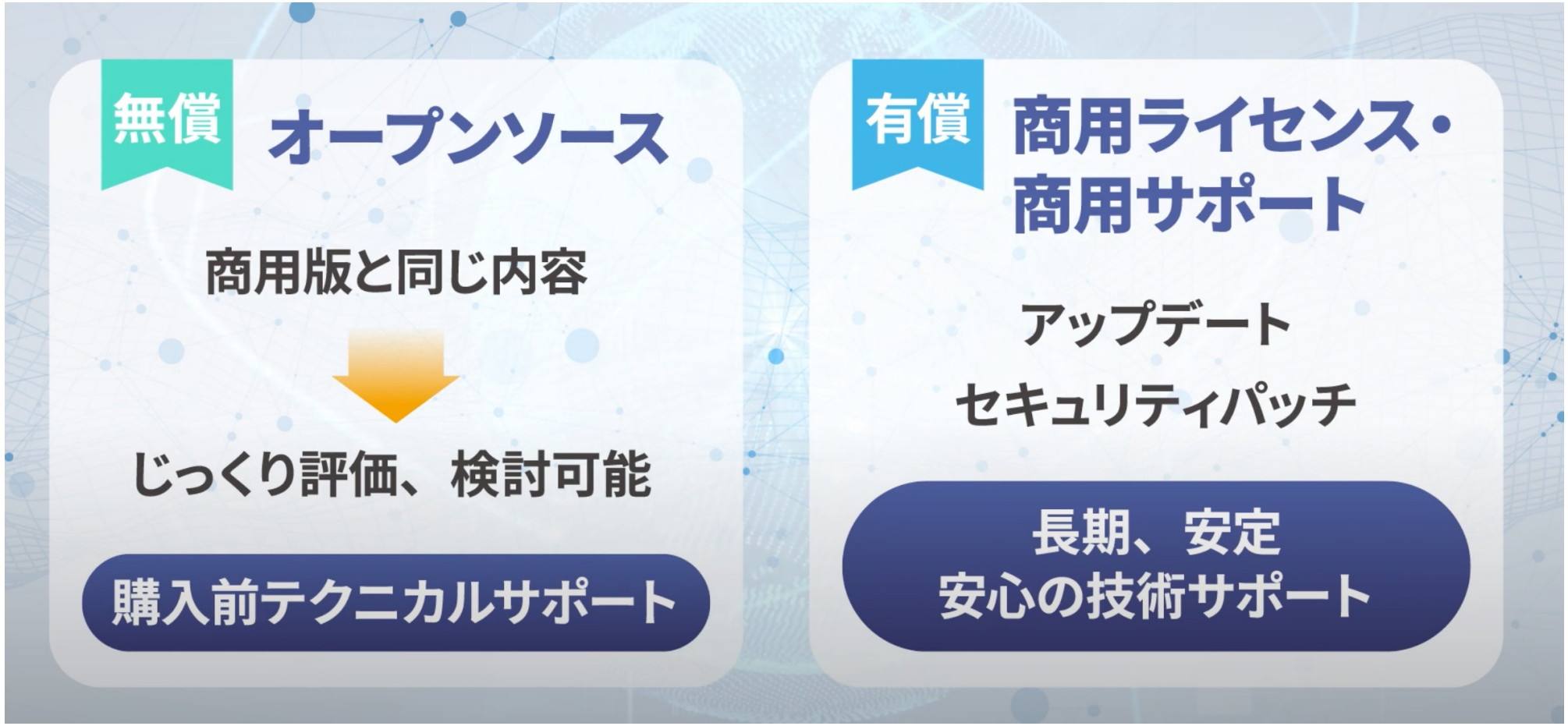
代表的なユースケース



wolfSSLの製品



wolfSSL製品の紹介 - デュアルライセンス



wolfSSL製品の紹介 – ダウンロード方法

- **Webサイト**

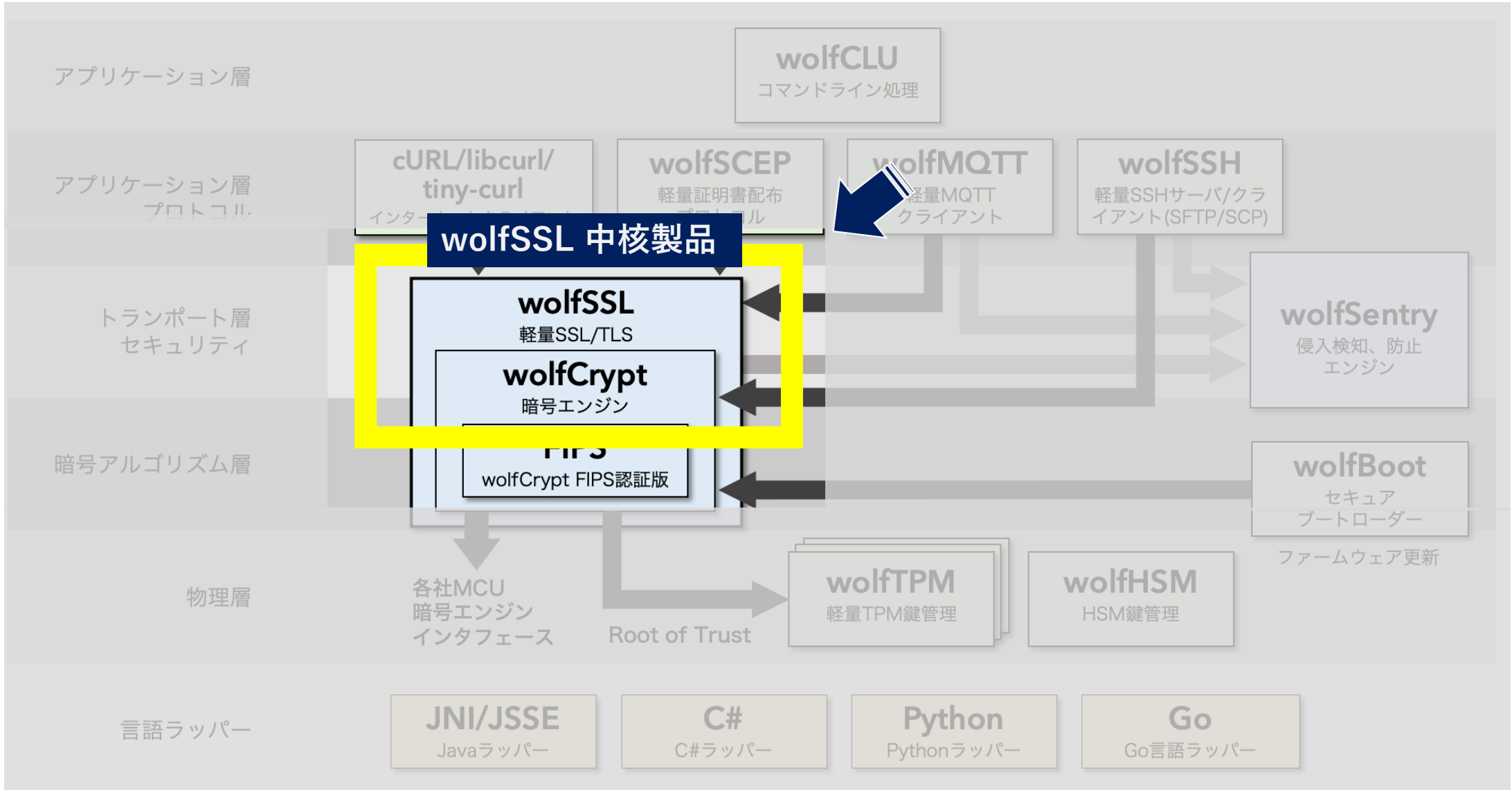
  www.wolfssl.jp

- **GitHub:**

  github.com/wolfssl/wolfssl

wolfSSL ライブラリの概要

wolfSSLの製品



wolfSSL 軽量 SSL/TLS ライブラリの概要

軽量・ポータブルなC言語ベースのソフトウェアライブラリ！

- ✓ **TLS 1.2、TLS1.3、DTLS 1.3対応**
- ✓ **ROM: 20-128 kB**
- ✓ **RAM: セッションあたり1-36 kB**
- ✓ **OpenSSLに比べて、最小20分の１**
- ✓ **多数のOSサポート**

Windows, Linux, Mac OS X, Solaris, ThreadX, VxWorks, FreeBSD, NetBSD, OpenBSD, embedded Linux, WinCE, Haiku, OpenWRT, Nintendo Wii and Gamecube through DevKitPro	iPhone (iOS), Android, QNX, MontaVista, Nucleus, NonStop, uITRON, uT-Kernel, T-Kernel, Micrium uC/OS-III, FreeRTOS, SafeRTOS, Freescale MQX	TinyOS, HP/UX, ARC MQX, TI-RTOS, uTasker, embOS, INtime, Mbed, RIOT, CMSIS-RTOS, FROSTED, Green Hills INTEGRITY, Keil RTX, TOPPERS, PetaLinux, Apache Mynewt

wolfCrypt 暗号ライブラリ

ハッシュ

SHA-2 (SHA-256, SHA-384, SHA512), SHA-3,
（保守：MD2/5, SHA-1など）

共通鍵暗号

Camellia, AES (CBC, CTR, CCM, GCM, OFB), ChaCha20
（保守：3DES, ARC4, RABBIT, HC-128など）

