



Vorbemerkung zu Projekt 197 „Sichere Implementierung einer allgemeinen Kryptobibliothek – Arbeitspaket 1: Sichtung und Analyse bestehender Kryptobibliotheken“

Zielsetzung bei der Untersuchung der Kryptobibliotheken

Ziel des Projektes „Sichere Implementierung einer allgemeinen Kryptobibliothek“ ist die Bereitstellung einer quelloffenen, sicheren, übersichtlichen, kontrollierbaren und gut dokumentierten Kryptobibliothek, die für möglichst viele Einsatzszenarien des BSI geeignet ist und auch in Anwendungen mit erhöhtem Sicherheitsbedarf eingesetzt werden kann.

In der (hier im Ergebnis vorliegenden) ersten Projektphase wurden bestehende Open Source Kryptobibliotheken gesichtet und analysiert, um einen für das BSI geeigneten Kandidaten für eine Weiterentwicklung zu identifizieren. Dazu wurde eine Liste von Kriterien mit Gewichtungen zusammengestellt, die spezifisch im Hinblick auf das Gesamtprojekt ausgewählt wurden. Der Fokus des Projekts war dabei klar auf eine Vorarbeit für die folgende Entwicklung einer sicheren Kryptobibliothek beschränkt. So stellt die folgende Sichtung der Bibliotheken keine eigenständige Studie zu Sicherheitseigenschaften von Kryptobibliotheken dar, da sowohl die Vorauswahl der Bibliotheken als auch die Methodik und die Aufbereitung der Analyse sowie die Ermittlung und Untersuchung der Kandidaten für die Weiterentwicklung nur auf den internen Gebrauch im Rahmen der Durchführung des Gesamtprojekts ausgerichtet war. Die vorliegenden Projektergebnisse beruhen somit nicht auf einer allgemeingültigen oder vollumfänglichen Untersuchung und sind insbesondere keine offizielle Bewertung von Open Source Kryptobibliotheken durch das BSI.

Mit der im Projektverlauf weiterentwickelten Bibliothek Botan (siehe <https://github.com/Rohde-Schwarz-Cybersecurity/botan>) steht jetzt eine Kryptobibliothek zur Verfügung, die den grundsätzlichen Vorstellungen des BSI entspricht und die zum Einsatz in Sicherheitsprodukten empfohlen wird. Botan steht unter der Simplified BSD Lizenz und kann freizügig sowohl für offene als auch für kommerzielle Anwendungen verwendet werden. Insbesondere sind auf GitHub alle Dokumente zu finden, die im Rahmen der Weiterentwicklung im Projekt entstanden sind und die Sicherheitseigenschaften von Botan ausführlich dokumentieren.



Bundesamt
für Sicherheit in der
Informationstechnik

Eine Studie im Auftrag des
Bundesamtes für Sicherheit in der Informationstechnik

Projekt 197

Sichere Implementierung einer allgemeinen Kryptobibliothek

Arbeitspaket 1:
Sichtung und Analyse bestehender Kryptobibliotheken

Version 1.0.0 / 2017-02-15

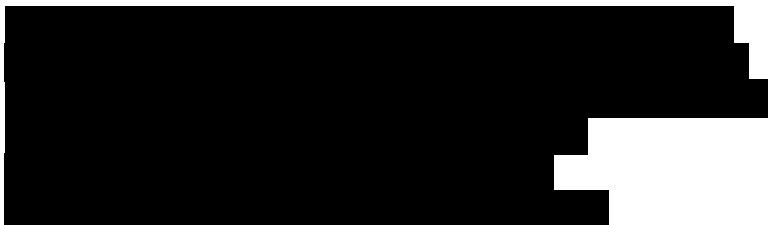


Zusammenfassung

In diesem Projekt wird die sichere Implementierung einer allgemeinen Kryptobibliothek realisiert, die alle gängigen und für den breiten kryptographischen Einsatz notwendigen kryptographischen Primitive beinhaltet. Dazu zählen unter anderem symmetrische und asymmetrische Verschlüsselungs- und Signaturverfahren, PRFs, Hashfunktionen und RNGs. Zusätzlich müssen verbreitete Sicherheitsstandards wie X.509-Zertifikate oder SSL/TLS unterstützt werden. Diese Bibliothek soll dann auch Herstellern von VS-Produkten zur Verfügung stehen, womit der Evaluierungsaufwand des BSI für solche Produkte erheblich reduziert wird.

Dieser Bericht ist das Ergebnis der Sichtung und Analyse bestehender Kryptobibliotheken. Es wird zunächst die Untersuchungsmethodik erläutert. Anschließend werden die Analyseergebnisse vorgestellt und es wird ein geeigneter Kandidat für eine Weiterentwicklung präsentiert. Die Analyse wurde im Zeitraum August 2015 bis November 2015 durchgeführt.

Autoren



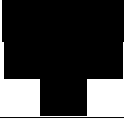
Copyright

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urhebergesetzes ist ohne Zustimmung des BSI unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigung, Übersetzung, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Sichere Implementierung einer allgemeinen Kryptobibliothek

Sichtung und Analyse bestehender Kryptobibliotheken

Änderungshistorie

Version	Autor	Kommentar	Datum
1.0		Initiale Dokumentenversion erstellt	2017-02-15

Inhaltsverzeichnis

1	Vorauswahl der Kryptobibliotheken.....	13
1.1	Methodik.....	14
1.2	Analyse.....	18
1.2.1	Beecrypt.....	18
1.2.2	Botan.....	21
1.2.3	Cryptlib.....	24
1.2.4	Crypto++.....	27
1.2.5	GnuTLS.....	30
1.2.6	Libgcrypt.....	33
1.2.7	LibreSSL.....	36
1.2.8	Libsodium.....	39
1.2.9	Libtomcrypt.....	42
1.2.10	MatrixSSL.....	45
1.2.11	mbed TLS.....	48
1.2.12	NaCl.....	51
1.2.13	Nettle.....	54
1.2.14	NSS.....	57
1.2.15	OpenSSL.....	60
1.2.16	Poco.....	63
1.2.17	s2n.....	66
1.2.18	WolfSSL.....	69
1.3	Ergebnis.....	73
2	Detaillierte Analyse der vorausgewählten Bibliotheken.....	75
2.1	Methodik.....	75
2.2	Analyse.....	75
2.2.1	Implementierung von SCA-Gegenmaßnahmen.....	75
2.2.1.1	RSA-Blinding.....	75
	Botan.....	75
	LibreSSL.....	76
	NSS.....	77
2.2.1.2	Bleichenbacher-Angriff.....	77
	Botan.....	78
	LibreSSL.....	79
	NSS.....	80
2.2.1.3	Invalid Curve Angriffe.....	81
	Botan.....	81
	LibreSSL.....	81
	NSS.....	82
2.2.1.4	Small Subgroup Angriffe.....	82
	Botan.....	82
	LibreSSL.....	83
	NSS.....	83
2.2.1.5	Padding Oracle Angriff auf CBC.....	83
	Botan.....	83
	LibreSSL.....	84
	NSS.....	84
2.2.2	Portierbarkeit.....	84
	Botan.....	84

LibreSSL.....	85
NSS.....	85
2.2.3 Design und Komplexität.....	85
2.2.3.1 Botan.....	86
2.2.3.2 LibreSSL.....	86
2.2.3.3 NSS.....	87
2.2.3.4 Komplexität der Funktionen.....	87
2.2.4 Performance.....	88
2.2.5 Zufallszahlengenerator.....	89
Botan.....	89
LibreSSL.....	89
NSS.....	90
2.2.6 PKCS-Unterstützung.....	90
Botan.....	90
LibreSSL.....	90
NSS.....	91
2.2.7 Testabdeckung.....	91
Botan.....	91
LibreSSL.....	91
NSS.....	91
2.2.8 Dokumentation.....	92
Botan.....	92
LibreSSL.....	92
NSS.....	93
2.2.9 Mechanismen zum sicheren Löschen von Geheimnissen.....	93
2.2.9.1 Botan.....	93
2.2.9.2 LibreSSL.....	94
2.2.9.3 NSS.....	95
2.2.10 Fehlende Algorithmen.....	95
2.2.10.1 Botan.....	95
2.2.10.2 LibreSSL.....	96
2.2.10.3 NSS.....	96
2.2.11 Nicht gewünschte Algorithmen.....	97
2.2.12 Fehler in Zertifikatsvalidierung.....	97
2.2.12.1 Botan.....	98
2.2.12.2 LibreSSL.....	99
2.2.12.3 NSS.....	101
2.2.13 Fehler in TLS-Protokollvalidierung.....	101
2.2.14 Statische Analyse.....	101
2.2.14.1 Warnungen.....	102
2.2.14.2 Cppcheck.....	103
2.2.14.3 Clang-Analyzer.....	104
2.2.15 Continuous Integration.....	104
2.2.15.1 Botan.....	104
2.2.15.2 LibreSSL.....	104
2.2.15.3 NSS.....	105
2.2.16 Kompatibilität mit der OpenSSL-Schnittstelle.....	105
2.3 Auswertung.....	105
3 Anhang: Messergebnisse.....	111
3.1 Komplexität.....	111

3.1.1 Botan.....	111
3.1.2 LibreSSL.....	111
3.1.3 NSS.....	112
3.2 Testabdeckung.....	113
3.2.1 Botan.....	113
3.2.2 LibreSSL.....	132
3.2.3 NSS.....	151
3.3 Fehler in Zertifikatsvalidierung.....	164
3.3.1 Botan.....	164
3.3.2 LibreSSL.....	167
3.3.3 NSS.....	169
3.4 Fehler in der TLS-Protokollvalidierung.....	172

1 Vorauswahl der Kryptobibliotheken

In diesem Kapitel wird eine Übersicht über existierende Kryptobibliotheken erstellt und anschließend anhand bestimmter Kriterien auf ihre Eignung für eine Weiterentwicklung im Rahmen des Projektes hin untersucht. Dazu wurde eine Liste von elementaren funktionalen als auch nichtfunktionalen Kriterien festgelegt, die einfach in den Bibliotheken zu identifizieren sind und eine schnelle Auswahl ermöglichen. Ziel ist es, maximal zwei bis drei Kryptobibliotheken für eine detailliertere Analyse zu identifizieren.

1.1 Methodik

Die geforderten kryptographischen Algorithmen und Protokolle leiten sich aus der in der Leistungsbeschreibung genannten Technischen Richtlinie BSI-TR-02102 ab. Dazu gehören Paddingverfahren, Modes of Operation, Hashfunktionen, Schlüsselherleitungsfunktionen, TLS-Unterstützung, symmetrische und asymmetrische Verfahren, Elliptische Kurven, MACs und X.509-Unterstützung. Da auch die Unterstützung von Smartcards gewünscht ist, ist eine PKCS#11-Schnittstelle zur Einbindung kryptographischer Hardware wünschenswert.

Zu den nicht-funktionalen Kriterien gehören eine statische Registrierung von Implementierungen, die Lizenzbestimmungen, ein Ansprechpartner für Sicherheitsfragen, die Aktivität des/der Entwicklerteams, die Häufigkeit von Releases, die Kompatibilität mit der OpenSSL-Schnittstelle, das Vorhandensein einer Testsuite mit Positiv- und Negativtests, Bindings für andere Programmiersprachen, die Portabilität und die Qualität der Dokumentation und Handbücher.

Jedem Kriterium wird ein Punktwert zugeordnet, je nach Wichtigkeit für eine Eignung der Bibliothek. Normal gewichtete Kriterien werden mit dem Punktwert 1 gewertet, höher gewichtete Kriterien mit dem Punktwert 2 und niedriger gewichtete Kriterien mit dem Punktwert 0,5. Bei den nicht-funktionalen Kriterien kann es zu Abwertungen kommen, beispielsweise wenn zwar eine API-Dokumentation vorhanden ist, ein Handbuch jedoch fehlt (Abwertung um einen Punkt). Für jede Bibliothek wird dann die Summe aller Punkte gebildet und mit den anderen Bibliotheken verglichen. Die vollständige Liste der Kriterien und zugehörigen Punktwertungen findet sich in der folgenden Tabelle.

Kriterium	Maximale Punktzahl
Paddingverfahren	
OAEP	1
PSS	1
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	1
XTS	0,5
CBC	1
CTR	1
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	1
TLS Peer	
Client	1
Server	1
TLS Version	
TLS 1.0	0,5
TLS 1.1	0,5
TLS 1.2	1
DTLS 1.0	0,5
DTLS 1.2	0,5
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	1
TLS-DHE	1
TLS-PSK	1
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0,5
DLIES	0,5

Asymm. Signatur	
RSA	1
DSA	1
ECDSA	1
ECKDSA	1
ECGDSA	1
Merkle-Signaturen	0,5
Elliptische Kurven	
Brainpool	1
NIST	0,5
Eigene Kurven (BSI-Kurven)	2
MAC	
HMAC	1
CMAC	1
GMAC	1
X.509	
Kettenvalidierung	1
Generierung	1
CRL	1
OCSP	1
DANE	1
PKCS#11-Schnittstelle	1
TPM-Unterstützung	1
Weitere Kriterien	
Statische Registrierung von Implementierungen	2
Lizenz	-
Ansprechpartner für Sicherheitsfragen	2
Aktivität des/der Entwickler/Häufigkeit v. Releases	2
Kompatibilität mit der OpenSSL-Schnittstelle	1
Testsuite	
Positivtests	1
Negativtests	1
Bindings für andere Programmiersprachen	2
Portierbarkeit	
OS X	0,5

Linux	0,5
Windows (Visual Studio)	0,5
iOS	0,5
Android	0,5
Dokumentation/Handbücher	2
Summe:	56

Tabelle 1.1: Kriterien und Punktwertung

1.2 Analyse

In Zusammenarbeit mit dem BSI wurden 18 Bibliotheken für eine Analyse ausgewählt. Untersucht werden die folgenden Bibliotheken in der zum Zeitpunkt der Analyse aktuellsten Version: BeeCrypt (4.2.1), Botan (1.11.18), Cryptlib (3.4.3-beta), Crypto++ (5.6.2), GnuTLS (3.3.16), Libgrypt (1.6.3), LibreSSL (2.2.2), Libsodium (1.0.3), Libtomcrypt (1.17), MatrixSSL (3.7.2b), mbedTLS (2.0.0), NaCl (20110221), Nettle (3.1), NSS (3.19.2), OpenSSL (1.0.2d), Poco (1.6.1), s2n (1dbb4a3c59) und WolfSSL (3.6.0).

