wolfCLU Documentation

2025-04-22

Contents

1 wolfCLU マニュアル

wolfSSL コマンドラインユーティリティ (version 0.0.7)
2021/Nov/24

1.1 始めに

wolfCLU は、アプリケーションを最初から作成するよりも簡単/迅速に、いくつかの一般的な暗号化操作を処理するために作成されました。処理できる操作の例としては、証明書の解析と鍵の生成があります。## wolfCLU のビルド

1.1.1 *NIX 上でのビルド

wolfCLU のビルドに先立ち、最初に –enable-wolfclu フラグを指定して 次のように wolfSSL をビルドしインストールする必要があります：

```
cd wolfssl
./configure --enable-wolfclu
make
sudo make install
```

RC2 を使用して PKCS12 ファイルを解析する場合、または CRL を使用する場合には、wolfSSL のビルドの際にフラグ “–enable-rc2” および “–enable-crl” も使用する必要があることに注意してください。

次に、作成した wolfSSL ライブラリをリンクして wolfCLU をビルドします。

```
cd wolfclu
./configure
make
sudo make install
```

あるいは

```
cd wolfclu
./configure --with-wolfssl=/path/to/wolfssl/install
make
sudo make install
```

ユニットテストを実行する場合には “make check” を実行してください。

1.1.2 Windows 上でのビルド

wolfCLU は、その Visual Studio ソリューションファイルである wolfclu.sln を使用してビルドすることもできます。このソリューションファイルは、32 ビットまたは 64 ビットのダイナミックリンクライブラリのデバッグ ビルドとリリース ビルドの両方を提供します。コンフィグレーションオプション指定のためにファイル `user_settings.h` を用意する必要があります。

この `user_settings.h` のテンプレートとして使用できるファイルが `wolfclu\ide\winvs\user_settings.h` に用意してありいます。このファイルを、ディレクトリ `wolfclu\ide\winvs` から `wolfssl\IDE\WIN` にコピーして使用してください。wolfCLU をサポートする wolfSSL をビルドすることができます。

wolfCLU をビルドする前に、wolfSSL で使用されているのと同じアーキテクチャ (Win32 または x64) が選択されていることを確認してください。

このプロジェクトでは、wolfSSH および wolfSSL ソース ディレクトリが各々バージョン番号がない状態でサイドバイサイドで配置されていることを前提としています：


```
    Projects\
        wolfclu\
        wolfssl\
```

リリース構成で wolfCLU をビルドすると、Release\Win32 あるいは Release\x64 フォルダに wolfssl.exe が生成されます。

1.1.2.1 ユニットテストの実行 シェルスクリプトのユニットテストを実行するには、sh コマンドまたは bash コマンドのいずれかが必要です。どちらのコマンドも、Windows の Git インストールに付属しています (ただし、それらを PATH に追加する必要がある場合があります)。

1. wolfssl.exe を wolfclu のルートディレクトリにコピーします。
2. ./wolfssl $1 の代わりに ./wolfssl.exe $1 を実行するように、目的のユニットテストの run 関数 (存在する場合は run_fail も) を変更します。
3. ターミナルで、ルート ディレクトリから sh <desired_unit_test> を実行します。たとえば、ハッシュ ユニット テストを実行するには、sh tests\hash\hash-test.sh を実行します。

1.2 コマンドリスト:

- bench
- ca
- crl
- dsaparam
- dgst
- ecparam
- enc
- genkey
- hash
- md5
- pkcs12
- pkey
- rand
- req
- rsa
- s_client
- verify
- x509
- dhparam
- sha256
- sha384
- sha512

1.2.1 BENCH コマンド

ベンチマーク アルゴリズムのコマンドは現在実装途中です。現在使用できるコマンドは、全アルゴリズムを実行する、“wolfSSL bench -all” です。

1.2.2 CA コマンド

証明書への署名に使用されます。このコマンドはコンフィグレーションファイルを指定し、そのファイルから基本的なコンフィグレーション内容を取得することが可能です。

指定可能な引数:

- [-in] 入力となる CSR ファイル

- [-out] 出力先ファイル
- [-keyfile] 秘密鍵ファイル
- [-cert] CA 証明書ファイル
- [-extensions] コンフィグレーションファイル内の解析すべきセクション
- [-md] ハッシュタイプ（sha, sha256, ...）
- [-inform] CSR ファイル形式（PEM/DER）
- [-config] コンフィグレーションファイル
- [-days] 証明書に与える有効期間（日数）
- [-selfsign] 自己署名する

使用例:

```
wolfssl ca -config ca.conf -in test.csr -out test.pem -md sha256 -selfsign -
    keyfile ./key
```

1.2.3 CRL コマンド

CA を指定して CRL ファイルを検証するために使用されます。または、CRL を フォーマット変換 [DER|PEM] することもできます。-out が指定されておらず、-noout が使用されていない場合、このコマンドは CRL を stdout に出力します。検証が成功すると「OK」を出力します。

引数：

- [-CAfile] CA 証明書ファイル
- [-inform] 入力フォーマット：pem あるいは der
- [-in] the CRL ファイル
- [-outform] 出力フォーマット：pem あるいは der
- [-out] 出力ファイル
- [-noout] 指定がある場合には出力しません

使用例:

```
wolfssl crl -CAfile ./certs/ca-cert.pem -in ./certs/crl.der -inform DER -noout
```

1.2.4 DSAPARAM コマンド

DSA パラメータと鍵の作成に使用されます。wolfSSL が `--enable-dsa` オプションを指定してコンパイルされていることを確認してください。

指定可能な引数:

- [-genkey] 新たに DSA 鍵を生成する
- [-in] 鍵生成に必要なパラメータを含んでいるファイル
- [-out] 出力先ファイル (デフォルト：stdout)
- [-noout] パラメータをプリントアウトしない

使用例:

```
wolfssl dsaparam -out dsa.params 1024
wolfssl dsaparam -in dsa.params -genkey
```

1.2.5 DGST コマンド

署名を検証することが可能です。最後の引数は署名対象となったデータです。

サポートしているハッシュアルゴリズム:

- [-sha]

- [-sha224]
- [-sha256]
- [-sha384]
- [-sha512]

署名

引数:

- [-sign] 署名作成に必要な鍵
- [-out] 署名出力先のファイル

使用例:

```
wolfssl dgst -sign keyPrivate.pem -out test.sig testfile
```

検証

引数:

- [-verify] 署名を検証する為に使用する鍵
- [-signature] 署名を含んだファイル

使用例:

```
wolfssl dgst -verify keyPublic.pem -signature test.sig testfile
```

1.2.6 DHPARAM コマンド

デフィー・ヘルマンパラメータと鍵の生成に使用されます。

引数:

- [-genkey] 新たな DH 鍵を生成します
- [-in] 鍵生成のためのパラメータを読み取るファイル
- [-out] 出力ファイル（デフォルトは stdout)
- [-check] 生成されたパラメータが有効かチェックする
- [-noout] パラメータをプリントしない

使用例:

```
wolfssl dhparam -check -out dh.params 1024
```

1.2.7 ECPARAM コマンド

ECC 鍵生成に使用します。

指定可能な引数：

- [-genkey] 新しい鍵を生成する
- [-out] 出力先ファイル
- [-name] 楕円曲線名（secp384r1 等）

使用例:

```
wolfssl ecparam -genkey -out new.key -name secp384r1
```

1.2.8 ENC コマンド

入力の暗号化に使用され、(-d) で復号することもできます。

指定可能な暗号化と復号アルゴリズム：

- aes-cbc-128
- aes-cbc-192
- aes-cbc-256
- aes-ctr-128
- aes-ctr-192
- aes-ctr-256
- 3des-cbc-56
- 3des-cbc-112
- 3des-cbc-168