公開鍵 鍵合意

RSA, DH, DHE, ECDH, ECDHE

公開鍵 署名

ECDSA, EdDSA(Ed25519/448), (保守：DSA)

楕円曲線サポート

NIST P-256他, Curve25519/448, Brainpool

メッセージ認証

HMAC, CMAC, Poly1305

パスワード認証

PBKDF2, PKCS#5

wolfSSL パッケージの構成

wolfSSL パッケージの構成

```
drwxr-xr-x   56 chrisc  staff    1792 Apr 21 11:56 .
drwxr-xr-x+  43 chrisc  staff    1376 May 20 13:32 ..
-rw-r--r--    1 chrisc  staff       0 Apr  1 18:17 AUTHORS
-rw-r--r--    1 chrisc  staff   18092 Apr  1 18:17 COPYING
-rw-r--r--    1 chrisc  staff  126855 Apr 21 11:50 ChangeLog.md
drwxr-xr-x   35 chrisc  staff    1120 Apr 21 11:56 IDE
-rw-r--r--    1 chrisc  staff    2170 Apr  1 18:17 INSTALL
drwxr-xr-x    3 chrisc  staff      96 Apr  1 18:17 IPP
-rw-r--r--    1 chrisc  staff     407 Apr  1 18:17 LICENSING
-rw-r--r--    1 chrisc  staff   25707 Apr  1 18:17 LPCExpresso.cproject
-rw-r--r--    1 chrisc  staff     845 Apr  1 18:17 LPCExpresso.project
-rw-r--r--    1 chrisc  staff    9251 Apr  1 18:17 Makefile.am
-rw-r--r--    1 chrisc  staff  479740 Apr 21 11:51 Makefile.in
-rw-r--r--    1 chrisc  staff    9188 Apr 21 11:50 README
-rw-r--r--    1 chrisc  staff    9188 Apr 21 11:50 README.md
-rw-r--r--    1 chrisc  staff   43698 Apr 21 11:51 aclocal.m4
drwxr-xr-x   10 chrisc  staff     320 Apr 21 11:56 build-aux
drwxr-xr-x  100 chrisc  staff    3200 Apr 21 11:56 certs
-rw-r--r--    1 chrisc  staff    5201 Apr 21 11:51 config.in
-rwxr-xr-x    1 chrisc  staff  866528 Apr 21 11:51 configure
-rw-r--r--    1 chrisc  staff  174840 Apr 21 11:21 configure.ac
drwxr-xr-x    3 chrisc  staff      96 Apr 21 11:56 ctaocrypt
drwxr-xr-x   19 chrisc  staff     608 Apr 21 11:56 cyassl
drwxr-xr-x    4 chrisc  staff     128 Apr 21 11:56 doc
drwxr-xr-x    9 chrisc  staff     288 Apr 21 11:56 examples
-rwxr-xr-x    1 chrisc  staff     401 Apr  1 18:17 fips-hash.sh
-rwxr-xr-x    1 chrisc  staff    9643 Apr 15 15:13 gencertbuf.pl
-rw-r--r--    1 chrisc  staff    1842 Apr  1 18:17 input
drwxr-xr-x    3 chrisc  staff      96 Apr 21 11:56 lib
drwxr-xr-x   28 chrisc  staff     896 Apr 21 11:56 m4
drwxr-xr-x   13 chrisc  staff     416 Apr 21 11:56 mcapi
drwxr-xr-x   11 chrisc  staff     352 Apr 21 11:56 mplabx
drwxr-xr-x    8 chrisc  staff     256 Apr 21 11:56 mqx
-rw-r--r--    1 chrisc  staff       6 Apr  1 18:17 quit
-rw-r--r--    1 chrisc  staff     387 Apr  1 18:17 resource.h
drwxr-xr-x    4 chrisc  staff     128 Apr 21 11:56 rpm
drwxr-xr-x   25 chrisc  staff     800 Apr 21 11:56 scripts
drwxr-xr-x   13 chrisc  staff     416 Apr 21 11:56 src
drwxr-xr-x    5 chrisc  staff     160 Apr 21 11:56 sslSniffer
-rw-r--r--    1 chrisc  staff       0 Apr  1 18:17 stamp-h.in
drwxr-xr-x    5 chrisc  staff     160 Apr 21 11:56 support
drwxr-xr-x   10 chrisc  staff     320 Apr 21 11:56 swig
drwxr-xr-x   35 chrisc  staff    1120 Apr 21 11:56 tests
drwxr-xr-x    8 chrisc  staff     256 Apr 21 11:56 testsuite
drwxr-xr-x    8 chrisc  staff     256 Apr 21 11:56 tirtos
-rwxr-xr-x    1 chrisc  staff     263 Apr  1 18:17 valgrind-error.sh
drwxr-xr-x    6 chrisc  staff     192 Apr 21 11:56 wolfcrypt
drwxr-xr-x   21 chrisc  staff     672 Apr 21 11:56 wolfssl
-rwxr-xr-x    1 chrisc  staff    4038 Apr  1 18:17 wolfssl-ntru.sln
-rwxr-xr-x    1 chrisc  staff    6871 Apr  1 18:17 wolfssl-ntru.vcproj
-rw-r--r--    1 chrisc  staff    4918 Apr 21 11:21 wolfssl.rc
-rwxr-xr-x    1 chrisc  staff    4631 Apr  1 18:17 wolfssl.sln
-rwxr-xr-x    1 chrisc  staff    7942 Apr  1 18:17 wolfssl.vcproj
-rw-r--r--    1 chrisc  staff   21121 Apr  1 18:17 wolfssl.vcxproj
-rw-r--r--    1 chrisc  staff   10287 Apr  1 18:17 wolfssl64.sln
drwxr-xr-x    5 chrisc  staff     160 Apr 21 11:56 wrapper
```

- **wolfssl.jpあるいはGitHubからダウンロード**

wolfSSL パッケージの構成 -ファイル/フォルダ構成

項　目		ディレクトリ名
プログラム	SSL/TLS層	src/
	暗号エンジン層	wolfcrypt/src
ヘッダー ファイル	SSL/TLS層	wolfssl/
	暗号エンジン層	wolfssl/wolfcrypt
ビルドツール		configure, cmake,…
IDEプロジェクト、設定ファイル （IAR, MDK ARM, XCODE, e2Studio …)		IDE/
サンプルプログラム		examples/client, server,… github.com/wolfssl-examples
wolfCryptテストプログラム		wolfcrypt/test
ベンチマークプログラム		wolfcrypt/benchmark
テスト用証明書, 鍵		certs/*.{pem, der} wolfssl/certs_test.h
自動テスト		tests, scripts
ユーザ定義オプション		user_settings.h

wolfSSL パッケージの構成 - ソースファイル

- **Cプログラム本体**

./src
　TLS層プロトコル
./wolfcrypt/src
　暗号ライブラリー

- **ヘッダーファイル**

./wolfssl
　TLS層プロトコル
./wolfssl/wolfcrypt
　暗号ライブラリー

オプション：

特定MCU向け
./wolfcrypt/src/port
./wolfssl/wolfcrypt/port

OpenSSL拡張API
./wolfssl/openssl

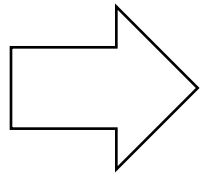
wolfSSL パッケージの構成 - ビルドツール

autotools

./autogen.sh
./configure
./Makefile

cmake

./CMakeLists.txt



Autoconf や cmakeが
利用できない環境。

user_settings.h というヘッダーファイルをプロジェクトに取り込んで使用する。

必要な機能の追加もしくは不要な機能の取り外しをマクロ定義で行う

wolfSSL パッケージの構成 – user_settings.h のサンプル

user_settings.h のサンプル
./examples/configs/
README.md
user_settings_template.h //Template として。#if 0/1
user_settings_all.h // --enable-all を指定したもの相当
user_settings_tls12.h // TLS v1.2 client Only
user_settings_wolfssh.h // wolfSSH を利用する場合
…

Configure コマンド実行時には wolfssl/option.h が作成される

wolfSSL パッケージの構成 - IDEプロジェクトのサンプル

./IDE

- Arduino IDE
- ARM TrustZone CryptoCell
- Eclipse
- Espressif ESP-IDF
- GCC-ARM
- Qualcomm HEXAGON SDK
- NXP Hexiwear
- IAR-EWARM
- Linux SGX
- LPCXpresso
- ARM Keil MDK-ARM
- OPENSTM32
- Rowley CrossWorks for ARM
- Renesas e2studio / CS+
- Atollic TrueSTUDIO
- Cypress WICED Studio
- XCode
- Xilinx SDK
- and more!