1.2.1 Beecrypt

BeeCrypt ist eine allgemeine Kryptobibliothek geschrieben in C und C++, die seit 1997 entwickelt wird. Sie unterstützt von den geforderten Paddingverfahren lediglich die PKCS#1 v1.5-Paddingverfahren. Als Modes of Operation fehlen GCM und XTS. Die Hashfunktionen der SHA-1 und SHA-2 Familien werden unterstützt. Eine KDF nach NIST SP800-56C ist nicht implementiert. Auf TLS-Funktionalität verzichtet die Bibliothek ganz. Es fehlen außerdem ECIES, DLIES, ECDSA, ECKDSA, ECGDSA, und Merkle-Signaturen. Elliptische Kurven fehlen gänzlich. Von den geforderten MACs werden CMAC und GMAC nicht unterstützt. X.509 wird als Standard, genauso wie PKCS#11, nicht unterstützt. Unterstützung für TPMs ist ebenfalls nicht vorhanden.

Eine statische Registrierung von Implementierungen ist möglich, es sind jedoch nur ausgewählte Teile der Bibliothek deaktivierbar. BeeCrypt steht unter der LGPL-Lizenz. Ein Ansprechpartner für Sicherheitsfragen wird nicht konkret benannt. Das letzte Release 4.2.1 wurde am 12.07.2009 veröffentlicht. Der Releasezyklus ist sehr unregelmäßig (zwischen wenigen Monaten bis Jahren). BeeCrypt bietet keine OpenSSL-kompatible Schnittstelle. Eine Testsuite ist vorhanden, enthält jedoch fast ausschließlich Positivtests. Bindings für Python und Java stehen zur Verfügung. OS X, Linux und Windows werden als Plattformen unterstützt, für iOS und Android ist die Unterstützung unklar. Eine Doxygen API-Dokumentation ist vorhanden, ein Handbuch mit Beispielcode fehlt jedoch.

Kriterium	Punktwertung Beecrypt
Paddingverfahren	
OAEP	0
PSS	0
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	0
XTS	0
CBC	1
CTR	1
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	0
Server	0
TLS Version	
TLS 1.0	0
TLS 1.1	0
TLS 1.2	0
DTLS 1.0	0
DTLS 1.2	0
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	0
TLS-DHE	0
TLS-PSK	0
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	1
DSA	1
ECDSA	0
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	0
NIST	0
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	0
GMAC	0
X.509	
Kettenvalidierung	0
Generierung	0
CRL	0
OCSP	0
DANE	0
PKCS#11-Schnittstelle	0
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	1
Lizenz	LGPL
Ansprechpartner für Sicherheitsfragen	0
Aktivität des/der Entwickler/Häufigkeit v. Releases	0
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	1
Negativtests	0
Bindings für andere Programmiersprachen	2
Portierbarkeit	1,5
Dokumentation/Handbücher	1

Summe:	17,5/56
---------------	----------------

Tabelle 1.2: Auswertung BeeCrypt

1.2.2 Botan

Botan ist eine allgemeine Kryptobibliothek geschrieben in C++, die seit 2001 entwickelt wird. Unterstützt werden alle geforderten Paddingverfahren, Modes of Operation und Hashfunktionen. Eine KDF nach NIST SP800-56C ist nicht implementiert. TLS wird als Client und als Server unterstützt, und das in den Versionen TLS 1.0, 1.1 und 1.2 und DTLS 1.0 und DTLS 1.2. Von den in TR-02102-2 gestatteten Cipher Suites werden mit TLS-ECDHE, TLS-DHE und TLS-PSK alle unterstützt. AES, RSA, DLIES und ECDSA werden von Botan unterstützt. Es fehlen jedoch ECIES, ECKDSA, ECGDSA und Merkle-Signaturen. Es werden die Brainpool- und die NIST-Kurven unterstützt, sowie auch eigene Kurven (über GF(p)). Lediglich HMAC steht als MAC zur Verfügung, während GMAC nicht unterstützt wird. Für X.509 wird die Kettenvalidierung und Generierung unterstützt, außerdem auch CRL und OCSP. DANE wird nicht unterstützt. Eine PKCS#11-Schnittstelle ist derzeit nicht vorhanden. Unterstützung für TPMs ist ebenfalls nicht vorhanden.

Eine statische Registrierung von Implementierungen ist möglich. Die Bibliothek ist sehr modular aufgebaut, sodass über 50 Module einzeln deaktiviert werden können. Botan steht unter der Simplified BSD-Lizenz. Ein Ansprechpartner für Sicherheitsfragen wird nicht konkret benannt. Das letzte Release 1.11.18 wurde am 5.7.2015 veröffentlicht. Neue Releases erscheinen in ein- bis dreimonatigen Abständen. Botan bietet keine OpenSSL-kompatible Schnittstelle. Eine Testsuite ist vorhanden und enthält Positiv- wie Negativtests. Bindings für Python, C89 und OCaml stehen zur Verfügung. Alle geforderten Plattformen werden von Botan unterstützt. Eine Doxygen API-Dokumentation ist vorhanden, auch ein Handbuch mit Beispielcode steht zur Verfügung.

Kriterium	Punktwertung Botan
Paddingverfahren	
OAEP	1
PSS	1
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	1
XTS	0,5
CBC	1
CTR	1
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	1
Server	1
TLS Version	
TLS 1.0	0,5
TLS 1.1	0,5
TLS 1.2	1
DTLS 1.0	0,5
DTLS 1.2	0,5
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	1
TLS-DHE	1
TLS-PSK	1
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0
DLIES	0,5

Asymm. Signatur	
RSA	1
DSA	1
ECDSA	1
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	1
NIST	0,5
Eigene Kurven (BSI-Kurven)	2
MAC	
HMAC	1
CMAC	1
GMAC	0
X.509	
Kettenvalidierung	1
Generierung	1
CRL	1
OCSP	1
DANE	0
PKCS#11-Schnittstelle	0
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	2
Lizenz	Simplified BSD
Ansprechpartner für Sicherheitsfragen	0
Aktivität des/der Entwickler/Häufigkeit v. Releases	2
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	1
Negativtests	1
Bindings für andere Programmiersprachen	2
Portierbarkeit	2,5
Dokumentation/Handbücher	2

Summe:	45/56
--------	-------

Tabelle 1.3: Auswertung Botan

1.2.3 Cryptlib

Cryptlib ist eine allgemeine Kryptobibliothek geschrieben in C, die seit 1995 entwickelt wird. Sie unterstützt von den geforderten Paddingverfahren nicht das PSS-Paddingverfahren. Als Modes of Operation fehlt XTS, während GCM, CBC und CTR vorhanden sind. Die Hashfunktionen der SHA-1 und SHA-2 Familien werden unterstützt. Eine KDF nach NIST SP800-56C ist nicht implementiert. TLS wird als Client und als Server unterstützt, und das in den Versionen TLS 1.0, 1.1 und 1.2. DTLS wird nicht unterstützt. Von den in TR-02102-2 gestatteten Cipher Suites werden mit TLS-ECDHE, TLS-DHE und TLS-PSK alle unterstützt. AES, RSA, DSA und ECDSA werden unterstützt, es fehlt jedoch die Unterstützung von ECIES, DLIES, ECKDSA, ECGDSA und Merkle-Signaturen. Es werden die Brainpool- und NIST-Kurven unterstützt, jedoch nicht eigene Kurven. Von den geforderten MACs wird nur HMAC, nicht aber CMAC und GMAC nicht unterstützt. Für X.509 wird die Kettenvalidierung und Generierung unterstützt, außerdem auch CRL und OCSP. DANE wird nicht unterstützt. Cryptlib unterstützt die Einbindung von kryptographischer Hardware über eine PKCS#11-Schnittstelle. Eine TPM-Unterstützung ist nicht gegeben.

Eine statische Registrierung von Implementierungen ist möglich, es sind jedoch nur ausgewählte Teile der Bibliothek deaktivierbar. Cryptlib steht unter der Berkeley DB-Lizenz. Ein Ansprechpartner für Sicherheitsfragen wird konkret benannt, jedoch wird kein PGP-Key zur verschlüsselten Kommunikation angegeben. Das letzte Release 3.4.2 wurde am 17.12.2012 veröffentlicht. Cryptlib bietet keine OpenSSL-kompatible Schnittstelle. Eine Testsuite ist vorhanden und enthält Positiv- wie Negativtests. Bindings für C++, C#/.NET, Delphi, Java, Python und Visual Basic stehen zur Verfügung. OS X, Linux und Windows werden als Plattformen unterstützt, für iOS und Android ist die Unterstützung unklar. Eine Doxygen API-Dokumentation ist vorhanden, sowie auch ein ausführliches Handbuch mit Beispielcode.

Kriterium	Punktwertung Cryptlib
Paddingverfahren	
OAEP	1
PSS	0
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	1
XTS	0
CBC	1
CTR	1
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	1
Server	1
TLS Version	
TLS 1.0	0,5
TLS 1.1	0,5
TLS 1.2	1
DTLS 1.0	0
DTLS 1.2	0
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	1
TLS-DHE	1
TLS-PSK	1
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	1
DSA	1
ECDSA	1
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	1
NIST	0,5
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	0
GMAC	0
X.509	
Kettenvalidierung	1
Generierung	1
CRL	1
OCSP	1
DANE	0
PKCS#11-Schnittstelle	1
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	1
Lizenz	Berkeley DB
Ansprechpartner für Sicherheitsfragen	1
Aktivität des/der Entwickler/Häufigkeit v. Releases	1
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	1
Negativtests	1
Bindings für andere Programmiersprachen	2
Portierbarkeit	1,5
Dokumentation/Handbücher	2

Summe:	38/56
---------------	--------------

Tabelle 1.4: Auswertung Cryptlib

1.2.4 Crypto++

Crypto++ ist eine allgemeine Kryptobibliothek geschrieben in C++, die seit 1995 entwickelt wird. Sie unterstützt alle geforderten Paddingverfahren. Als Modes of Operation fehlt XTS, während GCM, CBC und CTR vorhanden sind. Die Hashfunktionen der SHA-1 und SHA-2 Familien werden unterstützt. Eine KDF nach NIST SP800-56C ist nicht implementiert. Eine TLS-Unterstützung fehlt. AES, RSA, ECIES, DLIES, DSA und ECDSA werden unterstützt, es fehlt jedoch die Unterstützung von ECKDSA, ECGDSA und Merkle-Signaturen. Es werden die Brainpool- und NIST-Kurven unterstützt, jedoch nicht eigene Kurven. Von den geforderten MACs werden alle unterstützt. Weder X.509, noch PKCS#11, noch TPMs werden von Crypto++ unterstützt.

Eine statische Registrierung von Implementierungen ist nicht möglich. Crypto++ steht unter der Boost-Lizenz. Ein Ansprechpartner für Sicherheitsfragen wird konkret benannt und auch ein PGP-Key zur verschlüsselten Kommunikation wird angegeben. Das letzte Release 5.6.2 wurde am 20.2.2013 veröffentlicht. Neue Releases erscheinen unregelmäßig in Abständen von wenigen Monaten bis mehreren Jahren. Crypto++ bietet keine OpenSSL-kompatible Schnittstelle. Eine Testsuite ist vorhanden, enthält jedoch hauptsächlich Positivtests. Bindings für andere Programmiersprachen werden nicht angeboten. OS X, Linux, Windows und iOS werden als Plattformen unterstützt, für Android ist die Unterstützung unklar. Eine Doxygen API-Dokumentation ist vorhanden, sowie auch ein Handbuch mit Beispielcode in Form eines Wikis.

