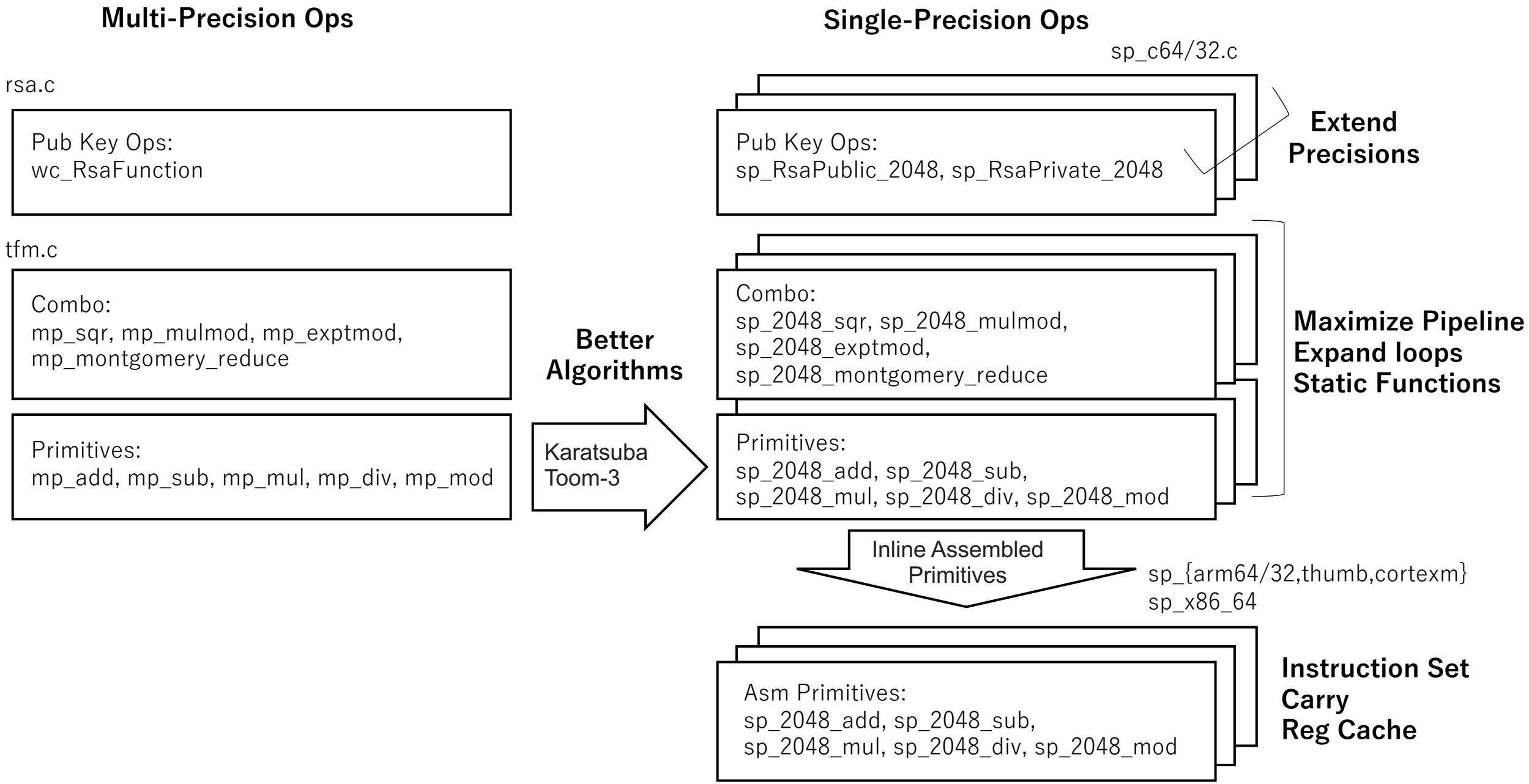
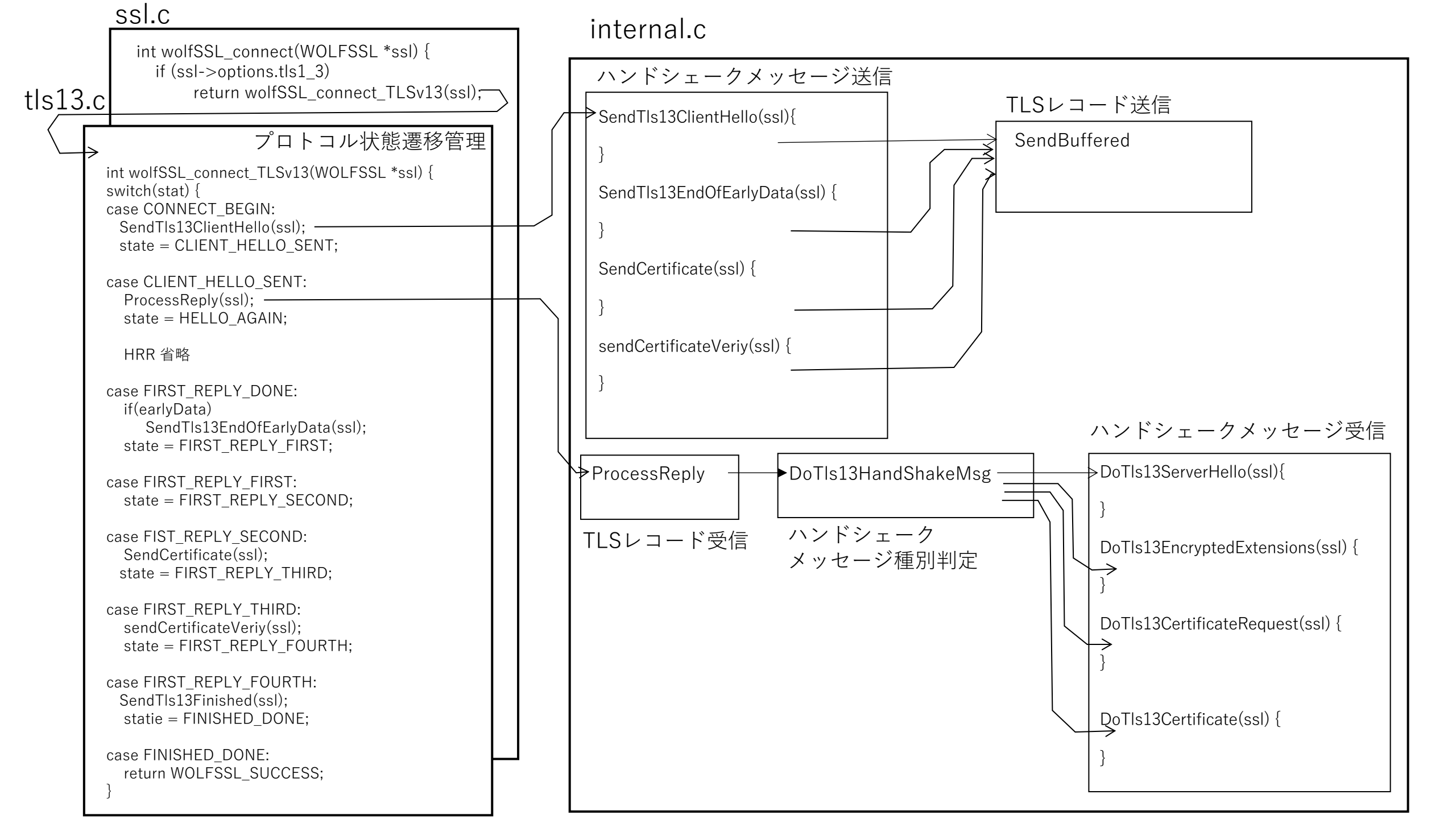
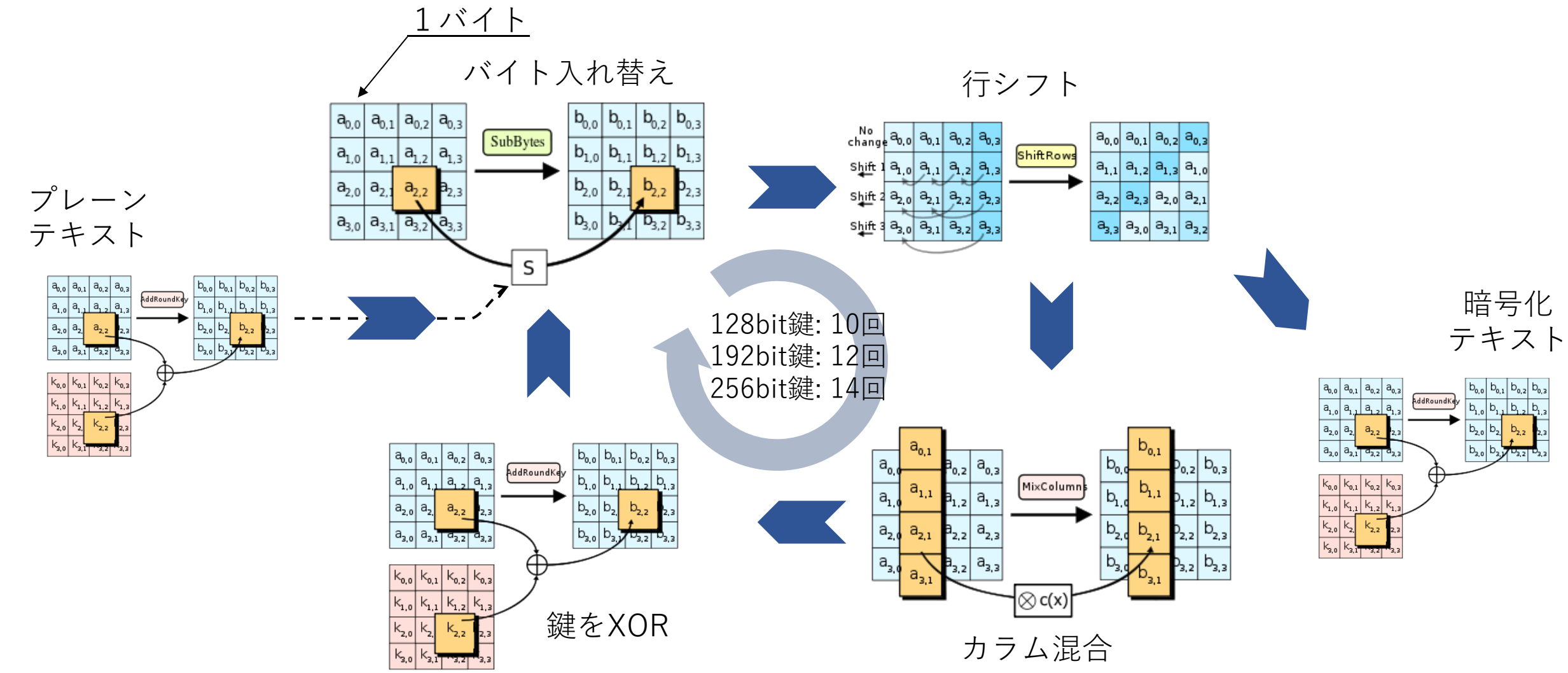
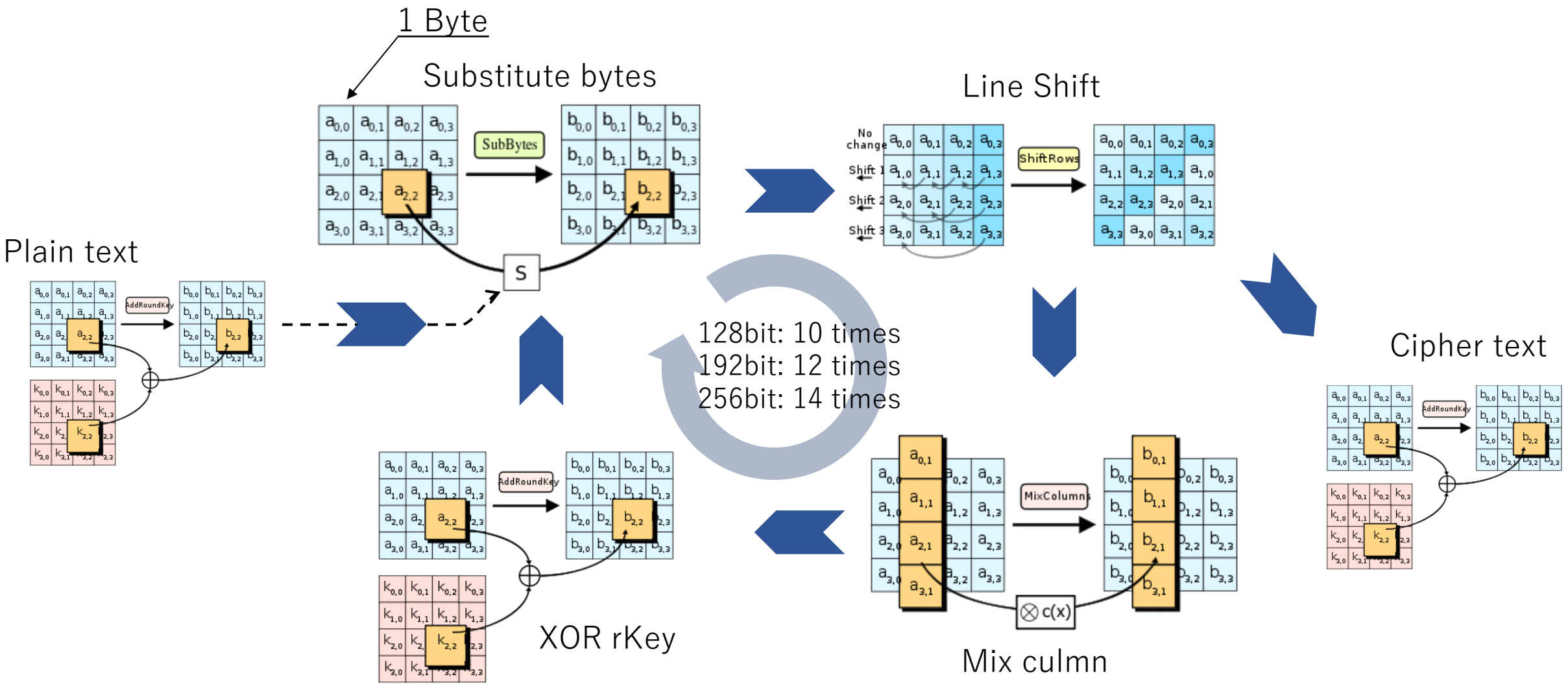
ファイル構成

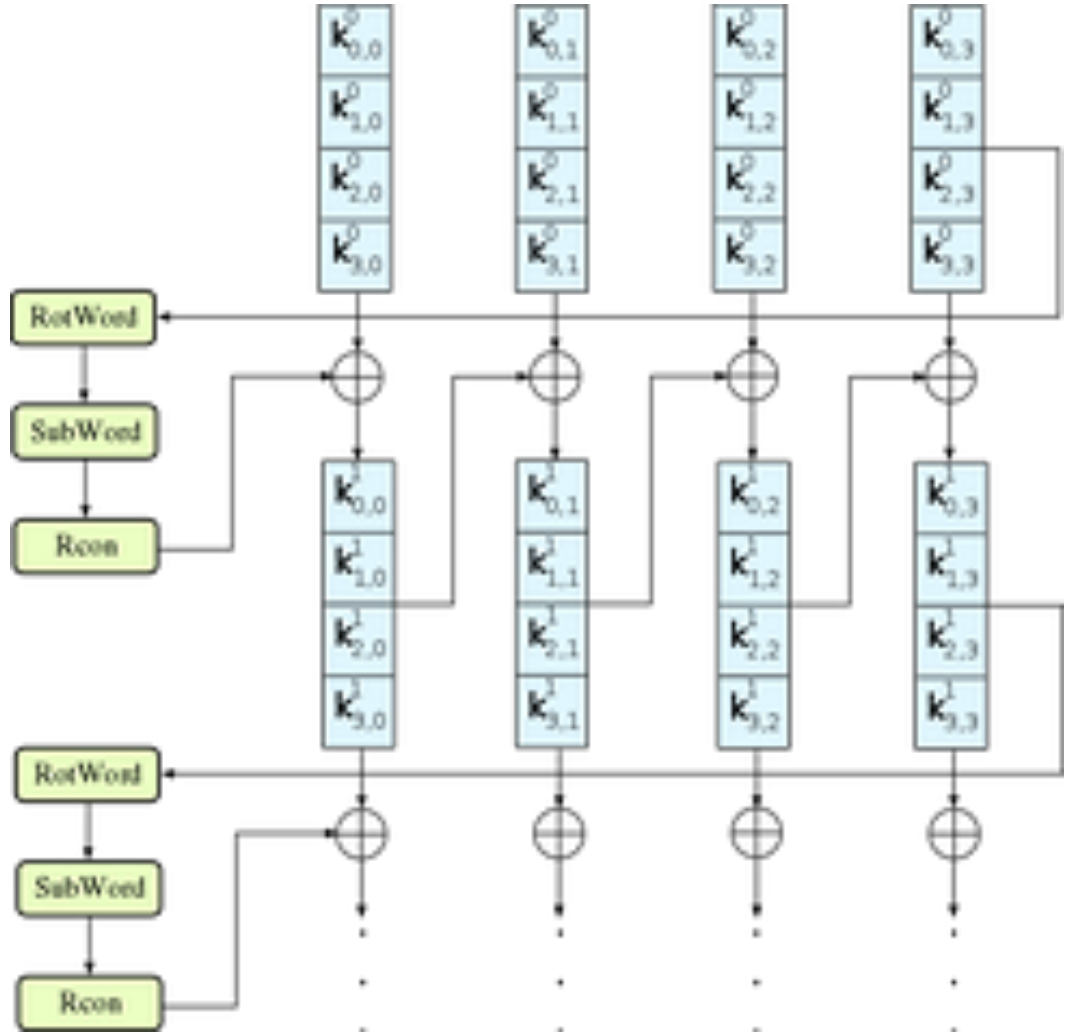
項　目		ディレクトリ名
プログラム	SSL/TLS層	**src/**
	暗号エンジン層	wolfcrypt/src
ヘッダーファイル	SSL/TLS層	wolfssl/
	暗号エンジン層	wolfssl/wolfcrypt
テスト		tests/
wolfCryptテストプログラム		wolfcrypt/test
ベンチマークプログラム		wolfcrypt/benchmark
サンプルプログラム		examples/client examples/server その他
IDEプロジェクト、設定ファイル（IAR, MDK ARM, XCODE…）		IDE/
テスト用証明書, 鍵		certs/*.{pem,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