Windows Visual Studio

./wolfssl.vxcproj
./wolfssl64.sln

wolfSSL パッケージの構成 - その他プラットフォーム

- **./INSTALL にプラットフォームごとのビルド方法の説明があります**
  - Linux (release, GitHub)
  - iOS
  - Windows
  - IAR
  - Keil
  - CMake
  - Microchip
  - NXP / Freescale
  - Rowley Crossworks
  - Arduino
  - New platforms

wolfSSL パッケージの構成 - サンプルサーバ/クライアント

./examples
- echoserver: 簡単なサンプルサーバ
- echoclient: 簡単なサンプルクライアント
- server: 対向テスト用サンプルサーバ
- client: 対向テスト用サンプルクライアント

さらにたくさんのサンプルプログラム：
https://github.com/wolfssl/wolfssl-examples

wolfSSL パッケージの構成 - サンプル証明書、鍵ファイル

./certs

- サーバ証明書、プライベート鍵
- クライアント証明書、プライベート鍵
- CA 証明書、プライベート鍵
- RSA 1024, 2048, 3072, 4096ビット
- ECC 256ビット
- CRL、OCSP
- DH パラメータ

wolfSSL パッケージの構成 - サンプル証明書、鍵データ

ファイルシステムの無いプラットフォーム向け
C言語初期値データ

./wolfssl/certs_test.h

ヘッダーファイルの生成スクリプト

./gencertbuf.pl

wolfSSL パッケージの構成 - テスト、ベンチマーク

- 暗号アルゴリズムテスト
  ./wolfcrypt/test
- 暗号アルゴリズムベンチマーク
  ./wolfcrypt/benchmark
- TLSベンチマーク
  ./examples/benchmark

wolfSSL パッケージの構成 - 自動テスト

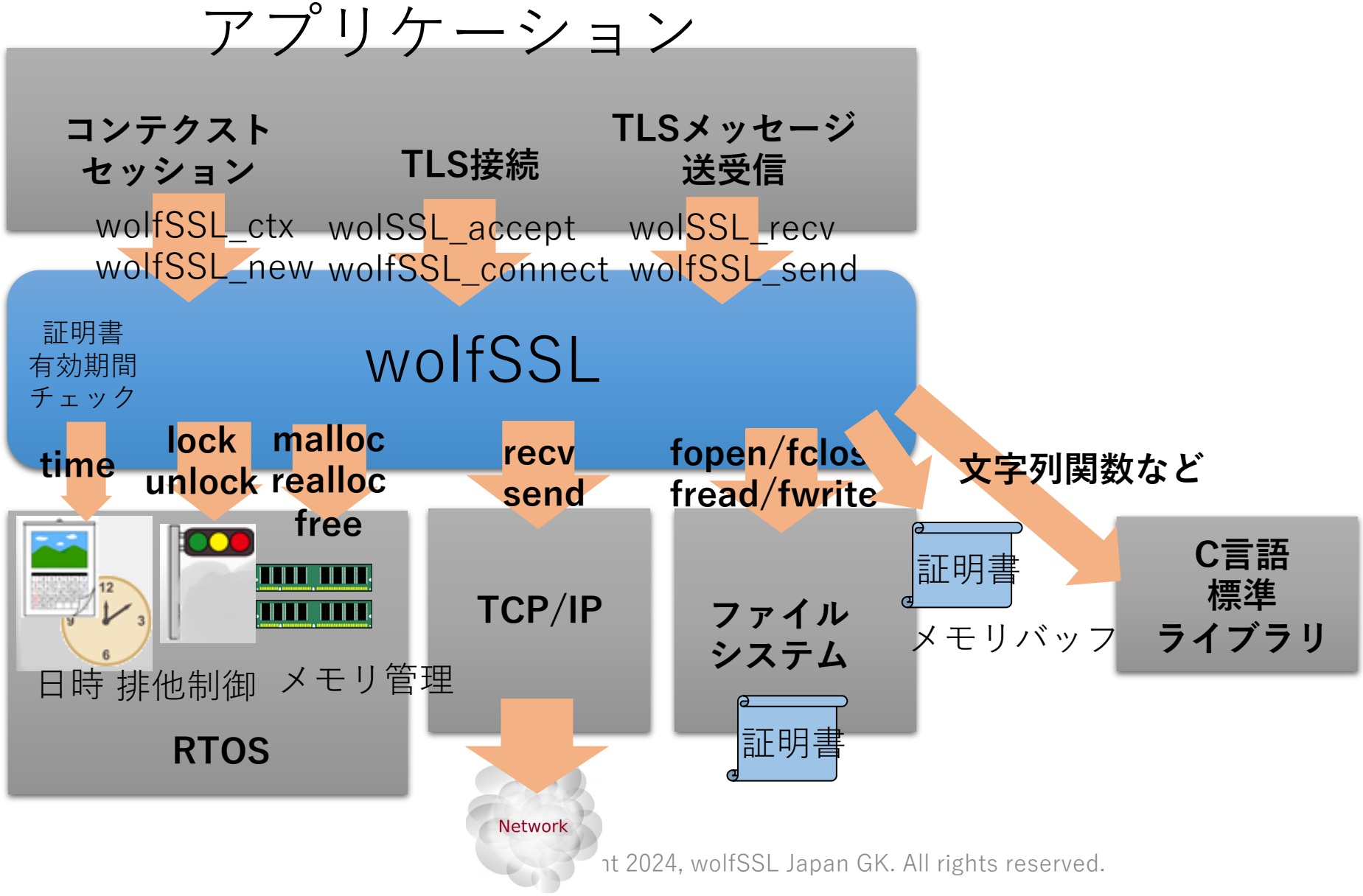
「make check」でビルド時に自動実行されるテスト

- 単体テスト (tests/unit.test)
- wolfSSL テストスィート (testsuite/testsuite.test
  - wolfCrypt テスト
  - クライアント・サーバテスト
- セッション再開テスト (scripts/resume.test)
- OpenSSL 互換機能テスト (scripts/openssl.test)
- 外部接続テスト (scripts/external.test)
- Googleインターオペラビリティテスト (scripts/google.test)

デモ１
wolfSSLビルドと実行

wolfSSLのポーティング

wolfSSLのポーティング- プラットフォーム依存



wolfSSLのポーティング – プロセッサ(CPU/MPU)

- アーキテクチャ、命令セットの違いは**Cコンパイラ**で吸収
- **ワードサイズ**
  必須：int 32ビット以上
- **エンディアン**
  デフォルト：リトルエンディアン
  オプション指定：ビッグエンディアン
- **メモリー量**
  スタックサイズ：16kバイト
  ヒープサイズ：6kバイト（ネットワークI/Oバッファ用）

wolfSSLのポーティング – OS 依存

- **排他制御**
  プラットフォームとなるRTOSのセマフォなどを利用

- **非互換、互換性に不安のある場合は**

  独自MUTEXオプション：WOLFSSL_USER_MUTEX
  - 初期化　　：int wc_InitMutex(wolfSSL_Mutex* m) { ... }
  - 解放　　：int wc_FreeMutex(wolfSSL_Mutex *m) { ... }
  - ロック　　：int wc_LockMutex(wolfSSL_Mutex *m) { ... }
  - アンロック：int wc_UnLockMutex(wolfSSL_Mutex *m) { ... }

wolfSSLのポーティング – TCP/IPスタック依存

TPC接続/切断はライブラリー外、アプリケーション側で適宜

- デフォルトは**BSD Socket**

- **コンフィグオプション**でサポート
  LwIP、ベンダー製品(MQX, FreeRTOS TCP, NetX, その他)

- 独自API
  - コンフィグオプション：**WOLFSSL_USER_IO**
  - **ユーザ定義のTCPメッセージ送受信コールバック関数**
    実行時にコールバック登録用APIにより登録
    TCP接続時にTCPソケット(またはディスクリプタなど)を登録

  サンプルプログラム：
  https://github.com/wolfSSL/wolfssl-examples/tree/master/tls

wolfSSLのポーティング – ファイルシステム依存

主な目的は証明書・鍵ファイルの取り扱い

- デフォルトはPOSIXファイル
- マクロ定義で、独自のファイルアクセスへマッピング可能

```
#define XFOPEN     fopen
#define XFDOPEN    fdopen
#define XFSEEK     fseek
#define XFTELL     ftell
#define XFREAD     fread
#define XFWRITE    fwrite
#define XFCLOSE    fclose
```

- ファイルシステム無し

  証明書、鍵などメモリーバッファーに格納
  同一機能のファイル用API、メモリーバッファー用APIがサポートされている
  例:プライベート鍵のロード
  　　メモリーバッファから：**wolfSSL_CTX_use_PrivateKey_buffer**
  　　ファイルから　　：**wolfSSL_CTX_use_PrivateKey_file**

wolfSSLのポーティング – 実時間依存

主な目的は証明書の有効期限確認

- デフォルト：UNIX Epoch Time を返却する time()