Kriterium	Punktwertung Crypto++
Paddingverfahren	
OAEP	1
PSS	1
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	1
XTS	0
CBC	1
CTR	1
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	0
Server	0
TLS Version	
TLS 1.0	0
TLS 1.1	0
TLS 1.2	0
DTLS 1.0	0
DTLS 1.2	0
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	0
TLS-DHE	0
TLS-PSK	0
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0,5
DLIES	0,5

Asymm. Signatur	
RSA	1
DSA	1
ECDSA	1
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	1
NIST	0,5
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	1
GMAC	1
X.509	
Kettenvalidierung	0
Generierung	0
CRL	0
OCSP	0
DANE	0
PKCS#11-Schnittstelle	0
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	0
Lizenz	Boost
Ansprechpartner für Sicherheitsfragen	2
Aktivität des/der Entwickler/Häufigkeit v. Releases	1
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	1
Negativtests	0
Bindings für andere Programmiersprachen	0
Portierbarkeit	2
Dokumentation/Handbücher	2

Summe:	27,5/56
--------	---------

Tabelle 1.5: Auswertung Crypto++

1.2.5 GnuTLS

GnuTLS ist eine TLS-Bibliothek geschrieben in C, die seit 2000 entwickelt wird. Sie unterstützt lediglich die PKCS#1 v1.5-Paddingverfahren. Als Modes of Operation werden GCM und CBC unterstützt, während XTS und CTR fehlen. Die Hashfunktionen der SHA-1 und SHA-2 Familien werden unterstützt. Eine KDF nach NIST SP800-56C ist nicht implementiert. TLS wird als Client und als Server unterstützt, und das in den Versionen TLS 1.0, 1.1 und 1.2 und DTLS 1.0 und DTLS 1.2. Von den in TR-02102-2 gestatteten Cipher Suites werden mit TLS-ECDHE, TLS-DHE und TLS-PSK alle unterstützt. AES, RSA, DSA und ECDSA werden unterstützt, es fehlt jedoch die Unterstützung von ECIES, DLIES, ECKDSA, ECGDSA und Merkle-Signaturen. Es werden nur die NIST-Kurven unterstützt, nicht jedoch die Brainpool-Kurven oder gar eigene Kurven. Von den geforderten MACs wird lediglich der HMAC unterstützt. Für X.509 wird die Kettenvalidierung und Generierung unterstützt, außerdem auch CRL und OCSP. Auch DANE wird unterstützt. GnuTLS unterstützt die Einbindung von kryptographischer Hardware über eine PKCS#11-Schnittstelle. Eine TPM-Unterstützung ist auch gegeben.

Eine statische Registrierung von Implementierungen ist nicht möglich. GnuTLS steht unter der LGPL-Lizenz. Ein Ansprechpartner für Sicherheitsfragen wird konkret benannt, ein PGP-Key zur verschlüsselten Kommunikation wird jedoch nicht angegeben. Das letzte Release 3.3.16 wurde am 12.7.2015 veröffentlicht. Neue Releases erscheinen üblicherweise in ein- bis zweimonatigen Abständen. GnuTLS bietet keine OpenSSL-kompatible Schnittstelle. Eine Testsuite ist vorhanden und enthält Positiv- wie Negativtests. Bindings für andere Programmiersprachen werden nicht angeboten. OS X, Linux und Android werden als Plattformen unterstützt, Windows jedoch nicht. Für iOS ist die Unterstützung unklar. Eine Doxygen API-Dokumentation ist vorhanden, sowie auch ein Handbuch, jedoch ohne Beispielcode.

Kriterium	Punktwertung GnuTLS
Paddingverfahren	
OAEP	0
PSS	0
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	1
XTS	0
CBC	1
CTR	0
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	1
Server	1
TLS Version	
TLS 1.0	0,5
TLS 1.1	0,5
TLS 1.2	1
DTLS 1.0	0,5
DTLS 1.2	0,5
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	1
TLS-DHE	1
TLS-PSK	1
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	1
DSA	1
ECDSA	1
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	0
NIST	0,5
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	0
GMAC	0
X.509	
Kettenvalidierung	1
Generierung	1
CRL	1
OCSP	1
DANE	1
PKCS#11-Schnittstelle	1
TPM-Unterstützung	1
Weitere Kriterien	
Statische Registrierung von Implementierungen	0
Lizenz	LGPL
Ansprechpartner für Sicherheitsfragen	2
Aktivität des/der Entwickler/Häufigkeit v. Releases	2
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	1
Negativtests	1
Bindings für andere Programmiersprachen	0
Portierbarkeit	1,5
Dokumentation/Handbücher	1

Summe:	35/56
--------	-------

Tabelle 1.6: Auswertung GnuTLS

1.2.6 Libgcrypt

Libgcrypt ist eine allgemeine Kryptobibliothek geschrieben in C, die aus dem GnuPG-Projekt entstanden ist. Sie unterstützt alle geforderten Paddingverfahren. Von den Modes of Operation fehlt lediglich XTS. Die Hashfunktionen der SHA-1 und SHA-2 Familien werden unterstützt. Eine KDF nach NIST SP800-56C ist nicht implementiert. Eine TLS-Unterstützung fehlt. AES, RSA, DSA und ECDSA werden unterstützt, es fehlt jedoch die Unterstützung von ECIES, DLIES, ECKDSA, ECGDSA und Merkle-Signaturen. Es werden die Brainpool- und NIST-Kurven unterstützt, jedoch nicht eigene Kurven. Von den geforderten MACs werden alle unterstützt. Weder X.509 noch TPMs werden von Libgcrypt unterstützt. Libgcrypt unterstützt kryptographische Hardware mittels PKCS#11.

Eine statische Registrierung von Implementierungen ist möglich, es sind jedoch nur ausgewählte Teile der Bibliothek deaktivierbar. Libgcrypt steht unter der LGPL-Lizenz. Ein Ansprechpartner für Sicherheitsfragen wird konkret benannt, ein PGP-Key zur verschlüsselten Kommunikation wird jedoch nicht angegeben. Das letzte Release 1.6.3 wurde am 27.2.2015 veröffentlicht. Neue Releases erscheinen üblicherweise in ein- bis dreimonatigen Abständen. Libgcrypt bietet keine OpenSSL-kompatible Schnittstelle. Eine Testsuite ist vorhanden, enthält jedoch hauptsächlich Positivtests. Bindings für andere Programmiersprachen werden nicht angeboten. OS X und Linux werden als Plattformen unterstützt, Windows jedoch nicht. Für iOS und Android ist die Unterstützung unklar. Eine Doxygen API-Dokumentation ist vorhanden, sowie auch ein Handbuch, jedoch ohne Beispielcode.

Kriterium	Punktwertung Libgcrypt
Paddingverfahren	
OAEP	1
PSS	1
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	1
XTS	0
CBC	1
CTR	1
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	0
Server	0
TLS Version	
TLS 1.0	0
TLS 1.1	0
TLS 1.2	0
DTLS 1.0	0
DTLS 1.2	0
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	1
TLS-DHE	1
TLS-PSK	1
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	1
DSA	1
ECDSA	1
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	1
NIST	0,5
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	1
GMAC	1
X.509	
Kettenvalidierung	0
Generierung	0
CRL	0
OCSP	0
DANE	0
PKCS#11-Schnittstelle	1
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	1
Lizenz	LGPL
Ansprechpartner für Sicherheitsfragen	1
Aktivität des/der Entwickler/Häufigkeit v. Releases	2
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	1
Negativtests	0
Bindings für andere Programmiersprachen	0
Portierbarkeit	1
Dokumentation/Handbücher	1

Summe:	26,5/56
---------------	----------------

Tabelle 1.7: Auswertung Libgcrypt

1.2.7 LibreSSL

LibreSSL ist ein Fork der sehr populären OpenSSL-Bibliothek mit dem Fokus auf sauberen, aufgeräumten Code. LibreSSL ist funktional identisch zu OpenSSL.

Eine statische Registrierung von Implementierungen ist möglich. LibreSSL steht unter verschiedenen Lizenzen: Apache, BSD, ISC und Public Domain. Ein Ansprechpartner für Sicherheitsfragen wird nicht benannt. Das letzte Release 2.2.3 wurde am 29.8.2015 veröffentlicht. Neue Releases erscheinen in ein- bis dreimonatigen Abständen. Meist finden in monatlichen Abständen neue Releases statt. LibreSSL bietet eine OpenSSL-kompatible Schnittstelle. Eine Testsuite ist vorhanden, enthält jedoch hauptsächlich Positivtests. Bindings für andere Programmiersprachen werden nicht angeboten. OS X, Linux und Windows werden als Plattformen unterstützt. Für iOS und Android ist die Unterstützung unklar. Es ist weder eine Doxygen API-Dokumentation noch ein Handbuch vorhanden. Es existieren lediglich ein paar wenige, unvollständige Manpages.

Kriterium	Punktwertung LibreSSL
Paddingverfahren	
OAEP	1
PSS	1
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	1
XTS	0,5
CBC	1
CTR	1
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	1
Server	1
TLS Version	
TLS 1.0	0,5
TLS 1.1	0,5
TLS 1.2	1
DTLS 1.0	0,5
DTLS 1.2	0,5
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	1
TLS-DHE	1
TLS-PSK	1
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	1
DSA	1
ECDSA	1
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	1
NIST	0,5
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	1
GMAC	0
X.509	
Kettenvalidierung	1
Generierung	1
CRL	1
OCSP	1
DANE	0
PKCS#11-Schnittstelle	0
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	2
Lizenz	Apache, BSD, ISC und Public Domain
Ansprechpartner für Sicherheitsfragen	0
Aktivität des/der Entwickler/Häufigkeit v. Releases	2
Kompatibilität mit der OpenSSL-Schnittstelle	1
Testsuite	
Positivtests	1
Negativtests	0
Bindings für andere Programmiersprachen	0
Portierbarkeit	1,5
Dokumentation/Handbücher	0

Summe:	37,5/56
---------------	----------------

Tabelle 1.8: Auswertung LibreSSL

1.2.8 Libsodium

Libsodium ist ein im Jahr 2014 gestarteter Fork der NaCl-Bibliothek (siehe 1.2.12). Libsodium unterstützt keines der geforderten Paddingverfahren. Von den Modes of Operation wird lediglich CTR unterstützt, von den Hashfunktionen nur die SHA-2 Familie. Eine KDF nach NIST SP800-56C ist nicht implementiert. TLS wird nicht unterstützt. AES wird unterstützt, RSA, DSA, ECDSA, ECIES, DLIES, ECKDSA, ECGDSA und Merkle-Signaturen jedoch nicht. Die Brainpool-, NIST- und eigene Kurven werden nicht unterstützt. Von den geforderten MACs wird nur HMAC unterstützt. X.509 wird nicht unterstützt, genauso wie PKCS#11 und TPMs.

Eine statische Registrierung von Implementierungen ist nicht möglich. Libsodium steht unter der ISC-Lizenz. Ein Ansprechpartner für Sicherheitsfragen wird nicht benannt. Das letzte Release 1.0.3 wurde am 9.5.2015 veröffentlicht. Die bisher 4 Releases erschienen in Abständen von zwei bis sechs Monaten. Libsodium bietet keine OpenSSL-kompatible Schnittstelle. Eine Testsuite ist vorhanden, enthält jedoch hauptsächlich Positivtests. Bindings werden für viele Sprachen angeboten, u.a. für .NET, C++, Java, Python, Perl, Ruby und viele weitere. Libsodium unterstützt alle geforderten Plattformen. Es sind eine API-Dokumentation und auch ein Handbuch mit vielen Codebeispielen vorhanden.

Kriterium	Punktwertung Libsodium
Paddingverfahren	
OAEP	0
PSS	0
PKCS#1 v1.5	0
Modes of Operation	
GCM	0
XTS	0
CBC	0
CTR	1
Hashfunktionen	
SHA-1	0
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	0
Server	0
TLS Version	
TLS 1.0	0
TLS 1.1	0
TLS 1.2	0
DTLS 1.0	0
DTLS 1.2	0
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	0
TLS-DHE	0
TLS-PSK	0
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	0
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	0
DSA	0
ECDSA	0
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	0
NIST	0
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	0
GMAC	0
X.509	
Kettenvalidierung	0
Generierung	0
CRL	0
OCSP	0
DANE	0
PKCS#11-Schnittstelle	0
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	0
Lizenz	ISC
Ansprechpartner für Sicherheitsfragen	0
Aktivität des/der Entwickler/Häufigkeit v. Releases	2
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	1
Negativtests	0
Bindings für andere Programmiersprachen	2
Portierbarkeit	2,5
Dokumentation/Handbücher	2

Summe:	15,5/56
---------------	----------------

Tabelle 1.9: Auswertung Libsodium

1.2.9 Libtomcrypt

Libtomcrypt ist eine allgemeine Kryptobibliothek geschrieben in C. Sie unterstützt alle geforderten Paddingverfahren, Modes of Operation und Hashfunktionen. Eine KDF nach NIST SP800-56C ist nicht implementiert. Eine TLS-Unterstützung fehlt. AES, RSA, DSA und ECDSA werden unterstützt, es fehlt jedoch die Unterstützung von ECIES, DLIES, ECKDSA, ECGDSA und Merkle-Signaturen. Es werden sowohl die Brainpool- und NIST-Kurven unterstützt, als auch eigene Kurven. Von den geforderten MACs fehlt lediglich GMAC. Weder X.509, noch PKCS#11, noch TPMs werden von Libtomcrypt unterstützt.

Eine statische Registrierung von Implementierungen ist möglich. Libtomcrypt steht unter einer Public Domain Lizenz. Ein Ansprechpartner für Sicherheitsfragen wird nicht konkret benannt. Das letzte Release 1.17 wurde vor 2010 veröffentlicht. Seitdem gab es keine neuen Releases mehr. Libtomcrypt bietet keine OpenSSL-kompatible Schnittstelle. Eine Testsuite ist nicht auffindbar. Bindings für andere Programmiersprachen wie Python und Haskell sind verfügbar. Linux und Windows werden als Plattformen unterstützt. Für OS X, iOS und Android ist die Unterstützung unklar. Eine API-Dokumentation ist vorhanden, sowie auch ein Handbuch, jedoch ohne Beispielcode.