Multi-Precision Ops
rsa.c
Pub Key Ops:
wc_RsaFunction
tfm.c
Combo:
mp_sqr, mp_mulmod, mp_exptmod,
mp_montgomery_reduce
Primitives:
mp_add, mp_sub, mp_mul, mp_div, mp_mod
Better
Algorithms
Karatsuba
Toom-3
Single-Precision Ops
sp_c64/32.c
Pub Key Ops:
sp_RsaPublic_2048, sp_RsaPrivate_2048
Extend
Precisions
Combo:
sp_2048_sqr, sp_2048_mulmod,
sp_2048_exptmod,
sp_2048_montgomery_reduce
Primitives:
sp_2048_add, sp_2048_sub,
sp_2048_mul, sp_2048_div, sp_2048_mod
Maximize Pipeline
Expand loops
Static Functions
Inline Assembled
Primitives
sp_{arm64/32,thumb,cortexm}
sp_x86_64
Asm Primitives:
sp_2048_add, sp_2048_sub,
sp_2048_mul, sp_2048_div, sp_2048_mod
Instruction Set
Carry
Reg Cache


ssl.c
int wolfSSL_connect(WOLFSSL *ssl) {
if (ssl->options.tls1_3)
return wolfSSL_connect_TLSv13(ssl);
tls13.c
プロトコル状態遷移管理
int wolfSSL_connect_TLSv13(WOLFSSL *ssl) {
switch(stat) {
case CONNECT_BEGIN:
SendTls13ClientHello(ssl);
state = CLIENT_HELLO_SENT;
case CLIENT_HELLO_SENT:
ProcessReply(ssl);
state = HELLO_AGAIN;
HRR 省略
case FIRST_REPLY_DONE:
if(earlyData)
SendTls13EndOfEarlyData(ssl);
state = FIRST_REPLY_FIRST;