- 独自APIを指定可能：USER_TIMEマクロ定義
  通常ハードウェアRTCから取得

wolfSSLのポーティング – ヒープ管理

- デフォルト：malloc/free/realloc

- 独自API指定
  コンフィグオプション指定：XMALLOC_USER

```
#define XMALLOC(n, h, t)     myMalloc(n, h, t)
#define XFREE(p, h, t)       myFree(p, h, t)
#define XREALLOC(p, n, h, t) myRealloc(p, n, h, t)
```

- wolfSSLのヒープ管理を使用
  ベアメタル、可変長バッファ管理が無い場合など
  アプリケーションで静的プール領域だけ確保し、wolfSSLで管理する
  コンフィグオプション：WOLFSSL_STATIC_MEMORY
  注意：一部、異なる初期化APIを使用する必要あり
  （詳細はAPIレファレンス参照）

wolfSSLのポーティング – C言語標準関数

- 標準I/O
- ストリング

  標準と異なる場合はマクロ定義：STRING_USER

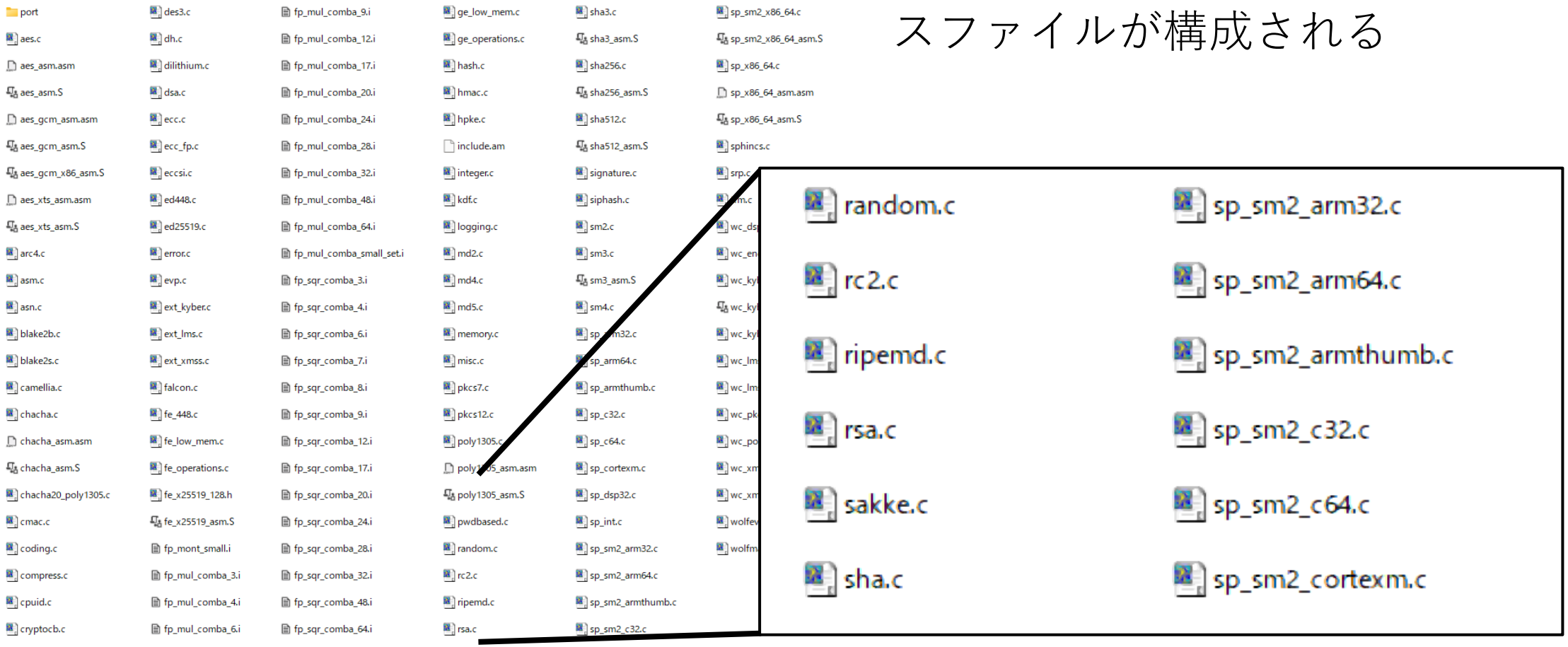
```
#define XSTRLEN(s1)             strlen((s1))
#define XSTRNCPY(s1,s2,n)   strncpy((s1),(s2),(n))
#define XSTRSTR(s1,s2)    strstr((s1),(s2))

#define XSTRNCMP(s1,s2,n)     strncmp((s1),(s2),(n))
#define XSTRNCAT(s1,s2,n)     strncat((s1),(s2),(n))
#define XSTRNCASECMP(s1,s2,n) strncasecmp((s1),(s2),(n))
```

wolfCrypt の利用方法や最適化

wolfCrypt の使用方法や最適化 – ソース構成

機能ごと・暗号アルゴリズムごとにソースファイルが構成される



wolfCrypt の使用方法や最適化 – 特定のアルゴリズムだけを使用する

機能ごと・暗号アルゴリズムごとにソースファイルが構成される

➔　特定のアルゴリズムだけを抽出して、その機能（アルゴリズム）を使用することが可能！

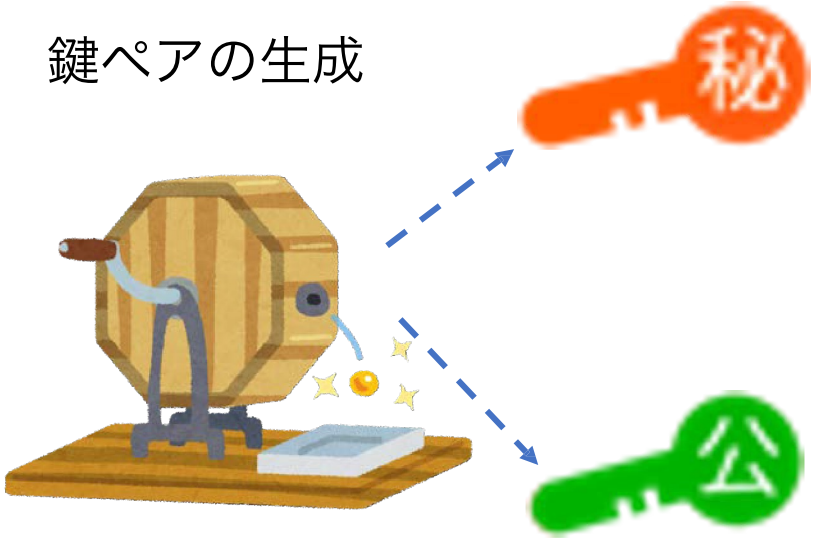
今回は、RSAおよびECCを使って署名検証を行うバイナリを作成し実行してみます。