Kriterium	Punktwertung LibreSSL
Paddingverfahren	
OAEP	1
PSS	1
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	1
XTS	0,5
CBC	1
CTR	1
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	0
Server	0
TLS Version	
TLS 1.0	0
TLS 1.1	0
TLS 1.2	0
DTLS 1.0	0
DTLS 1.2	0
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	0
TLS-DHE	0
TLS-PSK	0
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	1
DSA	1
ECDSA	1
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	1
NIST	0,5
Eigene Kurven (BSI-Kurven)	2
MAC	
HMAC	1
CMAC	1
GMAC	0
X.509	
Kettenvalidierung	0
Generierung	0
CRL	0
OCSP	0
DANE	0
PKCS#11-Schnittstelle	0
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	2
Lizenz	Public Domain
Ansprechpartner für Sicherheitsfragen	0
Aktivität des/der Entwickler/Häufigkeit v. Releases	1
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	0
Negativtests	0
Bindings für andere Programmiersprachen	2
Portierbarkeit	1
Dokumentation/Handbücher	1

Summe:	27/56
--------	-------

Tabelle 1.10: Auswertung Libtomcrypt

1.2.10 MatrixSSL

MatrixSSL ist eine TLS-Bibliothek geschrieben in C, die seit 2004 bei der Firma PeerSec Networks entwickelt wird. Sie unterstützt die PSS- und PKCS#1 v1.5-Paddingverfahren, jedoch nicht OAEP. Als Modes of Operation werden GCM und CBC unterstützt, während XTS und CTR fehlen. Die Hashfunktionen der SHA-1 und SHA-2 Familien werden unterstützt. Eine KDF nach NIST SP800-56C ist nicht implementiert. TLS wird als Client und als Server unterstützt, und das in den Versionen TLS 1.0, 1.1 und 1.2 und DTLS 1.0 und DTLS 1.2. Von den in TR-02102-2 gestatteten Cipher Suites werden mit TLS-ECDHE, TLS-DHE und TLS-PSK alle unterstützt. AES, RSA und ECDSA werden unterstützt, es fehlt jedoch die Unterstützung von DSA, ECIES, DLIES, ECKDSA, ECGDSA und Merkle-Signaturen. Es werden die Brainpool- und NIST-Kurven unterstützt, nicht jedoch eigene Kurven. Von den geforderten MACs werden alle unterstützt. Für X.509 werden Kettenvalidierung und CRLs unterstützt, jedoch nicht die Generierung, OCSP und DANE. PKCS#11 und TPMs werden nicht unterstützt.

Eine statische Registrierung von Implementierungen ist nicht möglich. MatrixSSL steht unter einer Dual License: GPLv2 und proprietär. Die proprietäre Lizenz erlaubt es, Änderungen am Code der Bibliothek genauso wie eigenen Code der die Bibliothek benutzt nicht veröffentlichen zu müssen. Der Einsatz unter dieser Lizenz ist allerdings kostenpflichtig. Ein Ansprechpartner für Sicherheitsfragen wird konkret benannt, ein PGP-Key zur verschlüsselten Kommunikation wird jedoch nicht angegeben. Das letzte Release 3.7.2b wurde am 13.7.2015 veröffentlicht. Neue Releases erscheinen üblicherweise in drei- bis fünfmonatigen Abständen. MatrixSSL bietet keine OpenSSL-kompatible Schnittstelle. Eine Testsuite ist vorhanden, enthält jedoch überwiegend nur Positivtests. Bindings für andere Programmiersprachen werden nicht angeboten. MatrixSSL unterstützt alle geforderten Plattformen. Eine API-Dokumentation ist vorhanden, sowie auch ein Handbuch, jedoch ohne Beispielcode.

Kriterium	Punktwertung MatrixSSL
Paddingverfahren	
OAEP	0
PSS	1
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	1
XTS	0
CBC	1
CTR	0
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	1
Server	1
TLS Version	
TLS 1.0	0,5
TLS 1.1	0,5
TLS 1.2	1
DTLS 1.0	0,5
DTLS 1.2	0,5
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	1
TLS-DHE	1
TLS-PSK	1
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	1
DSA	0
ECDSA	1
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	1
NIST	0,5
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	1
GMAC	1
X.509	
Kettenvalidierung	1
Generierung	0
CRL	1
OCSP	0
DANE	0
PKCS#11-Schnittstelle	0
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	0
Lizenz	GPLv2 und proprietär
Ansprechpartner für Sicherheitsfragen	1
Aktivität des/der Entwickler/Häufigkeit v. Releases	2
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	1
Negativtests	0
Bindings für andere Programmiersprachen	0
Portierbarkeit	2,5
Dokumentation/Handbücher	1

Summe:	33/56
--------	-------

Tabelle 1.11: Auswertung MatrixSSL

1.2.11 mbed TLS

mbed TLS (vormals PolarSSL) ist eine TLS-Bibliothek geschrieben in C, die seit 2006 unter dem Namen XySSL entwickelt, 2008 zu PolarSSL geforkt und 2014 nach der Übernahme durch die Firma ARM Holdings in mbed TLS umbenannt wurde. Die Bibliothek ist vor allem für den Einsatz auf eingebetteten Systemen vorgesehen. Sie unterstützt alle geforderten Paddingverfahren. Als Modes of Operation werden GCM, CBC und CTR unterstützt, während XTS fehlt. Die Hashfunktionen der SHA-1 und SHA-2 Familien werden unterstützt. Eine KDF nach NIST SP800-56C ist nicht implementiert. TLS wird als Client und als Server unterstützt, und das in den Versionen TLS 1.0, 1.1 und 1.2 und DTLS 1.0 und DTLS 1.2. Von den in TR-02102-2 gestatteten Cipher Suites werden mit TLS-ECDHE, TLS-DHE und TLS-PSK alle unterstützt. AES, RSA und ECDSA werden unterstützt, es fehlt jedoch die Unterstützung von DSA, ECIES, DLIES, ECKDSA, ECGDSA und Merkle-Signaturen. Es werden die Brainpool- und NIST-Kurven unterstützt, nicht jedoch eigene Kurven. Von den geforderten MACs wird nur HMAC unterstützt. Für X.509 werden Kettenvalidierung, Generierung und CRLs unterstützt, jedoch nicht OCSP und DANE. Kryptographische Hardware kann über eine PKCS#11-Schnittstelle eingebunden werden. TPMs werden nicht unterstützt.

Eine statische Registrierung von Implementierungen ist nicht möglich. mbed TLS steht unter der Apache-Lizenz. Ein Ansprechpartner für Sicherheitsfragen wird konkret benannt, ein PGP-Key zur verschlüsselten Kommunikation wird auch angegeben. Das letzte Release 2.0.0 wurde am 13.7.2015 veröffentlicht. Neue Releases erscheinen üblicherweise in ein- bis zweimonatigen Abständen. mbed TLS bietet keine OpenSSL-kompatible Schnittstelle. Eine Testsuite ist vorhanden und enthält Positivtests- wie Negativtests. Bindings für andere Programmiersprachen werden nicht angeboten. mbed TLS unterstützt alle geforderten Plattformen. Eine API-Dokumentation ist vorhanden, sowie auch ein Handbuch, jedoch ohne Beispielcode.

Kriterium	Punktwertung mbed TLS
Paddingverfahren	
OAEP	1
PSS	1
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	1
XTS	0
CBC	1
CTR	1
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	1
Server	1
TLS Version	
TLS 1.0	0,5
TLS 1.1	0,5
TLS 1.2	1
DTLS 1.0	0,5
DTLS 1.2	0,5
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	1
TLS-DHE	1
TLS-PSK	1
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	1
DSA	0
ECDSA	1
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	1
NIST	0,5
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	0
GMAC	0
X.509	
Kettenvalidierung	1
Generierung	1
CRL	1
OCSP	0
DANE	0
PKCS#11-Schnittstelle	1
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	0
Lizenz	Apache, GPLv2 und proprietär
Ansprechpartner für Sicherheitsfragen	2
Aktivität des/der Entwickler/Häufigkeit v. Releases	2
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	1
Negativtests	1
Bindings für andere Programmiersprachen	0
Portierbarkeit	2,5
Dokumentation/Handbücher	1

Summe:	37/56
---------------	--------------

Tabelle 1.12: Auswertung mbed TLS

1.2.12 NaCl

NaCl ist eine Kryptobibliothek von Daniel J. Bernstein mit dem Fokus auf Performance und auf eine sichere Implementierung von Algorithmen und Protokollen. NaCl bietet zu jedem Themenkomplex jeweils nur einen Algorithmus an, um dem Anwender nicht die Wahl überlassen zu müssen. Die angebotenen Algorithmen sind größtenteils von Daniel J. Bernstein entwickelte Algorithmen wie Curve25519, Salsa20 und Poly1305. NaCl unterstützt keines der geforderten Paddingverfahren oder Modes of Operation. Als Hashfunktion wird nur die SHA-2 Familie unterstützt. Eine KDF nach NIST SP800-56C ist nicht implementiert. TLS wird nicht unterstützt. AES wird unterstützt, RSA, DSA, ECDSA, ECIES, DLIES, ECKDSA, ECGDSA und Merkle-Signaturen jedoch nicht. Die Brainpool-, NIST- und eigene Kurven werden nicht unterstützt. Von den geforderten MACs wird nur HMAC nicht unterstützt. X.509 wird nicht unterstützt, genauso wie PKCS#11 und TPMs.

Eine statische Registrierung von Implementierungen ist nicht möglich. NaCl steht unter Public Domain. Ein Ansprechpartner für Sicherheitsfragen wird nicht benannt. Das letzte Release 20110221 stammt aus dem Jahr 2011. NaCl bietet keine OpenSSL-kompatible Schnittstelle. Eine Testsuite ist vorhanden, enthält jedoch hauptsächlich Positivtests. Bindings werden u.a. für Python, PHP und Ruby angeboten. NaCl unterstützt Linux als Plattform, Unterstützung für OS X, Windows, iOS und Android sind unklar. Es sind eine API-Dokumentation und auch ein Handbuch mit vielen Codebeispielen vorhanden.

Kriterium	Punktwertung NaCl
Paddingverfahren	
OAEP	0
PSS	0
PKCS#1 v1.5	0
Modes of Operation	
GCM	0
XTS	0
CBC	0
CTR	0
Hashfunktionen	
SHA-1	0
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	0
Server	0
TLS Version	
TLS 1.0	0
TLS 1.1	0
TLS 1.2	0
DTLS 1.0	0
DTLS 1.2	0
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	0
TLS-DHE	0
TLS-PSK	0
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	0
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	0
DSA	0
ECDSA	0
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	0
NIST	0
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	0
GMAC	0
X.509	
Kettenvalidierung	0
Generierung	0
CRL	0
OCSP	0
DANE	0
PKCS#11-Schnittstelle	0
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	0
Lizenz	Public Domain
Ansprechpartner für Sicherheitsfragen	0
Aktivität des/der Entwickler/Häufigkeit v. Releases	0
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	1
Negativtests	0
Bindings für andere Programmiersprachen	2
Portierbarkeit	0,5
Dokumentation/Handbücher	2

Summe:	10,5/56
---------------	----------------

Tabelle 1.13: Auswertung NaCl

1.2.13 Nettle

Nettle ist eine allgemeine Kryptobibliothek geschrieben in C, deren Entwicklung 2001 als eine Sammlung von kryptographischen Funktionen im Rahmen der Entwicklung der lsh Secure Shell begann. Sie unterstützt nur die PKCS#1 v1.5-Paddingverfahren, jedoch nicht OAEP oder PSS. Unterstützung für den XTS-Modus fehlt. SHA-1 und SHA-2 werden unterstützt. Eine KDF nach NIST SP800-56C ist nicht implementiert. Eine TLS-Unterstützung fehlt. AES, RSA, DSA und ECDSA werden unterstützt, es fehlt jedoch die Unterstützung von ECIES, DLIES, ECKDSA, ECGDSA und Merkle-Signaturen. Es werden nur die NIST-Kurven unterstützt. Von den geforderten MACs wird nur HMAC angeboten. Weder X.509, noch PKCS#11, noch TPMs werden von Nettle unterstützt.

Eine statische Registrierung von Implementierungen ist möglich, jedoch sind nur Teile der Algorithmen deaktivierbar. Nettle steht unter der LPGP-Lizenz. Ein Ansprechpartner für Sicherheitsfragen wird konkret benannt. Auch ein PGP-Key ist angegeben. Das letzte Release 3.1 wurde am 7.4.2015 veröffentlicht. Neue Releases erscheinen ein- bis zweimal pro Jahr. Nettle bietet keine OpenSSL-kompatible Schnittstelle. Eine Testsuite ist vorhanden, enthält jedoch vorwiegend Positivtests. Bindings für andere Programmiersprachen wie Perl, Haskell oder Pike sind verfügbar. NaCl unterstützt Linux als Plattform, Unterstützung für OS X, Windows, iOS und Android sind unklar. Eine API-Dokumentation sowie auch ein Handbuch mit Beispielcode sind verfügbar.