指定可能な引数：

- [-in] 対象の入力ファイル
- [-out] 出力先ファイル（デフォルト：stdout)
- [-pwd] パスワード
- [-key] 鍵データ（hex)
- [-iv] 鍵初期化ベクトル (hex)
- [-inkey] 鍵ファイル
- [-pbkdf2] KDF version2 を使用する
- [-md] ハッシュアルゴリズムを指定（md5, sha256 等）
- [-d] 入力ファイルを復号する
- [-p] デバッグ出力 (key / iv 等) をプリントアウトする
- [-k] パスワード入力のオプション
- [-base64] base64 エンコードされている入力を処理する
- [-nosalt] KDF にソルトを使用しない

使用例:

```
wolfssl enc -aes-128-cbc -k Thi$i$myPa$$w0rd -in somefile.txt
```

1.2.9 GENKEY コマンド

RSA、ECC、ED25519、および DSA 鍵の生成に使用されます。-output KEY を使用すると、-out 引数で与えたファイル名に.priv が追加された秘密鍵ファイルあるいは.pub が追加された公開鍵ファイルが作成されます。ED25519 鍵を生成する場合には、wolfSSL を –enable-ed25519 でコンパイルしておく必要があります。

指定可能な引数：

- [-out] 出力先ファイル
- [rsa | ecc | ed25519] 生成する鍵のタイプ
- [-inkey] 入力ファイル
- [-size] 生成する鍵のサイズ（ビット数）
- [-outform] 出力形式（DER あるいは PEM）(デフォルト：DER)
- [-output] 生成する鍵（PUB, PRIV あるいは KEYPAIR）(デフォルト：KEYPAIR)
- [-exponent] RSA 指数サイズ

使用例:

```
wolfssl genkey rsa -size 2048 -out mykey -outform pem  -output KEYPAIR
```

1.2.10 HASH コマンド

入力データのハッシュを生成します。

サポートしているハッシュアルゴリズム:

- md5
- sha
- sha256
- sha384
- sha512
- base64enc
- base64dec

使用例:

```
wolfssl -hash sha -in <some file>
```

1.2.11 MD5 コマンド

入力データの MD5 ハッシュを作成するために使用されます。最後の引数は、ハッシュ対象のファイルです。ファイル引数が使用されていない場合、ハッシュ対象のデータは stdin から取得します。

使用例:

```
wolfssl md5 configure.ac
978425cba5277d73db2a76d72b523d48

echo "hi" | wolfssl md5
764efa883dda1e11db47671c4a3bbd9e
```

1.2.12 PKCS12 コマンド

現在、PKCS12 の解析のみがサポートされており、PKCS12 の生成はまだサポートされていません。デフォルトでは、wolfSSL をビルドするときに使用される –enable-wolfclu オプションは PKCS12 サポートも有効にしますが、RC2 は有効にしません。RC2 を使用して暗号化された PKCS12 バンドルを解析する場合、wolfSSL をコンパイルするときに –enable-rc2 も使用する必要があります。

引数:

- [-in] file input for pkcs12 bundle
- [-out] 処理結果出力先のファイル (デフォルト：stdout)
- [-nodes] DES 暗号化を使用しない
- [-nocerts] 証明書をプリントアウトしない
- [-nokeys] 鍵をプリントアウトしない
- [-passin] パスワードを含んだファイル
- [-passout] パスワードの出力先ファイル

使用例:

```
./wolfssl pkcs12 -nodes -passin pass:"wolfSSL test" -in ./certs/test-
   servercert.p12
```

1.2.13 PKEY コマンド

一般的な鍵操作を処理するために使用されます。読み込まれた鍵を stdout に出力します。

引数:

- [-in] 入力される鍵ファイル
- [-inform] 入力ファイル形式：pem あるいは der (デフォルト：pem)
- [-pubout] 公開鍵のみプリントアウトする
- [-pubin] 公開鍵を入力として期待する