https://github.com/wolfssl-jp/wolfssl-examples/tree/master/embedded/signature

wolfCrypt の使用方法や最適化 – 署名検証のおさらい

秘密鍵（署名鍵）

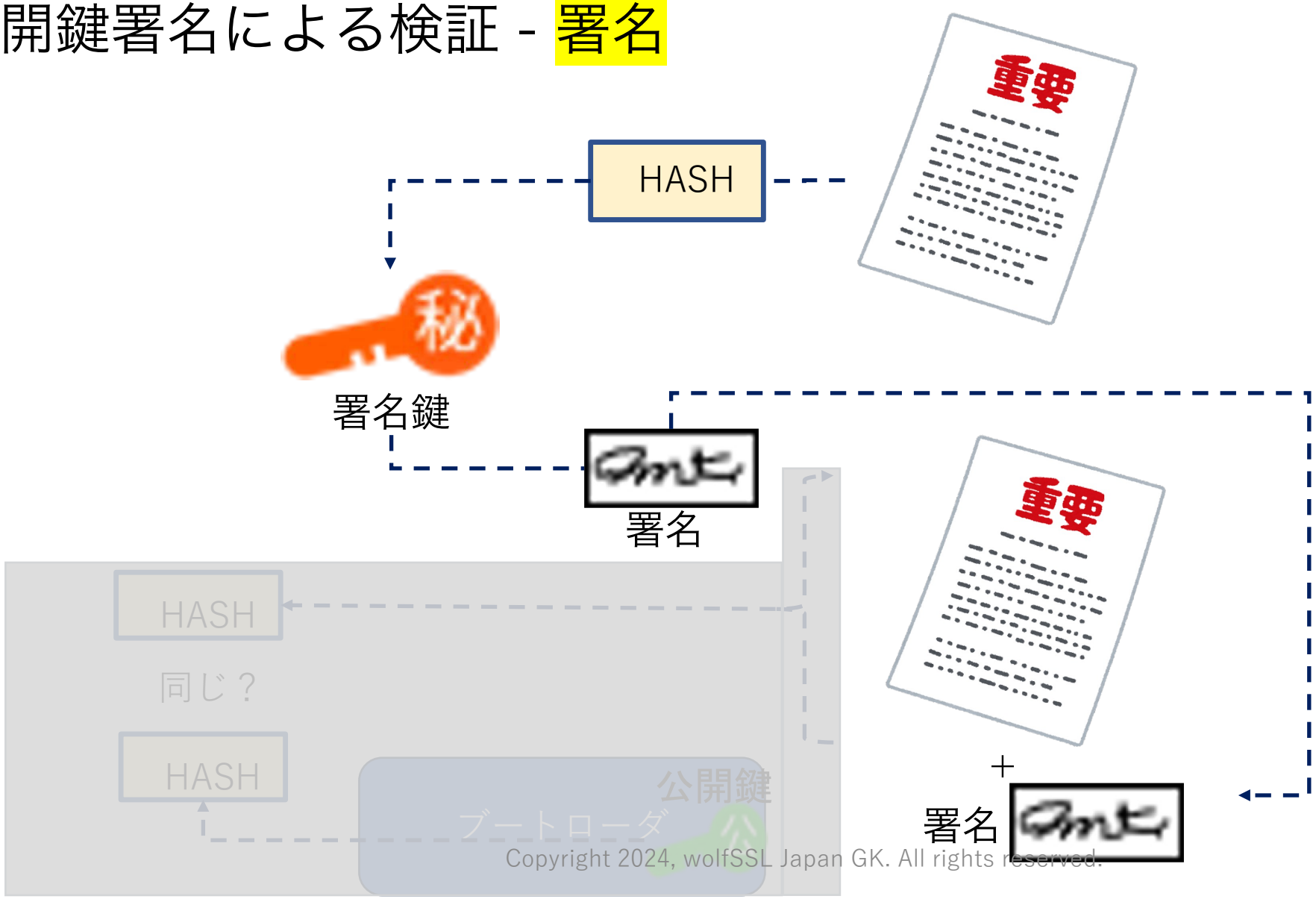
鍵ペアの生成



公開鍵（検証鍵）

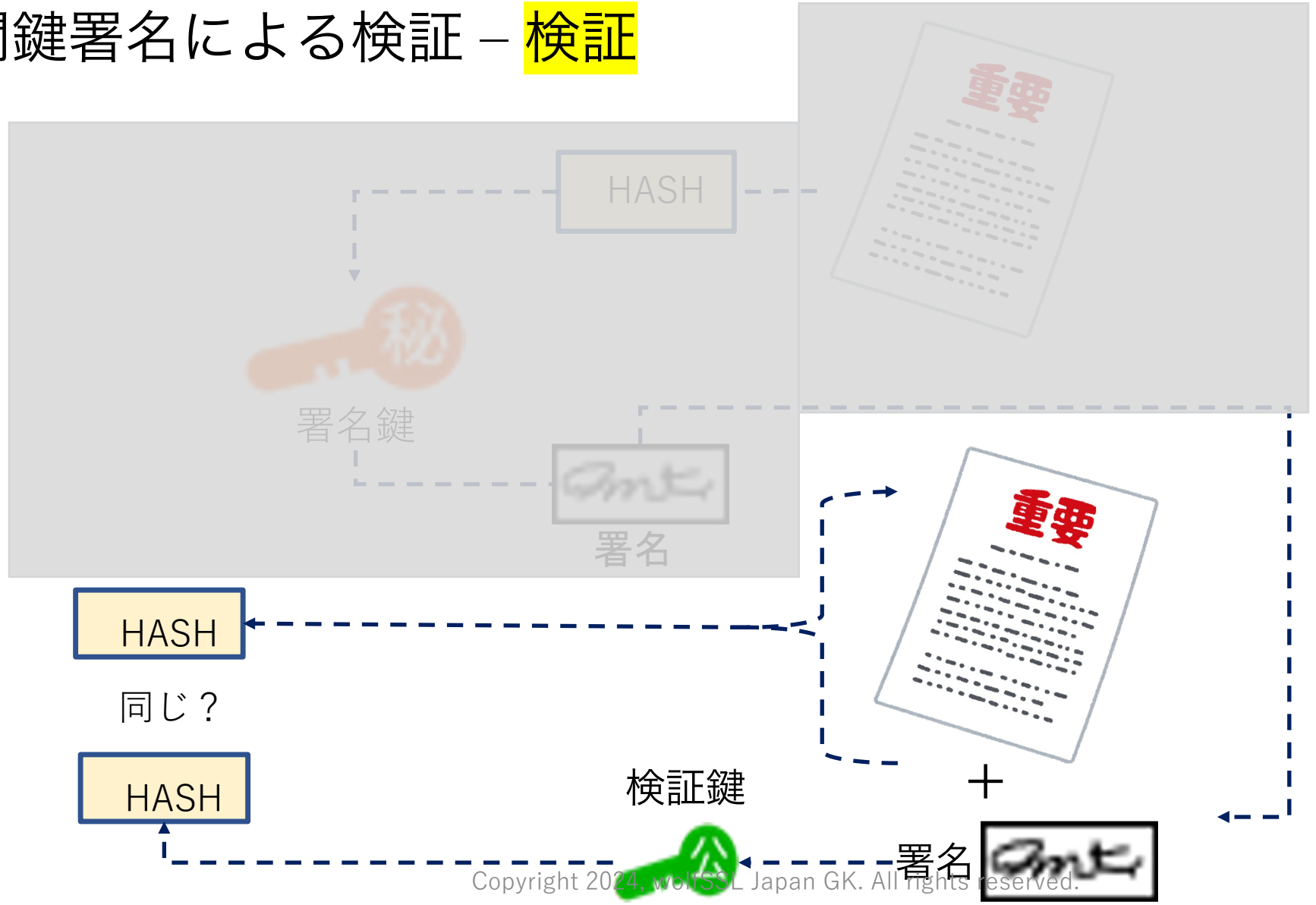
wolfCrypt の使用方法や最適化 – 署名検証のおさらい

公開鍵署名による検証 - 署名



wolfCrypt の使用方法や最適化 – 署名検証のおさらい

公開鍵署名による検証 – 検証



wolfCrypt の使用方法や最適化 – RSA署名検証

必要となるソースファイル		
rsa.c	asn.c	wolfmath.c
sha256.c	wc_port.c	wc_encrypt.c
hash.c	coding.c	
random.c	memory.c	

整数演算に必要となるソースファイル – SP x86_64
sp_int.c
cpuid.c
sp_x86_64.c
sp_x86_64_asm.S

整数演算に必要となるソースファイル – 64bit
sp_int.c
sp_c64.c

整数演算に必要となるソースファイル – 32bit
sp_int.c
sp_c32.c

wolfCrypt の使用方法や最適化 – RSA署名検証

必要となるソースファイル	
rsa.c	RSA処理
sha256.c	SHA256
hash.c	汎用ハッシュ関数
random.c	乱数生成
asn.c	署名エンコード用関数
wc_port.c	wolfCrypt Init/Cleanup 関数
coding.c	Base64 エンコード用関数
memory.c	メモリ割り当て・解放用関数
wolfmath.c	整数ライブラリの補足
wc_encrypt.c	wc_CryptKey

wolfCrypt だけで使用してみる – SP整数演算関数群

“sp_”で始まるCソースファイルは、

- SP(Single Precision)と呼ばれる整数演算
- wolfSSLの独自実装

CPUアーキテクチャ x86_64に特化した性能最適化した構成

多倍長整数に必要となるソースファイル – SP x86_64	
sp_int.c	任意の鍵サイズに対応したSP関数を収録したもの
cpuid.c	CPU毎の処理を行うための関数
sp_x86_64.c	X86_64に特化・高速化したSP関数。C言語記述
sp_x86_64_asm.S	X86_64に特化・高速化したSP関数。アセンブラ記述

64/32 bit 向けに特化した構成

多倍長整数に必要となるソースファイル – 64/32bit	
sp_int.c	任意の鍵サイズに対応したSP関数を収録したもの
sp_c64/32.c	64/32ビット向けSP関数群。特定のビット数に特化したもの

wolfCrypt の使用方法や最適化 – RSA署名検証

必要となるソースファイル		
rsa.c	asn.c	wolfmath.c
sha256.c	wc_port.c	wc_encrypt.c
hash.c	coding.c	
random.c	memory.c	