Kriterium	Punktwertung Nettle
Paddingverfahren	
OAEP	0
PSS	0
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	1
XTS	0
CBC	1
CTR	1
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	0
Server	0
TLS Version	
TLS 1.0	0
TLS 1.1	0
TLS 1.2	0
DTLS 1.0	0
DTLS 1.2	0
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	0
TLS-DHE	0
TLS-PSK	0
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	1
DSA	1
ECDSA	1
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	0
NIST	0,5
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	0
GMAC	0
X.509	
Kettenvalidierung	0
Generierung	0
CRL	0
OCSP	0
DANE	0
PKCS#11-Schnittstelle	0
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	1
Lizenz	LGPL
Ansprechpartner für Sicherheitsfragen	2
Aktivität des/der Entwickler/Häufigkeit v. Releases	2
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	1
Negativtests	0
Bindings für andere Programmiersprachen	2
Portierbarkeit	0,5
Dokumentation/Handbücher	2

Summe:	24/56
--------	-------

Tabelle 1.14: Auswertung Nettle

1.2.14 NSS

NSS ist eine Kryptobibliothek mit Unterstützung für die beiden Protokolle TLS und S/MIME und vielen weiteren kryptographischen Algorithmen. Sie unterstützt alle geforderten Paddingverfahren. Als Modes of Operation werden GCM, CBC und CTR unterstützt, während XTS fehlt. Die Hashfunktionen der SHA-1 und SHA-2 Familien werden unterstützt. Eine KDF nach NIST SP800-56C ist nicht implementiert. TLS wird als Client und als Server unterstützt, und das in den Versionen TLS 1.0, 1.1 und 1.2 und DTLS 1.0 und DTLS 1.2. Von den in TR-02102-2 gestatteten Cipher Suites wird nur TLS-ECDHE unterstützt. AES, RSA, DSA und ECDSA werden unterstützt, es fehlt jedoch die Unterstützung von ECIES, DLIES, ECKDSA, ECGDSA und Merkle-Signaturen. Es werden ausschließlich die NIST-Kurven unterstützt. Von den geforderten MACs wird nur HMAC unterstützt. Für X.509 werden Kettenvalidierung, Generierung, CRL und OCSP unterstützt, DANE jedoch nicht. Kryptographische Hardware kann über eine PKCS#11-Schnittstelle eingebunden werden. TPMs werden nicht unterstützt.

Eine statische Registrierung von Implementierungen ist möglich, jedoch können nur bestimmte Teile deaktiviert werden. NSS steht unter der MPL-Lizenz. Ein Ansprechpartner für Sicherheitsfragen wird konkret benannt, ein PGP-Key zur verschlüsselten Kommunikation wird auch angegeben. Das letzte Release 3.19.2 wurde am 19.8.2015 veröffentlicht. Neue Releases erscheinen üblicherweise in ein- bis zweimonatigen Abständen. NSS bietet keine OpenSSL-kompatible Schnittstelle. Eine Testsuite ist vorhanden und enthält Positivtests- wie Negativtests. Bindings für andere Programmiersprachen wie Python, Perl und Rust sind verfügbar. NaCl unterstützt OS X, Linux, Windows und Android als Plattform. Die Unterstützung für iOS ist unklar. Eine API-Dokumentation ist vorhanden, sowie auch ein Handbuch, jedoch ohne Beispielcode.

Kriterium	Punktwertung NSS
Paddingverfahren	
OAEP	1
PSS	1
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	1
XTS	0
CBC	1
CTR	1
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	1
Server	1
TLS Version	
TLS 1.0	0,5
TLS 1.1	0,5
TLS 1.2	1
DTLS 1.0	0,5
DTLS 1.2	0,5
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	1
TLS-DHE	0
TLS-PSK	0
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	1
DSA	1
ECDSA	1
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	0
NIST	0,5
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	0
GMAC	0
X.509	
Kettenvalidierung	1
Generierung	1
CRL	1
OCSP	1
DANE	0
PKCS#11-Schnittstelle	1
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	1
Lizenz	MPL 2.0
Ansprechpartner für Sicherheitsfragen	2
Aktivität des/der Entwickler/Häufigkeit v. Releases	2
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	1
Negativtests	1
Bindings für andere Programmiersprachen	2
Portierbarkeit	2
Dokumentation/Handbücher	1

Summe:	38,5/56
---------------	----------------

Tabelle 1.15: Auswertung NSS

1.2.15 OpenSSL

OpenSSL ist die wohl populärste TLS- und Kryptobibliothek. OpenSSL unterstützt alle geforderten Paddingverfahren, Modes of Operation und Hashfunktionen. Eine KDF nach NIST SP800-56C ist nicht implementiert. TLS wird als Client und als Server unterstützt, und das in den Versionen TLS 1.0, 1.1 und 1.2 und DTLS 1.0 und DTLS 1.2. Von den in TR-02102-2 gestatteten Cipher Suites werden mit TLS-ECDHE, TLS-DHE und TLS-PSK alle unterstützt. AES, RSA, DSA und ECDSA werden unterstützt, es fehlt jedoch die Unterstützung von ECIES, DLIES, ECKDSA, ECGDSA und Merkle-Signaturen. Es werden die Brainpool- und NIST-Kurven unterstützt, jedoch nicht eigene Kurven. Von den geforderten MACs wird nur GMAC nicht unterstützt. Für X.509 wird die Kettenvalidierung und Generierung unterstützt, außerdem auch CRL und OCSP. DANE wird nicht unterstützt. Auch PKCS#11 und TPM-Unterstützung sind nicht vorhanden.

Eine statische Registrierung von Implementierungen ist möglich. OpenSSL steht unter der Apache- und BSD-Lizenz. Ein Ansprechpartner für Sicherheitsfragen wird konkret benannt, ein PGP-Key zur verschlüsselten Kommunikation wird auch angegeben. Das letzte Release 1.0.2d wurde am 9.7.2015 veröffentlicht. Meist finden in monatlichen Abständen neue Releases statt. Eine Testsuite ist vorhanden, enthält jedoch hauptsächlich Positivtests. Bindings für andere Programmiersprachen werden nicht angeboten. OpenSSL unterstützt alle geforderten Plattformen. Die API-Dokumentation ist sehr lückenhaft und auch Beispielcode ist nur sehr wenig vorhanden. Ein Handbuch ist nicht vorzufinden.

Kriterium	Punktwertung OpenSSL
Paddingverfahren	
OAEP	1
PSS	1
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	1
XTS	0,5
CBC	1
CTR	1
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	1
Server	1
TLS Version	
TLS 1.0	0,5
TLS 1.1	0,5
TLS 1.2	1
DTLS 1.0	0,5
DTLS 1.2	0,5
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	1
TLS-DHE	1
TLS-PSK	1
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	1
DSA	1
ECDSA	1
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	1
NIST	0,5
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	1
GMAC	0
X.509	
Kettenvalidierung	1
Generierung	1
CRL	1
OCSP	1
DANE	0
PKCS#11-Schnittstelle	0
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	2
Lizenz	Apache, BSD
Ansprechpartner für Sicherheitsfragen	2
Aktivität des/der Entwickler/Häufigkeit v. Releases	2
Kompatibilität mit der OpenSSL-Schnittstelle	1
Testsuite	
Positivtests	1
Negativtests	0
Bindings für andere Programmiersprachen	2
Portierbarkeit	2
Dokumentation/Handbücher	0

Summe:	42/56
--------	-------

Tabelle 1.16: Auswertung OpenSSL

1.2.16 Poco

Die Poco Libraries bestehen aus einer Reihe von C++ Bibliotheken für verschiedene Anwendungsgebiete, beispielsweise für Netzwerkkommunikation, Kompression, XML, Datenbanken und Kryptographie. Die für Kryptographie zuständige Poco Crypto-Bibliothek implementiert selbst keine kryptographischen Algorithmen und Protokolle, sondern verwendet intern die OpenSSL-Bibliothek. Die Poco Crypto-Bibliothek bietet jedoch nicht alle Algorithmen und Protokolle der OpenSSL-Bibliothek über die C++-Schnittstellen an.

Die Poco Crypto-Bibliothek unterstützt nur die OAEP und PKCS#1 v1.5-Paddingverfahren, nicht jedoch PSS. Nur CBC wird als Mode of Operation angeboten, während SHA-1 und SHA-2 komplett unterstützt werden. Eine KDF nach NIST SP800-56C ist nicht implementiert. TLS wird als Client und als Server unterstützt, und das in den Versionen TLS 1.0, 1.1 und 1.2. DTLS 1.0 und DTLS 1.2 sind nicht unterstützt. Von den in TR-02102-2 gestatteten Cipher Suites werden mit TLS-ECDHE, TLS-DHE und TLS-PSK alle unterstützt. AES und RSA werden unterstützt, es fehlt jedoch die Unterstützung von DSA, ECDSA, ECIES, DLIES, ECKDSA, ECGDSA und Merkle-Signaturen. Elliptische Kurven werden nicht unterstützt. Von den geforderten MACs wird nur HMAC unterstützt. Für X.509 wird nur die Kettenvalidierung und Generierung unterstützt, CRL, OCSP und DANE jedoch nicht. Auch PKCS#11 und TPM-Unterstützung sind nicht vorhanden.

Eine statische Registrierung von Implementierungen ist möglich, jedoch sind nur einige Teile deaktivierbar. Poco steht unter der Boost-Lizenz. Ein Ansprechpartner für Sicherheitsfragen wird konkret benannt, ein PGP-Key zur verschlüsselten Kommunikation jedoch nicht angegeben. Das letzte Release 1.6.1 wurde am 3.8.2015 veröffentlicht. Neue Releases finden in unregelmäßigen Abständen zwischen 2 und 6 Monaten statt. Eine Testsuite ist vorhanden, enthält jedoch hauptsächlich Positivtests. Bindings für andere Programmiersprachen werden nicht angeboten. Die Poco Libraries unterstützen OS X, Linux und Windows. Ob iOS oder Android unterstützt werden ist unklar. Eine Doxygen API-Dokumentation ist vorhanden, ein Handbuch mit Codebeispielen ebenfalls.

Kriterium	Punktwertung Poco
Paddingverfahren	
OAEP	1
PSS	0
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	0
XTS	0
CBC	1
CTR	0
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	1
Server	1
TLS Version	
TLS 1.0	0,5
TLS 1.1	0,5
TLS 1.2	1
DTLS 1.0	0
DTLS 1.2	0
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	1
TLS-DHE	1
TLS-PSK	1
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	1
DSA	0
ECDSA	0
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	0
NIST	0
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	0
GMAC	0
X.509	
Kettenvalidierung	1
Generierung	1
CRL	0
OCSP	0
DANE	0
PKCS#11-Schnittstelle	0
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	1
Lizenz	Boost
Ansprechpartner für Sicherheitsfragen	1
Aktivität des/der Entwickler/Häufigkeit v. Releases	2
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	1
Negativtests	0
Bindings für andere Programmiersprachen	0
Portierbarkeit	1,5
Dokumentation/Handbücher	2

Summe:	27,5/56
--------	---------

Tabelle 1.17: Auswertung Poco

1.2.17 s2n

Die s2n-Bibliothek ist eine kürzlich von Amazon vorgestellte TLS-Bibliothek, geschrieben in C und mit dem Fokus auf TLS-Server. s2n implementiert einen TLS-Stack, für die darunterliegenden kryptographischen Algorithmen kommt jedoch wahlweise OpenSSL, LibreSSL, BoringSSL oder Apple's CommonCrypto-Bibliothek zum Einsatz.

s2n bietet keines der geforderten Paddingverfahren auf der öffentlichen Schnittstelle an. Nur CBC wird als Mode of Operation angeboten, während SHA-1 und SHA-2 auch nicht auf der öffentlichen Schnittstelle angeboten werden. Eine KDF nach NIST SP800-56C ist nicht implementiert. TLS wird als Server unterstützt, und das in den Versionen TLS 1.0, 1.1 und 1.2. DTLS 1.0 und DTLS 1.2 sind nicht unterstützt. Von den in TR-02102-2 gestatteten Cipher Suites werden TLS-ECDHE und TLS-DHE unterstützt, TLS-PSK jedoch nicht. Unterstützung für TLS-Clients ist zwar implementiert, jedoch ist das Feature derzeit deaktiviert, da keine Kettenvalidierung durchgeführt wird. AES und RSA werden unterstützt, es fehlt jedoch die Unterstützung von DSA, ECDSA ECIES, DLIES, ECKDSA, ECGDSA und Merkle-Signaturen. Nur die elliptischen Kurven der NIST werden unterstützt. Von den geforderten MACs keiner unterstützt. Für X.509 wird nur OCSP unterstützt, Kettenvalidierung, Generierung, CRL und DANE jedoch nicht. Auch PKCS#11 und TPM-Unterstützung fehlen.

Eine statische Registrierung von Implementierungen ist nicht möglich. s2n steht unter der Apache-Lizenz. Ein Ansprechpartner für Sicherheitsfragen wird konkret benannt, ein PGP-Key zur verschlüsselten Kommunikation jedoch nicht angegeben. Es gibt noch keine offiziellen Releases, da die Bibliothek erst am 30.06.2015 der Öffentlichkeit vorgestellt wurde. Eine Testsuite ist vorhanden und enthält Positiv- wie Negativtests. s2n bietet keine OpenSSL-kompatible Schnittstelle. Bindings werden für Lua angeboten. s2n unterstützt derzeit nur Linux. Eine API-Dokumentation ist vorhanden, ein Handbuch mit Codebeispielen ebenfalls.