使用例:

```
./wolfssl pkey -in ./certs/server-key.pem -inform pem -pubout
```

1.2.14 RAND コマンド

raw または base64 形式でランダムデータのバイト列を生成します。デフォルトでは、結果を stdout に出力しますが、'-out' 引数を使用してリダイレクトできます。渡される最後の引数は、生成するランダムデータのバイト数です。

引数:

- [-base64] ランダムデータを base64 エンコードする
- [-out] 結果を出力するファイル

使用例:

```
wolfssl rand -base64 10
```

1.2.15 REQ コマンド

証明書要求または自己署名証明書の作成に使用されます。証明書をセットアップするための.conf ファイルのいくつかの基本的な解析を処理できます。構成ファイルが使用されていない場合、stdin は証明書情報の入力を求められます。

指定可能な引数:

- [-in] 入力ファイル
- [-out] 出力先ファイル (デフォルト：stdout)
- [-key] 証明書要求に含める公開鍵ファイル
- [-inform] 入力ファイル形式：pem あるいは der (デフォルト：pem)
- [-outform] 出力ファイル形式：pem あるいは der (デフォルト：pem)
- [-config] 証明書のコンフィグレーションファイル
- [-days] 有効期間（日数）
- [-x509] 自己署名証明書を生成する

使用例:

```
wolfssl ecparam -genkey -out ecc.key -name secp384r1
wolfssl req -new -x509 -days 3650 -config selfsigned.conf -key ecc.key -out
    ecc.cert \
-outform der -sha256
```

1.2.16 RSA コマンド

RSA 操作を行います。RSA 鍵の読み取り、RSA 鍵またはモジュラスの出力、暗号化された PEM ファイルの読み取りが含まれます。入力と出力の DER と PEM 形式の両方を処理できます。

引数:

- [-in] 入力となる鍵ファイル
- [-inform] 入力ファイル形式：PEM あるいは DER (デフォルト：PEM)
- [-out] 出力ファイル (デフォルト：stdout)

- [-outform] 出力ファイル形式：PEM あるいは DER (デフォルト：PEM)
- [-passin] PEM 形式の暗号化されたファイルのパスワード
- [-noout] 鍵をプリントアウトしない
- [-modulus] RSA modulus (n value) をプリントする
- [-RSAPublicKey_in] 公開鍵入力を期待する

1.2.17 sha256, sha384, and sha512 コマンド

各コマンドを使用して、指定タイプのハッシュを作成できます。例えば -sha256 は sha256 ハッシュを生成します。コマンドは、stdin または指定された入力ファイルの形式で入力を受け入れます。

使用例：

```
wolfssl -sha384 <some file>
```

1.2.18 S_CLIENT コマンド

基本的な TLS 接続がサポートされています。現在、ピアを検証していません。-CAfile オプションはまだ完了していません。

引数：

- [-connect] ip アドレス:port 番号

使用例:

```
wolfssl s_client -connect 127.0.0.1:11111
```

1.2.19 VERIFY コマンド

CA が指定された X509 証明書を検証します。コマンドに渡される最後の引数は、検証する証明書ファイルの名前です。検証が成功すると、「OK」が stdout に出力されます。それ以外の場合、エラー値と理由が出力されます。

引数:

- [-CAfile] 検証に使用する CA 証明書ファイル
- [-crl_check] CRL 検証が必要

使用例:

```
wolfssl verify -CAfile ./certs/ca-cert.pem ./certs/server-cert.pem
```

1.2.20 X509 コマンド

このコマンドは、証明書の解析と出力に使用されます。

引数:

- [-in] 入力となる X509 証明書ファイル
- [-inform] 入力ファイル形式：pem あるいは der (デフォルト：pem)
- [-out] 出力先ファイル
- [-outform] 出力ファイル形式：pem あるいは der(デフォルト：pem)
- [-pubkey] 公開鍵のみ出力する
- [-text] 証明書を出力する

使用例:

```
wolfssl x509 -in ./certs/server-cert.pem -text
```