多倍長整数に必要となるソースファイル – SP x86_64
sp_int.c
cpuid.c
sp_x86_64.c
sp_x86_64_asm.S

多倍長整数演算に必要となるソースファイル – 64bit
sp_int.c
sp_c64.c

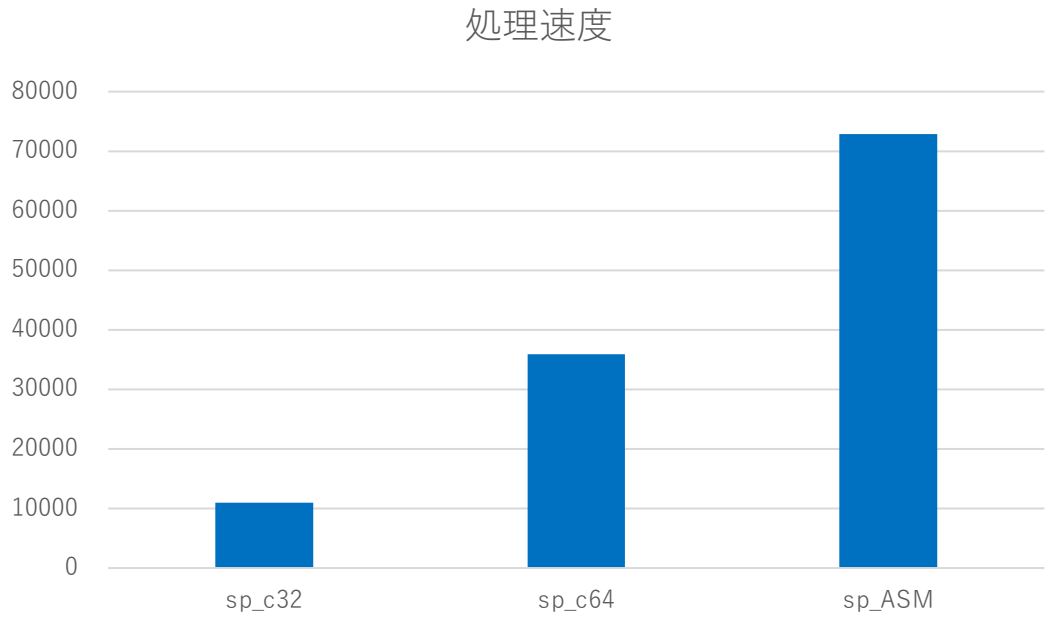
多倍長整数に必要となるソースファイル – 32bit
sp_int.c
sp_c32.c

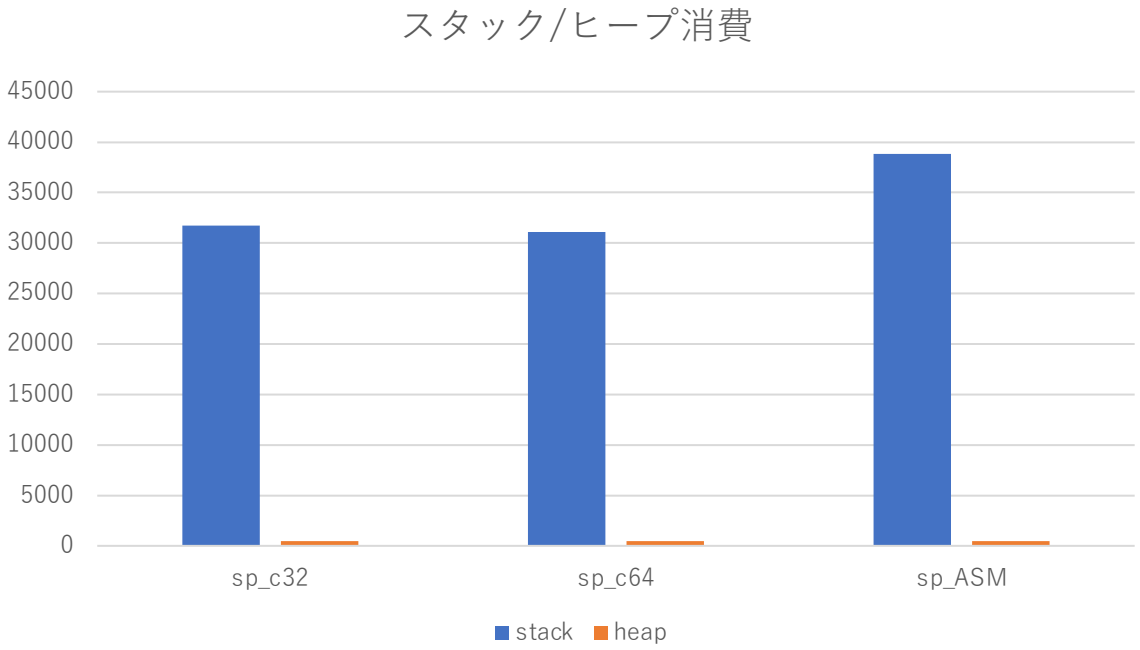
デモ２
RSAよる署名検証デモ

wolfCrypt だけで使用してみる – デモプログラムの説明

デモプログラム	
rsa_sign_verify	RSA 2048 bit 鍵サイズで署名検証を行う。デフォルトで PSS パディング方式を使用。
rsa_sign_verify_bench	検証の回数を測定する。デフォルトは 3 秒
rsa_sign_verify_mem	署名・検証時に使用したスタックとヒープのメモリをダンプする。wolfSSLのメモリトラック機能を利用しています。