Kriterium	Punktwertung s2n
Paddingverfahren	
OAEP	0
PSS	0
PKCS#1 v1.5	0
Modes of Operation	
GCM	0
XTS	0
CBC	1
CTR	0
Hashfunktionen	
SHA-1	0
SHA-256	0
SHA-384	0
SHA-512	0
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	0
Server	1
TLS Version	
TLS 1.0	0,5
TLS 1.1	0,5
TLS 1.2	1
DTLS 1.0	0
DTLS 1.2	0
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	1
TLS-DHE	1
TLS-PSK	0
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	0
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	0
DSA	0
ECDSA	0
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	0
NIST	0,5
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	0
CMAC	0
GMAC	0
X.509	
Kettenvalidierung	0
Generierung	0
CRL	0
OCSP	1
DANE	0
PKCS#11-Schnittstelle	0
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	0
Lizenz	Apache 2.0
Ansprechpartner für Sicherheitsfragen	2
Aktivität des/der Entwickler/Häufigkeit v. Releases	1
Kompatibilität mit der OpenSSL-Schnittstelle	0
Testsuite	
Positivtests	1
Negativtests	1
Bindings für andere Programmiersprachen	2
Portierbarkeit	0,5
Dokumentation/Handbücher	2

Tabelle 1.18: Auswertung s2n

1.2.18 WolfSSL

WolfSSL, formerly CyaSSL oder yet another SSL ist eine TLS-Bibliothek geschrieben in C, die seit 2006 mit dem Fokus auf eingebettete Systeme entwickelt wird. Sie unterstützt die PKCS#1 v1.5-Paddingverfahren, jedoch nicht OAEP. Als Modes of Operation werden GCM, CBC und CTR unterstützt, während XTS fehlt. Die Hashfunktionen der SHA-1 und SHA-2 Familien werden unterstützt. Eine KDF nach NIST SP800-56C ist nicht implementiert. TLS wird als Client und als Server unterstützt, und das in den Versionen TLS 1.0, 1.1 und 1.2 und DTLS 1.0 und DTLS 1.2. Von den in TR-02102-2 gestatteten Cipher Suites werden mit TLS-ECDHE, TLS-DHE und TLS-PSK alle unterstützt. AES, RSA, DSA und ECDSA werden unterstützt, es fehlt jedoch die Unterstützung von ECIES, DLIES, ECKDSA, ECGDSA und Merkle-Signaturen. Es werden nur die NIST-Kurven unterstützt. Von den geforderten MACs fehlt nur CMAC. Für X.509 werden Kettenvalidierung und Generierung unterstützt, genauso wie CRL und OCSP. DANE, PKCS#11 und TPMs werden nicht unterstützt.

Eine statische Registrierung von Implementierungen ist nicht möglich. WolfSSL steht unter einer Dual License: GPLv2 und proprietär. Die proprietäre Lizenz erlaubt es, Änderungen am Code der Bibliothek genauso wie eigenen Code der die Bibliothek benutzt nicht veröffentlichen zu müssen. Der Einsatz unter dieser Lizenz ist allerdings kostenpflichtig. Ein Ansprechpartner für Sicherheitsfragen wird konkret benannt, ein PGP-Key zur verschlüsselten Kommunikation wird jedoch nicht angegeben. Das letzte Release 3.6.0 wurde am 13.7.2015 veröffentlicht. Neue Releases erscheinen üblicherweise in zwei- bis dreimonatigen Abständen. WolfSSL stellt einen Kompatibilitätsheader¹ für OpenSSL-Anwender zur Verfügung. Dieser enthält Funktionen für den Aufbau einer TLS-Verbindung und für das Laden von Private Keys und Zertifikaten aus einer Datei. Damit soll OpenSSL-Anwendern der Umstieg auf WolfSSL erleichtert werden. Eine Testsuite ist vorhanden, enthält jedoch überwiegend nur Positivtests. Bindings für andere Programmiersprachen werden nicht angeboten. WolfSSL unterstützt alle geforderten Plattformen. Eine API-Dokumentation ist vorhanden, sowie auch ein Handbuch mit Beispielcode.

¹ <https://wolfssl.com/wolfSSL/Docs-wolfssl-manual-13-openssl-compatibility.html>

Kriterium	Punktwertung WolfSSL
Paddingverfahren	
OAEP	0
PSS	0
PKCS#1 v1.5	0,5
Modes of Operation	
GCM	1
XTS	0
CBC	1
CTR	1
Hashfunktionen	
SHA-1	0,5
SHA-256	1
SHA-384	1
SHA-512	1
Key Derivation	
KDF nach NIST SP800-56C	0
TLS Peer	
Client	1
Server	1
TLS Version	
TLS 1.0	0,5
TLS 1.1	0,5
TLS 1.2	1
DTLS 1.0	0,5
DTLS 1.2	0,5
TLS Cipher Suites nach TR-02102-2	
TLS-ECDHE	1
TLS-DHE	1
TLS-PSK	1
Symm. Verfahren	
AES	1
Asymm. Verschlüsselung	
RSA	1
ECIES	0
DLIES	0

Asymm. Signatur	
RSA	1
DSA	1
ECDSA	1
ECKDSA	0
ECGDSA	0
Merkle-Signaturen	0
Elliptische Kurven	
Brainpool	0
NIST	0,5
Eigene Kurven (BSI-Kurven)	0
MAC	
HMAC	1
CMAC	0
GMAC	1
X.509	
Kettenvalidierung	1
Generierung	1
CRL	1
OCSP	1
DANE	0
PKCS#11-Schnittstelle	0
TPM-Unterstützung	0
Weitere Kriterien	
Statische Registrierung von Implementierungen	2
Lizenz	GPL und proprietär
Ansprechpartner für Sicherheitsfragen	1
Aktivität des/der Entwickler/Häufigkeit v. Releases	2
Kompatibilität mit der OpenSSL-Schnittstelle	1
Testsuite	
Positivtests	1
Negativtests	0
Bindings für andere Programmiersprachen	0
Portierbarkeit	2,5
Dokumentation/Handbücher	2

Summe:	38/56
---------------	--------------

Tabelle 1.19: Auswertung WolfSSL

1.3 Ergebnis

Die folgende Tabelle zeigt das Ergebnis aller 18 analysierten Bibliotheken in der Übersicht.

Platzierung	Bibliothek	Punktzahl
1.	Botan	45
2.	OpenSSL	42,5
3.	NSS	38,5
4.	Cryptlib	38
4.	WolfSSL	38
6.	LibreSSL	37,5
7.	mbed TLS	37
8.	GnuTLS	35
9.	MatrixSSL	33
10.	Crypto++	27,5
10.	Poco	27,5
12.	Libtomcrypt	27
13.	Libgcrypt	26,5
14.	Nettle	24
15.	Beecrypt	17,5
16.	s2n	17
17.	Libsodium	15,5
18.	NaCl	10,5

Tabelle 1.20: Ergebnisse der analysierten Bibliotheken

Crypto++, Libtomcrypt, Libgcrypt, Nettle, Beecrypt, Libsodium und NaCl implementieren TLS nicht. Eine vollständige Neuimplementierung eines TLS-Stacks ist im Rahmen des Projektes nicht vorgesehen. Daher scheiden diese Bibliotheken als Kandidaten für eine detailliertere Analyse aus. s2n ist als TLS-Bibliothek noch sehr neu und daher eher als experimentell zu betrachten. Außerdem bietet es für den Anwender keine Schnittstellen für kryptographische Primitive, sondern lediglich solche für TLS an. Daher scheidet auch s2n als Kandidat aus. Poco erscheint lediglich als Wrapper um OpenSSL und wegen der im Vergleich mit anderen Kandidaten geringeren Punktzahl auch nicht für eine Weiterentwicklung geeignet.

Die drei Kandidaten mit der höchsten Punktzahl sind Botan, OpenSSL und NSS. Botan deckt als allgemeine Kryptobibliothek mit zusätzlichem TLS-Stack die Anforderungen zur Weiterentwicklung im Rahmen des Projektes am besten ab. Die Simplified BSD Lizenz unter der Botan veröffentlicht wird ist für die Weiterentwicklung im Rahmen des Projektes und den Einsatz durch VS-Hersteller gänzlich geeignet, da sie die Modifikation der Bibliothek gestattet ohne den Zwang, Änderungen offenzulegen und unter gleicher Lizenz zu veröffentlichen (wie beispielsweise im Falle der

GPL). Daher soll Botan im nächsten Schritt detailliert analysiert werden. Eine detaillierte Analyse der zweitplatzierten OpenSSL-Bibliothek, die ja bereits im Projekt 154 "Quellcode-basierte Untersuchung von kryptographisch relevanten Aspekten der OpenSSL-Bibliothek (OpenSSL)" [OpenSLA] durchgeführt wurde, erscheint nicht sinnvoll. Auf Wunsch des BSI soll dafür LibreSSL als API-kompatible Alternative zu OpenSSL detailliert analysiert werden. Die einzelnen Teile von LibreSSL sind unter unterschiedlichen Lizenzen veröffentlicht (Apache, BSD, ISC und Public Domain). All diese Lizenzen sind für die Weiterentwicklung im Rahmen des Projektes und den Einsatz durch VS-Hersteller gänzlich geeignet, da sie die Modifikation der Bibliothek gestatten ohne den Zwang, Änderungen offenzulegen und unter gleicher Lizenz zu veröffentlichen. Als drittplatzierte Bibliothek soll schließlich auch die bei Mozilla entwickelte NSS-Bibliothek detailliert auf ihre Eignung hin analysiert werden. Die MPL 2.0 Lizenz, unter der NSS veröffentlicht wird, erlaubt den Einsatz einer proprietären Lizenz für neu entwickelte Teile, alle unter der MPL 2.0 stehenden vorhandenen Quelldateien müssen jedoch auch nach einer Modifikation unter den Bedingungen der MPL 2.0 frei zugänglich gemacht wird. Damit ist NSS jedoch immer noch für die Weiterentwicklung im Rahmen des Projektes und den Einsatz durch VS-Hersteller geeignet, da etwaige Änderungen an der Bibliothek im Rahmen des Projektes an das Originalprojekt eingereicht werden sollen und eine Änderung der Bibliothek durch einen VS-Hersteller im Normalfall nicht nötig sein sollte.

2 Detaillierte Analyse der vorausgewählten Bibliotheken

In diesem Kapitel werden die ausgewählten Bibliotheken **Botan 1.11.18**, **LibreSSL 2.2.2** und **NSS 3.19.2** einer detaillierten Analyse unterzogen. Das Ziel besteht darin *die* Bibliothek zu identifizieren, mit der die gesteckten Ziele mit möglichst wenig Aufwand bei gleichzeitig hoher Qualität erreicht werden können.

Untersucht werden die Implementierung von Seitenkanalanalyse-Gegenmaßnahmen, die Plattformerunterstützung, Design und Komplexität, die Unterstützung kryptographischer Hardware, Performance, enthaltene Zufallszahlengeneratoren, die Unterstützung der PKCS-Standards 1, 10, 11 und 12, die Testabdeckung, Grad und Qualität der Dokumentation, die Mechanismen zum sicheren Löschen von Geheimnissen, fehlende und zu entfernende Algorithmen, Fehler in der Zertifikatsvalidierung, Fehler in der TLS-Protokollvalidierung, aufgedeckte Fehler in der statischen Analyse, der Einsatz von Continuous Integration und die Kompatibilität mit der OpenSSL-Schnittstelle.

2.1 Methodik

Für jedes Kriterium wird zunächst der Istzustand festgestellt und dann der Aufwand zum Erreichen eines Sollzustandes in Personentagen (PT) angegeben. Beispielsweise wird für das Kriterium Implementierung von SCA-Gegenmaßnahmen untersucht, ob die Implementierung der jeweiligen Bibliothek bestimmte Maßnahmen gegen Seitenkanalanalysen trifft. Ist dies der Fall, so ist nichts zu tun (0 Personentage). Sind die Maßnahmen nicht oder nur unvollständig implementiert, so besteht der Aufwand darin, die Maßnahmen ganz oder teilweise zu ergänzen. Die angegebenen Schätzungen sind jedoch nur als Vergleichswerte zwischen den Bibliotheken zu verstehen und beziehen sich ausdrücklich nicht auf die vom Auftragnehmer im Angebot genannten Aufwände.

2.2 Analyse

2.2.1 Implementierung von SCA-Gegenmaßnahmen

In diesem Abschnitt wird die Resistenz einiger Implementierungen gegen verschiedene Seitenkanalangriffe untersucht.

2.2.1.1 RSA-Blinding

Brumley und Boneh haben im Jahr 2003 neue Timing-basierte Seitenkanalangriffe gegen die RSA-Verschlüsselung vorgestellt [BB2003]. Das Ziel der Angriffe war die Berechnung des privaten RSA-Schlüssels anhand der Dauer von RSA-Exponentiationen mit privatem Schlüssel (Entschlüsselung oder Signaturerstellung). Als Gegenmaßnahme wurde RSA-Blinding für private RSA-Operationen vorgestellt.

Botan

Die relevanten Funktionen befinden sich in folgenden Dateien:

- `lib/pubkey/blinding.c`: `Blinder-Konstruktor`, `Blinder.blind`,

Blinder.unblind: Der Konstruktor erzeugt neue Elemente $\mathbf{A}$ und $\mathbf{A}_{inv}$, wobei $A \cdot A_{inv} \equiv 1 \mod N$. Funktionen `blind()` und `unblind()` kümmern sich um das Blinding. In dem Aufruf der Funktion `blind()` wird die Aktualisierung von $\mathbf{A}$ und $\mathbf{A}_{inv}$ durchgeführt: $A' \equiv A \cdot A \mod N$ und $A'_{inv} \equiv A_{inv} \cdot A_{inv} \mod N$.

- **lib/pubkey/rsa/rsa.cpp: RSA_Private_Operation** (Z. 74): In dem Konstruktor dieser Klasse wird der Blinder einmalig initialisiert. Der Blinder wird bei der Ausführung der privaten Operation eingesetzt.