case FIRST_REPLY_FIRST:
state = FIRST_REPLY_SECOND;
case FIST_REPLY_SECOND:
SendCertificate(ssl);
state = FIRST_REPLY_THIRD;
case FIRST_REPLY_THIRD:
sendCertificateVeriy(ssl);
state = FIRST_REPLY_FOURTH;
case FIRST_REPLY_FOURTH:
SendTls13Finished(ssl);
statie = FINISHED_DONE;
case FINISHED_DONE:
return WOLFSSL_SUCCESS;
}
internal.c
ハンドシェークメッセージ送信
SendTls13ClientHello(ssl){
}
SendTls13EndOfEarlyData(ssl) {
}
SendCertificate(ssl) {
}
sendCertificateVeriy(ssl) {
}
TLSレコード送信
SendBuffered
ProcessReply
TLSレコード受信
DoTls13HandShakeMsg
ハンドシェーク
メッセージ種別判定
ハンドシェークメッセージ受信
DoTls13ServerHello(ssl){
}
DoTls13EncryptedExtensions(ssl) {
}
DoTls13CertificateRequest(ssl) {
}
DoTls13Certificate(ssl) {
}


１バイト
バイト入れ替え
SubBytes
S
行シフト
No change
Shift 1
Shift 2
Shift 3
ShiftRows
プレーン
テキスト
AddRoundKey
128bit鍵: 10回
192bit鍵: 12回
256bit鍵: 14回
暗号化
テキスト
AddRoundKey
鍵をXOR
MixColumns
c(x)
カラム混合


1 Byte
Substitute bytes
SubBytes
S
Line Shift
No change
Shift 1
Shift 2
Shift 3
ShiftRows
Plain text
AddRoundKey
128bit: 10 times
192bit: 12 times
256bit: 14 times
Cipher text
AddRoundKey
XOR rKey
MixColumns
c(x)
Mix culmn


RotWord
SubWord
Rcon
RotWord
SubWord
Rcon