wolfCrypt だけで使用してみる –
整数演算処理の違いによるパフォーマンス(RSA 2048 bit) 参考データ

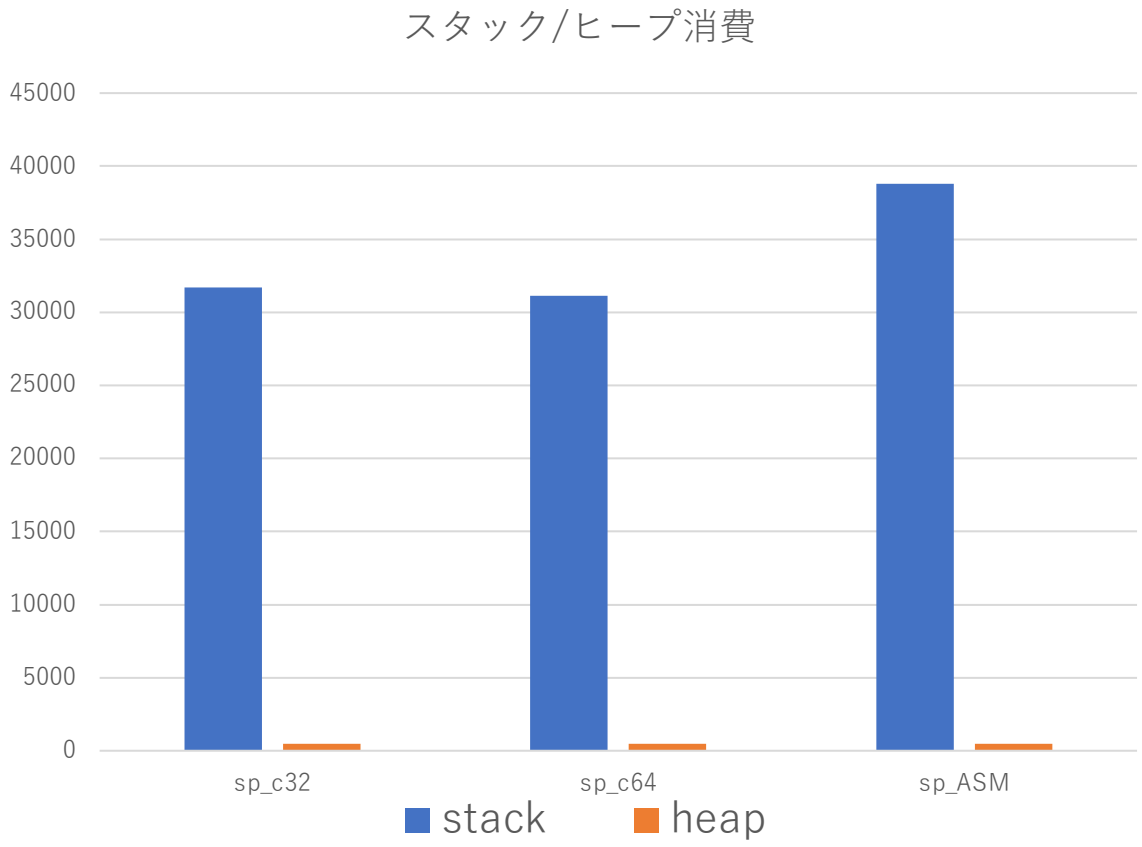


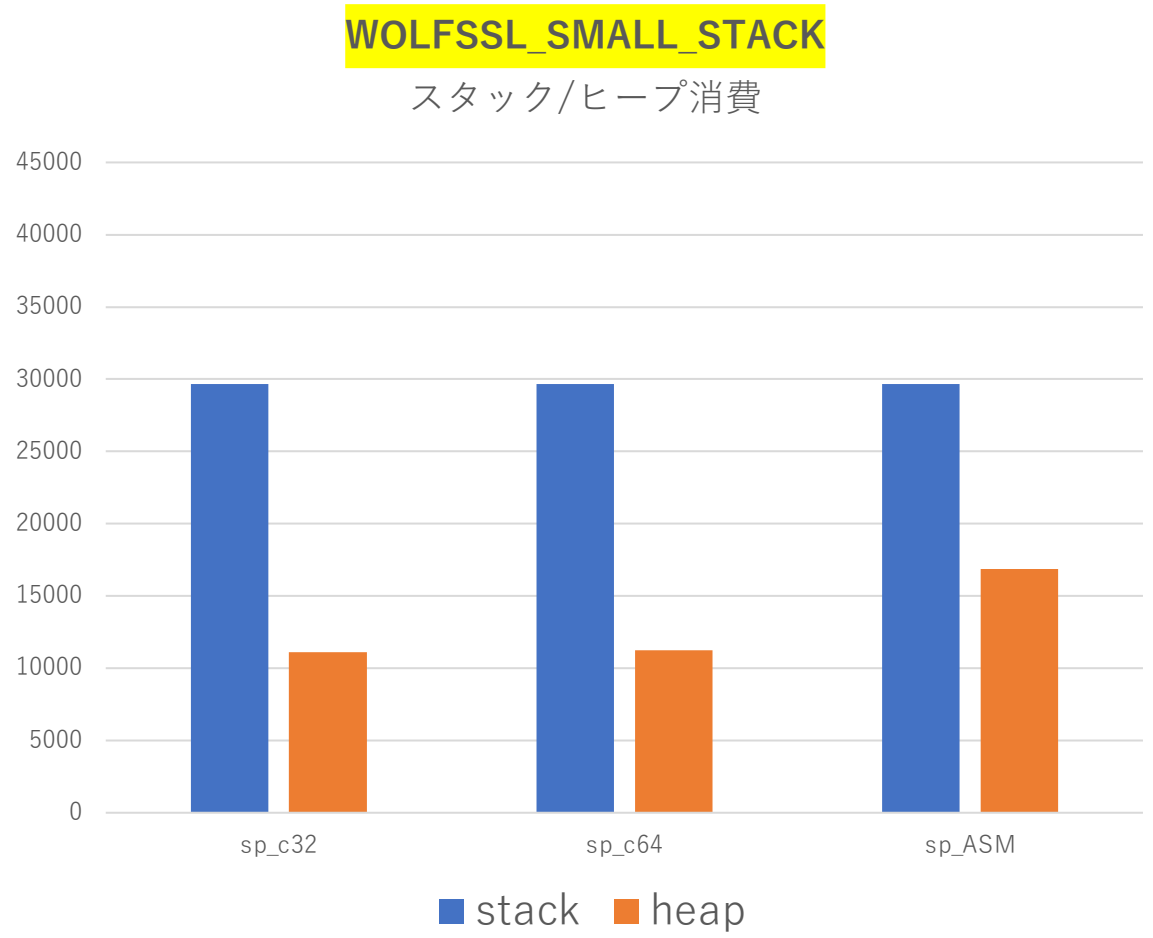


Intel i7 2.20Gを使用

wolfCrypt だけで使用してみる –
整数演算処理の違いによるパフォーマンス(RSA 2048 bit) 参考データ

WOLFSSL_SMALL_STACK マクロを有効





知っておくと差がでるノウハウ/TIPS

知っておくと差がでるノウハウ/TIPS – TB1 よく見るエラーとその原因

エラーコード	エラーメッセージ	原因
-188 ASN_NO_SIGNER_E	ASN no signer to confirm failure	認証にからむ問題で発生する。Peer認証に失敗した場合に発生する。
-151 ASN_AFTER_DATE_E	ASN date error, current date after	サーバ認証に使用するCA証明書の有効期限が切れた場合
-132 BUFFER_E	output buffer too small or input too large	Peer（対向）が送信したRawメッセージ処理中に問題がある場合に発生。ハンドシェークが正常に行われている場合に発生することはなく、スタックの破壊やメモリ割り当てに問題がある場合に発生することがある。
-328 BUFFER_ERROR	malformed buffer input	
-308 SOCKET_ERROR_E	error state on socket	表面上はエラーだが、対向が意図せずソケットをクローズした場合にも発生する。
-326 VERSION_ERROR	record layer version error	TLS バージョンの不整合。例えば、クライアントはTLS1.3で通信を行いたいが、サーバがTLS1.3をサポートしていない場合など

知っておくと差がでるノウハウ/TIPS – TB2 デバックログをON！

デバックログのON

Configure のオプション： --enable-debug
User_settings.h : WOLFSSL_DEBUG

```
miyazakh@delli7 ~/workspace/wolfssl [master]
$ ./examples/server/server
wolfSSL Entering wolfSSL_Init
wolfSSL Entering wolfCrypt_Init
wolfSSL Entering TLSv1_2_server_method_ex
wolfSSL Entering wolfSSL_CTX_new_ex
wolfSSL Entering wolfSSL_CertManagerNew
heap param is null
DYNAMIC_TYPE_CERT_MANAGER Allocating = 240 bytes
wolfSSL Leaving wolfSSL_CTX_new_ex, return 0
wolfSSL Entering wolfSSL_CTX_use_certificate_chain_file
wolfSSL Entering ProcessBuffer
wolfSSL Entering PemToDer
wolfSSL Entering ProcessUserChain
Processing Cert Chain
wolfSSL Entering PemToDer
   Consumed another Cert in Chain
wolfSSL Leaving ProcessUserChain, return 0
Checking cert signature type
Getting Cert Name
Getting Cert Name
wolfSSL Entering GetAlgoId
Cert signature not supported
wolfSSL Leaving ProcessBuffer, return 1
```

基本的なフォーマット：

wolfSSL Entering XXXXX : 関数の入り口

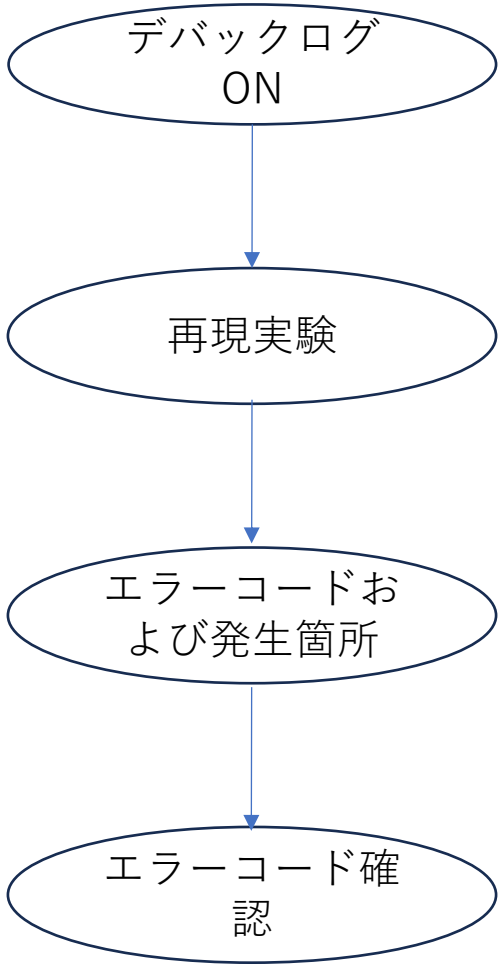
…
…

wolfSSL Leaving XXXXX, return Z

知っておくと差がでるノウハウ/TIPS – TB2 デバックログをON！

$ ./examples/server/server
$ ./examples/client/client -A ./certs/ca-ecc-cert.pem

```
No CA signer to verify with
Failed to verify CA from chain
wolfSSL error occurred, error = -188
wolfSSL Entering SendAlert
wolfSSL Entering SendAlert
SendAlert: 48 unknown_ca
growing output buffer
Shrinking output buffer
wolfSSL Leaving SendAlert, return 0
wolfSSL Leaving ProcessPeerCerts, return -188
wolfSSL Leaving DoCertificate, return -188
wolfSSL Leaving DoHandShakeMsgType(), return -188
wolfSSL Leaving DoHandShakeMsg(), return -188
wolfSSL error occurred, error = -188
wolfSSL error occurred, error = -188
wolfSSL Entering wolfSSL_get_error
wolfSSL Leaving wolfSSL_get_error, return -188
wolfSSL Entering wolfSSL_get_error
wolfSSL Leaving wolfSSL_get_error, return -188
wolfSSL Entering wolfSSL_ERR_error_string
wolfSSL_connect error -188, ASN no signer error to confirm failure
wolfSSL Entering wolfSSL_free
```



知っているだけで差がでるノウハウ/TIPS – TB3 メモリトラック

メモリトラックの機能 ON

Configure のオプション： --enable-trackmemory
User_settings.h : WOLFSSL_TRACK_MEMORY

API	
InitMemoryTracker()	メモリトラック機能を初期化
ShowMemoryTracker()	現在までの状況を表示

```
$ ./examples/client/client
SSL version is TLSv1.2
SSL cipher suite is
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
SSL curve name is SECP256R1
I hear you fa shizzle!
total   Allocs   =      103          // 割り当て回数
total   Deallocs =      103          // 解放回数
total   Bytes    =   121033          // 総割り当てバイト
peak    Bytes    =    32962          // ピーク時の総バイト数
current Bytes    =        0          //  未開放のバイト数
```