Bei dem Aufruf des Konstruktors **RSA_Private_Operation** wird der Blinder einmalig initialisiert. Bei jeder privaten RSA-Operation wird der Wert für RSA-Blinding aktualisiert, sehr ähnlich zu LibreSSL. Im Vergleich zu LibreSSL werden die Blinding-Werte aber *nicht* neu initialisiert. Neue Blinding Werte $\mathbf{A}$ und $\mathbf{A}_{inv}$ werden lediglich wie folgt berechnet: $A' \equiv A \cdot A \mod N$ und $A'_{inv} \equiv A_{inv} \cdot A_{inv} \mod N$.

Wir schlagen vor, die Blinding-Werte wie bei LibreSSL neu zu initialisieren und schätzen den Aufwand dafür mit einem Personentag ein.

LibreSSL

Die relevanten Funktionen befinden sich in folgenden Dateien:

- **crypto/rsa/rsa_eay.c: RSA_eay_private_encrypt, RSA_eay_private_decrypt** (Z. 346, 482): RSA-Operationen mit privatem Exponenten (RSA-Signaturerstellung und -Entschlüsselung).
- **crypto/bn/bn_blind.c: BN_BLINDING_create_param** (Z. 319): Aufruf der Parameter-Generierung, erzeugt neue Elemente $\mathbf{A}$ und $\mathbf{A}_{inv}$, wobei $A \cdot A_{inv} \equiv 1 \mod N$ mit der Funktion `BN_rand_range`, so dass $0 \leq A < N$.
- **crypto/bn/bn_blind.c: BN_BLINDING_convert_ex** (Z. 231): Modulare Multiplikation mit den Blinding-Parametern.
- **crypto/bn/bn_blind.c: BN_BLINDING_update** (Z. 192): Die Blinding-Parameter können bis zu 32 mal aktualisiert werden, so dass jede RSA-Operation mit unterschiedlichen Blinding-Parametern ausgeführt wird. Bei der Aktualisierung werden $\mathbf{A}$ und $\mathbf{A}_{inv}$ neu gesetzt: $A' \equiv A \cdot A \mod N$ und $A'_{inv} \equiv A_{inv} \cdot A_{inv} \mod N$. Nach 32 Aktualisierungen wird wieder Funktion `BN_BLINDING_create_param` ausgeführt, die komplett neue Elemente $\mathbf{A}$ und $\mathbf{A}_{inv}$ erzeugt.

Bei dem ersten Aufruf der Funktion

rsa_get_blinding(rsa, &local_blinding, ctx)

wird die Funktion `BN_BLINDING_create_param` ausgeführt. `BN_BLINDING_create_param` generiert eine zufällige Zahl und ihre Inverse für die Blinding-Parameter (Funktionen `BN_rand_range` und `BN_mod_inverse`). Damit sind die Blinding-Parameter initialisiert. Anschließend kann die Funktion

rsa_blinding_convert(blinding, f, unblind, ctx) (Z. 553)

vor der Entschlüsselung und die Funktion

rsa_blinding_invert(blinding, ret, unblind, ctx) (Z. 589)

nach der Entschlüsselung ausgeführt werden.

Bei jeder neuen Blinding-Operation (in der Funktion **BN_BLINDING_convert_ex**) werden die Blinding-Parameter aktualisiert (Funktion **BN_BLINDING_update**). Bei der Aktualisierung werden die Elemente **A** und **A_{inv}** nicht komplett neu generiert, sie werden anhand von den bestehenden Elementen wie folgt berechnet: $A' \equiv A \cdot A \bmod N$ und $A'_{inv} \equiv A_{inv} \cdot A_{inv} \bmod N$. Damit wird gewährleistet, dass $A' \cdot A'_{inv} \equiv 1 \bmod N$. Jede neue RSA-Operation wird somit mit unterschiedlichen Blinding-Parametern ausgeführt. Nach 32 RSA-Operationen (die Zahl 32 wird mit **BN_BLINDING_COUNTER** definiert) werden mit der Funktion **RSA_setup_blinding** die Blinding-Parameter komplett neu generiert.

Ein ähnliches Verhalten war auch bei der Signaturerstellung zu sehen.

NSS

Die relevanten Funktionen befinden sich in folgenden Dateien:

- **lib/freebl/rsa.c: generate_blinding_params** (Z. 1034): Diese Funktion erzeugt neue Elemente **A** und **A_{inv}**, wobei $A \cdot A_{inv} \equiv 1 \bmod N$.
- **lib/freebl/rsa.c: rsa_PrivateKeyOp** (Z. 1274): private RSA-Operation benutzt die generierten Blinding-Parameter.

Bei dem Aufruf des Konstruktors **rsa_PrivateKeyOp** werden die Blinding-Parameter eingesetzt. Die Parameter werden 50 mal benutzt (definiert mit **RSA_BLINDING_PARAMS_MAX_REUSE** 50). Die Werte werden dazwischen nicht aktualisiert.

2.2.1.2 Bleichenbacher-Angriff

Der Bleichenbacher-Angriff [Blei] gehört zu einer Klasse von adaptiven Chosen-Ciphertext-Angriffen. Dabei sendet der Angreifer veränderte Chiffretexte an den Server und beobachtet die Server-Antworten, welche es erlauben, Rückschlüsse auf die Nachrichtenvalidität zu ziehen. Bei jeder Server-Antwort lernt der Angreifer etwas über den Chiffretext.

Wenn der Angreifer keine Möglichkeit bekommt, valide von invaliden Nachrichten zu unterscheiden, kann er diesen Angriff nicht durchführen. Daher ist es nötig, dass der Server alle Anfragen mit gleichen Antworten bearbeitet, welche in konstanter Zeit produziert werden.

Um konkret den Bleichenbacher-Angriff zu verhindern, wurde folgender Abschnitt im TLS 1.2-Standard [RFC5246] hinzugefügt:

1. Generate a string R of 46 random bytes
2. Decrypt the message to recover the plaintext M
3. If the PKCS#1 padding is not correct, or the length of message M is not exactly 48 bytes:
 pre_master_secret = ClientHello.client_version || R

```

        else If ClientHello.client_version <= TLS 1.0, and
version number check is explicitly disabled:
            pre_master_secret = M
        else:
            pre_master_secret = ClientHello.client_version ||
M[2..47]

```

Damit ist gewährleistet, dass ein PremasterSecret immer in konstanter Zeit generiert wird. Wenn die Struktur der Nachricht nach der Entschlüsselung inkorrekt ist, wird als PremasterSecret eine zufällig gezogene Zahl verwendet. Diese muss immer erzeugt werden, unabhängig davon ob die entschlüsselte Struktur korrekt war (vgl. Zeile 1).

Im Folgenden wird analysiert, ob diese Gegenmaßnahme in den analysierten Bibliotheken korrekt implementiert wurde.

Botan

Die relevanten Funktionen befinden sich in folgenden Dateien:

- **lib/tls/msg_client_kex.cpp: Client_Key_Exchange::Client_Key_Exchange** (Z. 258): ClientKeyExchange-Bearbeitung, Generierung einer zufälligen Zahl, Aufruf der PKCS#1-Entschlüsselung, Überprüfung der TLS-Version in der entschlüsselten Struktur.
- **lib/pk_pad/eme_pkcs1/eme_pkcs.cpp: EME_PKCS1v15::unpad** (Z. 43): PKCS#1-Unpadding

Mithilfe von TLS-Attacker haben wir versucht, verschiedene Fehlermeldungen nach der Entschlüsselung von invaliden PKCS#1-/TLS-Nachrichten zu provozieren.

Eine valid-formatierte PKCS#1-Nachricht hat im TLS-Protokoll folgende Struktur:

0x00 0x02 padding 0x00 pms

Das PremasterSecret pms ist dabei 48 bytes lang. Das Padding ist so gewählt, dass die gesamte Nachricht die Länge des RSA-Schlüssels erreicht. Das Padding beinhaltet kein 0x00 Byte. Die ersten zwei Bytes des PremasterSecrets entsprechen der TLS-Version. Folgende Änderungen können anhand von dieser Struktur berücksichtigt werden:

- Die ersten zwei Bytes werden geändert, so dass die Nachricht nicht mit 0x00 0x02 anfängt. Wenn der Server in solchem Fall mit einem Fehler antwortet, führt es zu einem direkten Bleichenbacher-Angriff, so wie in dem ursprünglichen Artikel beschrieben wurde [Blei].
- In das Padding wird an verschiedenen Stellen ein 0x00 Byte eingesetzt. Dies führt dazu, dass das PremasterSecret verlängert wird und damit eine invalide Länge bekommt.
- Die Länge des PremasterSecrets wird verkürzt, in dem das 0x00 Byte in der Mitte verschoben wird.
- Das 0x00 Byte in der Mitte wird gelöscht (zusammen mit anderen 0x00 Bytes im PremasterSecret), so dass der Server die Grenze zwischen dem PremasterSecret und dem Padding nicht erkennen kann.

- Die TLS-Version in dem PremasterSecret wird invalidiert.

Alle diese Veränderungen könnten ein invalides Server-Verhalten provozieren.

Unsere Tests haben ergeben, dass der Server korrekt alle Nachrichten bearbeitet, ein zufälliges PremasterSecret erzeugt, und erst nach der Überprüfung der Finished-Nachricht einen Fehler liefert. Damit kann kein direkter Seitenkanal für den Bleichenbacher-Angriff ausgenutzt werden.

Eine Analyse des Source-Codes hat ergeben, dass die Bibliothek PKCS#1 nicht Zeit-konstant arbeitet. An mehreren Stellen werden Exceptions geworfen, welche zu Timing-basierten Seitenkanälen führen könnten. Obwohl diese Seitenkanäle höchstwahrscheinlich zu keinen praktischen Angriffen führen würden, empfehlen wir diese zu beheben und schätzen den Aufwand mit 12 Personentagen ab.

LibreSSL

Mithilfe von TLS-Attacker haben wir versucht, verschiedene Fehlermeldungen nach der Entschlüsselung von invaliden PKCS#1-/TLS-Nachrichten zu provozieren. Dies war nicht möglich und der Server hat bei jedem Versuch mit einem Alert geantwortet.

Die Analyse der `crypto/rsa_pk1.c` Datei hat ergeben, dass die Funktion `RSA_padding_check_PKCS1_type_2` nicht in konstanter Zeit Ergebnisse liefert:

```
int RSA_padding_check_PKCS1_type_2(unsigned char *to, int tlen,
    const unsigned char *from, int flen, int num)
{
    int i,j;
    const unsigned char *p;
    p=from;
    if ((num != (flen+1)) || (*(p++) != 02))
    {
        RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_2, RSA_R_BLOCK_TYPE_IS_NOT_02);
        return(-1);
    }
#ifdef PKCS1_CHECK
    return(num-11);
#endif
    /* scan over padding data */
    j=flen-1; /* one for type. */
    for (i=0; i<j; i++)
        if (*(p++) == 0) break;

    if (i == j)
    {
        RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_2, RSA_R_NULL_BEFORE_BLOCK_MISSING);
    }
}
```

```

        return(-1);
    }
    if (i < 8)
    {
        RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_2, RSA_R_BAD_PAD_BYTE_COUNT);
        return(-1);
    }
    i++; /* Skip over the '\0' */
    j-=i;
    if (j > tlen)
    {
        RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_2, RSA_R_DATA_TOO_LARGE);
        return(-1);
    }
    memcpy(to,p,(unsigned int)j);
    return(j);
}

```

Wie in dem Code zu sehen ist, bricht der Server früher ab, wenn z.B. das zweite Byte der Nachricht kein 0x02 Byte an der zweiten Position enthält oder wenn die Nachricht kein 0x00 Byte hat. Dieses Verhalten wurde schon in der ersten Analyse [OpenSSLA] beobachtet, anhand von dem Code von OpenSSL 1.0.1g. Obwohl der resultierende Zeitunterschied sehr klein und damit unpraktisch wird, empfehlen wir die Implementierung zu fixen und den Seitenkanal zu schließen. Der Seitenkanal kann nämlich zu größeren Auswirkungen führen, z.B. wenn Szenarien mit Flush-Reload Angriffen [FLUSH] ermöglicht werden.

In der OpenSSL 1.0.1j Version wurden einige dieser Probleme behoben und die PKCS#1 Validierung zeitkonstant neu implementiert. Damit sollten diese Seitenkanäle beseitigt werden. Wir empfehlen in LibreSSL den Code von OpenSSL zu übernehmen und schätzen den Aufwand mit sechs Personentagen ab.

NSS

Die relevanten Funktionen befinden sich in folgenden Dateien:

- **lib/ssl/ssl3con.c: ssl3_HandleRSAClientKeyExchange** (Z. 9199): ClientKeyExchange-Bearbeitung, Aufruf der PKCS#1-Entschlüsselung, Überprüfung der TLS-Version in der entschlüsselten Struktur, Generierung einer zufälligen Zahl
- **lib/freebl/rsapkcs.c: RSA_DecryptBlock** (Z. 903): PKCS#1-Unpadding

Mithilfe von TLS-Attacker haben wir versucht, verschiedene Fehlermeldungen nach der Entschlüsselung von invaliden PKCS#1-/TLS-Nachrichten zu provozieren. Dies war nicht möglich und der Server hat bei jedem Versuch mit einem Alert geantwortet.