知っているだけで差がでるノウハウ/TIPS – TB3 メモリトラック（スタック）

<wolfSSL>/wolfssl/wolfcrypt/mem_track.h 内の StackSizeCheck() を利用
pthead が使用可能な環境でのみ利用できる

スタック領域に規定値（0x01)を予めセットしておき、その値が上書きされた領域からスタックに消費されたバイト数を求める。対象関数はスレッドとして起動する

StackSizeCheck(struct func_args* args, thread_func tf)	第一引数：VERBOSEがON時のみ必要 第二引数：対象関数ポインタ

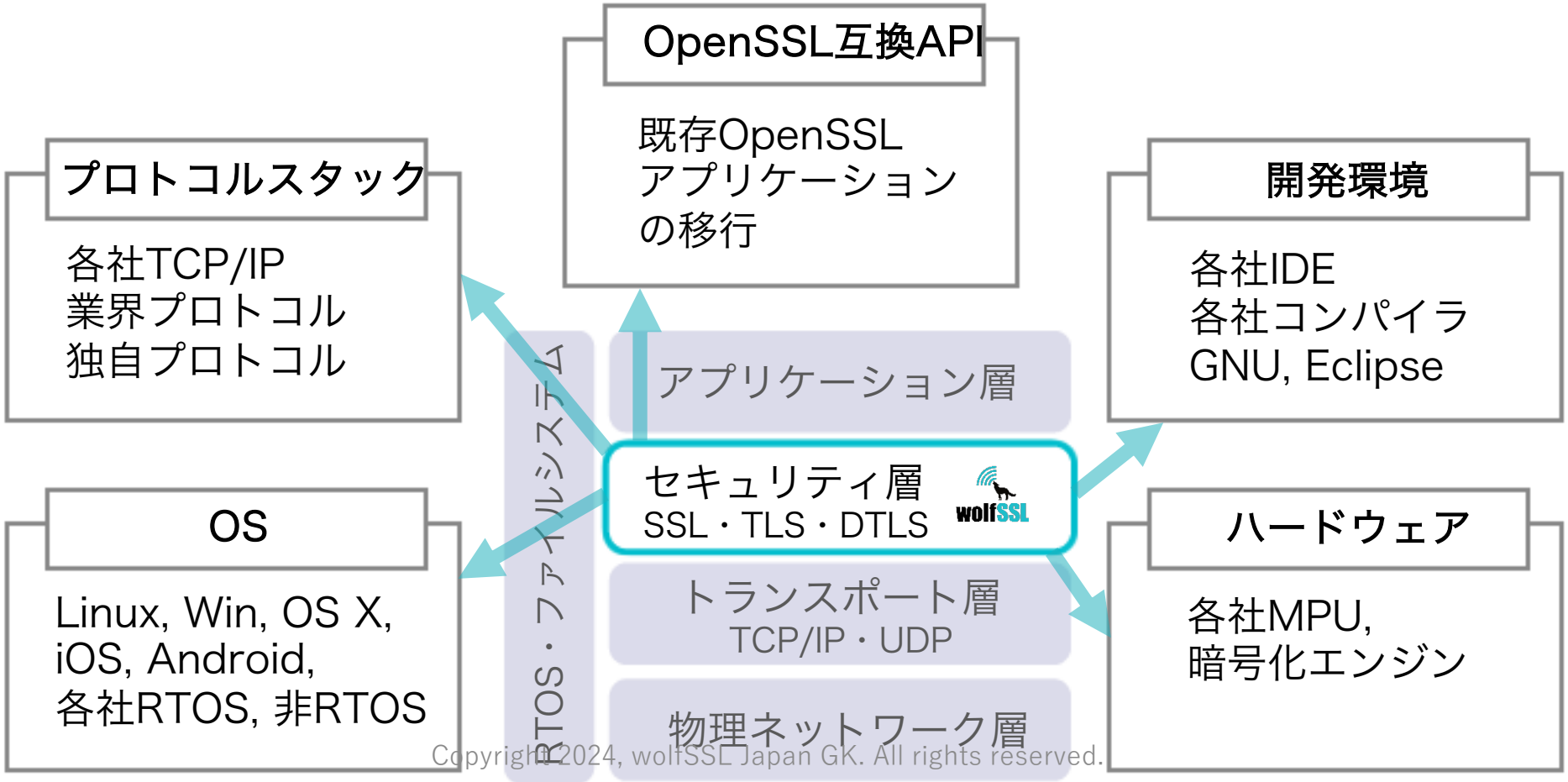
まとめ

wolfSSL 軽量 SSL/TLS ライブラリ – まとめ

各社環境への親和性

軽量・ポータブルなC言語ベースのライブラリ

➔ さまざまな環境で動作する既存製品に容易にセキュリティプロトコルを追加



wolfSSL 軽量 SSL/TLS ライブラリ – まとめ

幅広いプラットフォームにパッケージ提供

- Alpine Linux
- erpc64le
- bullseye
- chimaera
- Debian
- FreeBSD
- GNU Guix
- HaikuPorts master
- Homebrew
- Manjaro
- MidnightBSD
- mipssf-k3.4
- Parabola
- Pardus 21
- Parrot
- Raspbian
- Ubuntu

wolfSSL 軽量 SSL/TLS ライブラリ – まとめ

OSS インテグレーション実績

- MySQL
- OpenWRT
- Gargoyle
- devkitPro
- stunnel
- Lighttpd (Lighty)
- LuCI
- XBMC
- FRITZ!Box
- Neufbox
- TomatoUSB
- BMX6
- cURL
- mongoose
- libscs
- TinyPKC
- Gearman
- fb4nds
- openGalaxy
- ChibiOS
- Open Vehicle Monitoring System
- Riot-OS
- OpenVPN
- cjose
- hostapd/wpa_supplicant
- libest
- nginx
- OpenResty
- OpenSSH
- StrongSwan

wolfCrypt の利用方法 – まとめ

- 豊富なアルゴリズムの選択肢
  - ハッシュ
    SHA-2(256/384/512), SHA-3
  - メッセージ認証コード
    HMAC,CMAC, Poly1305
  - 共通鍵暗号
    AES-CBC/CTR/CCM/CFB
  - 共通鍵暗号（認証付き）
    AES-GCM/CCM, Chacha-Poly
  - 公開鍵暗号
    RSA-1024/2048/3072/4096

    鍵合意
    DHE/ECDHE(SECP/Curve25519/488)

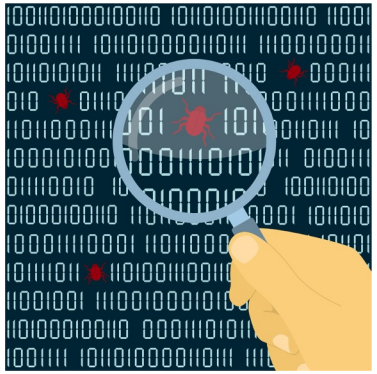
    公開鍵署名
    RSA/DSA/ECDSA/EdDSA(Ed25519/488
- パフォーマンスやリソース消費抑制の最大化
  - 整数演算を選択する　－　SP 32ビット向け、64ビット向け、特定MPU向け

wolfCrypt の利用方法 – まとめ

- ハードウェア暗号・アクセラレーション対応
- 第三者機関認証 FIPS140-3, DO-178C

wolfSSL 軽量 SSL/TLS ライブラリ – まとめ

多面的 なテストの実施:



- API 単体テスト
- 暗号スイートテスト
- アルゴリズムテスト
- ベンチマークテスト
- 静的解析(Coverity, scan-build, Facebook Infer)
- メモリーエラー(valgrind, fsanitize=address)
- 相互接続性テスト
- 他社環境ビルド
- コンパイラーテスト
- ピアレビュー
- 第三者テスト
- ファジング(custom, network, AFL, tlsfuzzer, libfuzzer, OSS-fuzz)
- 継続的インテグレーション(CI)
- 夜間テストサイクル

wolfSSL 軽量 SSL/TLS ライブラリ – まとめ

脆弱性への対応/体制

- 情報源
  - 自社CIテストの一環として強化された継続的なFuzzing
  - オープンソースによる暗号、セキュリティ研究コミュニティーからの報告
  - 顧客、セキュリティ部門からの報告
- 情報公開、告知
  - CVE(Common Vulnerabilities and Exposures)
  - 自社脆弱性情報サイト
  - 商用サポート顧客への通知
- 対応
  - 深刻度によらず全ての脆弱性に対応
  - 最高プライオリティで対応
- 実績
  - 過去三年の平均対応時間：**３８時間**

デバック方法のTIPS – まとめ

- よく見るエラーコードへの初期対処
- デバックログのONとそのフォーマット
- メモリトラック機能

ご清聴ありがとうございました

ご質問は info@wolfssl.jpへもお気軽にお問い合わせください。