Eine Analyse des Source-Codes hat ergeben, dass die Bibliothek PKCS#1 nicht zeitkonstant arbei-

tet. An mehreren Stellen finden Verzweigungen statt, so dass der Code von den Secrets abhängig ist. Dies könnte zu Timing-basierten Seitenkanälen führen. Obwohl diese Seitenkanäle höchstwahrscheinlich zu keinen praktischen Angriffen führen würden, empfehlen wir diese zu fixen. Zum Beispiel in der Zeile 9302 wird ein neues PMS generiert (Datei lib/ssl/ssl3con.c):

```
pms = ssl3_GenerateRSAPMS(ss, ss->ssl3.prSpec, slot);
```

Dies passiert nur in dem Fall, wenn die PKCS#1v15-Entschlüsselung fehlgeschlagen ist. Bei der Entschlüsselung werden die ersten zwei PKCS#1 Bytes überprüft. Das Resultat der Entschlüsselung kann beliebig lang sein. Es wird nicht überprüft, ob das minimale Padding von 8 Bytes eingehalten wird. Wir schätzen den Aufwand zur Behebung der Probleme mit 12 Personentagen ab.

2.2.1.3 Invalid Curve Angriffe

Für die Berechnung gemeinsamer Schlüssel über elliptische Kurven werden Kurven mit großer Ordnung verwendet. Standardisierte Kurven sind z.B. von NIST (secp224r1, secp256r1, secp384r1) definiert worden. Wenn der Angreifer es schafft, das Opfer auf einer anderen Kurve mit kleiner Ordnung rechnen zu lassen, kann er unter Umständen auch den privaten Schlüssel des Opfers extrahieren.

Um diese Angriffe zu verhindern, muss die Implementierung immer überprüfen, ob der eingehende Punkt auf der Kurve liegt. In diesem Abschnitt wird dies analysiert.

Botan

Die relevanten Funktionen für die Bearbeitung von Kurven über endlichen Körpern der Form F_p befinden sich in folgenden Dateien:

- **lib/math/ec_gfp/point_gfp.cpp: OS2ECP** (Z. 498): Punkt-Dekodierung, Aufruf der Überprüfung ob der eingehende Punkt auf der Kurve liegt.
- **lib/pubkey/ecdh/ecdh.cpp: raw_agree** (Z. 33): ECDH Operation, Aufruf der Überprüfung ob der eingehende Punkt auf der Kurve liegt.
- **lib/math/ec_gfp/point_gfp.cpp: on_the_curve** (Z. 368): Überprüfung, ob der Punkt auf der Kurve liegt.

Unsere Analyse hat ergeben, dass die Implementierung überprüft, ob der eingehende Punkt auf der Kurve liegt. Die Validierung findet direkt nach der Punktdekodierung als auch nach der Punkt-Multiplikation in der ECDH-Operation statt. Damit ist es für den Angreifer *nicht* möglich, das Opfer dazu zu zwingen, mit einem falschen Punkt eine Berechnung zu starten.

LibreSSL

Die relevanten Funktionen für die Bearbeitung von Kurven über endlichen Körpern der Form F_p befinden sich in folgenden Dateien:

- **crypto/ecc/ecp_oct.c: int ec_GFp_simple_oct2point** (Z. 306): Punkt-Dekodierung, Aufruf der Überprüfung ob der eingehende Punkt auf der Kurve liegt.
- **crypto/ecc/ec_lib.c: int EC_POINT_is_on_curve** (Z. 980): Aufruf der Überprüfung,

ob der eingehende Punkt auf der Kurve liegt.

- **crypto/ec/ecp_smpl.c: int ec_GFp_simple_is_on_curve** (Z. 980): Überprüfung, ob der Punkt auf der Kurve liegt.

Unsere Analyse hat ergeben, dass die Implementierung überprüft, ob der eingehende Punkt auf der Kurve liegt. Der Aufruf der Validierungsfunktion wird aus der Funktion `ec_GFp_simple_oct2point` gestartet, welche für die Punkt-Dekodierung zuständig ist. Damit ist es für den Angreifer *nicht* möglich, das Opfer dazu zu zwingen, mit einem falschen Punkt eine Berechnung zu starten.

Die analysierten Funktionen decken sich mit den Funktionen ab, die in der OpenSSL-Analyse gefunden wurden.

NSS

Die relevanten Funktionen für die Bearbeitung von Kurven über endlichen Körpern der Form F_p befinden sich in folgenden Dateien:

- **lib/freebl/ec/ecp_aff.c: ec_GFp_validate_point** (Z. 249): Wird vor der Multiplikation ausgeführt. Validiert, ob der Punkt auf der Kurve liegt.
- **lib/freebl/ec/ecp_aff.c: ECPPoint_validate** (Z. 363): Führt eine spezifische Validierungsfunktion aus, z.B. `ec_GFp_validate_point`.
- **lib/softtoken/pkcs11.c: CK_RV NSC_DeriveKey** (Z. 5800): Führt EC Key Derivation aus. Der Punkt wird vor der Derivation überprüft.

Unsere Analyse hat ergeben, dass die Implementierung überprüft, ob der eingehende Punkt auf der Kurve liegt. Die Validierung findet direkt nach der Punktdkodierung statt. Damit ist es für den Angreifer *nicht* möglich, das Opfer dazu zu zwingen, mit einem falschen Punkt eine Berechnung zu starten.

2.2.1.4 Small Subgroup Angriffe

Bei der Generierung von zyklischen Gruppen Z_p (z.B. für DH Schlüsselaustausch) muss auf die Sicherheit der Parameter geachtet werden. Dabei sollte p eine „safe Prime“ sein, so dass $(p-1)/2=q$ auch prim ist. Der Generator g muss eine Ordnung $(p-1)$ haben.

Botan

Die Funktionen für die Generierung und Importierung von sicheren zyklischen Gruppen sind in folgenden Dateien zu finden:

- **lib/pubkey/dl_group/dl_group.cpp**

Die zyklischen Gruppen können mit einem Parameter Strong generiert werden. Dabei wird als p eine strong prime generiert. Zuerst wird dabei der Parameter q generiert, p wird dann als $p=2q+1$ nachgerechnet. Es wird anschließend überprüft, ob p und q prim sind.

Als Generator wird bei der Strong-Generierung der Wert $g=2$ genommen. Es wird nicht überprüft, ob dieser eine Ordnung von $(p-1)$ hat. Daher sollte ein zusätzlicher Test hinzugefügt werden, ob $g^q=1 \bmod p$. Falls dies der Fall ist, sollte ein anderer Wert als Generator genommen werden. Wir

schätzen den Aufwand hierfür mit 2 Personentagen ab.

Anzumerken ist noch, dass eine Generierung von 512 Bits großen Gruppen erlaubt ist (siehe Zeile 45 in `dl_group.cpp`). Es sollte daher näher evaluiert werden, ob die Gruppengröße in anderen Funktionen überprüft wird und welchen Einfluss dies auf die gesamte Schnittstelle hat.

LibreSSL

Die Funktion für die Generierung von sicheren zyklischen Gruppen befindet sich in der folgenden Datei:

- `crypto/dh/dh_gen.c`

Die Gruppen-Parameter werden in der Funktion `dh_builtin_genparams` generiert. Diese generiert eine safe prime p , und anschließend sucht nach einem geeigneten Generator (ob dieser Ordnung $(p-1)$ hat).

NSS

Die Funktionen für die Generierung und Importierung von sicheren zyklischen Gruppen sind in folgenden Dateien zu finden:

- `lib/freebl/dh.c`: `DH_GenParam` (Z.45)

In NSS wird für die Generierung eine strong prime benutzt. Es wird überprüft, ob der Generator g eine Ordnung $(p-1)$ hat.

2.2.1.5 Padding Oracle Angriff auf CBC

Padding Oracle Angriffe auf das CBC Mode of Operation können ermöglicht werden, wenn der Angreifer anhand von Seitenkanälen erkennen kann, ob der entschlüsselte Plaintext ein valides Padding beinhaltet.

In den meisten Fällen werden diese Angriffe damit verhindert, dass der Chiffretext authentiziert wird. Somit werden Veränderungen am Chiffretext noch vor der Entschlüsselung erkannt, wenn die Signatur bzw. der MAC nicht korrekt sind.

Ein Spezialfall stellt TLS vor. In diesem Standard ist vorgeschrieben, dass die Daten zuerst entschlüsselt werden, erst dann wird MAC verifiziert. Dies hat in der letzten Zeit zu Komplikationen geführt. Im Folgenden werden wir speziell auf CBC-Padding Behandlung im Zusammenhang mit TLS eingehen.

Botan

Die relevanten Funktionen befinden sich in folgenden Dateien:

- `lib/tls/tls_record.cpp`: `decrypt_record` (Z. 353): Diese Funktion entschlüsselt und verifiziert den Inhalt eines TLS-Records.

Der Code führt in einer konstanten Zeit zuerst die Entschlüsselung und MAC-Validierung aus. Wenn ein Fehler in einer dieser zwei Funktionen aufgetreten ist, antwortet der TLS-Server mit einer `BAD_RECORD_MAC` Nachricht.

Allerdings beinhaltet der Code nach der Entschlüsselung eine Überprüfung, ob die entschlüsselten

Daten genug Inhalt für einen MAC haben:

```
if(record_len < mac_pad_iv_size)
    throw Decoding_Error("Record sent with invalid length");
```

Dies führt zu einer anderen Alert-Nachricht (DECODE_ERROR), die von einem Angreifer für praktische Padding-Oracle-Angriffe in einem BEAST-Szenario ausnutzen kann. Wir schätzen die Behebung dieses Problems mit zwei Personentagen ab.

LibreSSL

Die relevanten Funktionen befinden sich in folgenden Dateien:

- **lib/ssl/s3_cbc.c: tls1_cbc_remove_padding** (Z. 146): Diese Funktion entfernt das Padding.

Die Funktion der Padding-Entfernung ist Zeit-konstant und wurde als Reaktion auf die Lucky13-Angriffe implementiert: <https://www.imperialviolet.org/2013/02/04/luckythirteen.html>

NSS

Die relevanten Funktionen befinden sich in folgenden Dateien:

- **lib/ssl/ssl3con.c: ssl_RemoveTLSCBCPadding** (Z. 11150): Entfernt Padding.
- **lib/ssl/ssl3con.c: ssl_CBCEExtractMAC** (Z. 11212): Validiert und entfernt den MAC.

Diese Funktionen kümmern sich um das CBC-Padding und um die MAC-Validierung. Sie sind Zeit-konstant. Beim Testen des TLS-Servers konnte keine relevante Fehlermeldung provoziert werden, welche die Seitenkanal-Angriffe ermöglicht hätte.

2.2.2 Portierbarkeit

In diesem Abschnitt wird die Plattformunterstützung der Bibliotheken untersucht. Aus der Herstellerabfrage haben sich die Plattformen Windows, Linux, iOS und Android als zu unterstützende Plattformen ergeben.

Botan

Botan unterstützt mit AIX, Android, iOS, Linux, FreeBSD, HP-UX, NetBSD, OpenBSD, OS X, Solaris, QNX und Windows eine ganze Reihe unterschiedlicher Plattformen. Dies erreicht es vor allem durch die extensive Nutzung von C++11-Features.

Unter Windows haben wir Botan mit Visual Studio 2013 und Visual Studio 2015 erfolgreich getestet. Auch unter Linux konnten wir Botan mit Clang 3.6.2 und GCC 5.2.0 erfolgreich kompilieren und die mitgelieferte Testsuite ausführen. Die im Handbuch veröffentlichten Anweisungen für die Kompilation für iOS haben mit der getesteten Version 1.11.18 nicht funktioniert, da das erzeugte Makefile Fehler enthielt. Die Fehler konnten jedoch in Zusammenarbeit mit den Entwicklern behoben² und eine neue Vorabversion erfolgreich kompiliert werden. Da zum Zeitpunkt der Untersu-

² <https://github.com/randombit/botan/issues/188>

chung kein iOS-Gerät mit Jailbreak zur Verfügung stand, konnte die Testsuite nicht ausgeführt werden. Den Aufwand, die Testsuite unter iOS zu testen und ggf. Fehler zu beheben schätzen wir mit einem Personentag ab. Auch für Android stehen Anweisungen für die Kompilation zur Verfügung, welche erfolgreich getestet werden konnten. Den Aufwand, die Testsuite unter Android zu testen und ggf. Fehler zu beheben schätzen wir mit einem Personentag ab.

LibreSSL

LibreSSL unterstützt mit AIX, Linux, FreeBSD, HP-UX, NetBSD, OpenBSD, OS X, Solaris und Windows eine ganze Reihe unterschiedlicher Plattformen. Dabei wird zwischen den Projekten LibreSSL (nur für OpenBSD) und LibreSSL Portable (für alle anderen Plattformen) unterschieden.

Unter Windows haben wir LibreSSL Portable erfolgreich mit Visual Studio 2013 getestet. Auch unter Linux konnten wir LibreSSL Portable mit Clang 3.6.0 und GCC 4.9.2 erfolgreich kompilieren und die mitgelieferte Testsuite ausführen. iOS wird offiziell nicht unterstützt. Da offiziell OS X unterstützt wird nehmen wir an, dass eine Portierung auf iOS mit vertretbarem Aufwand umsetzbar ist, da sich beide Systeme grundsätzlich ähneln, und schätzen den Aufwand mit fünf Personentagen ab. Auch Android wird offiziell nicht unterstützt. Da jedoch Linux unterstützt wird nehmen wir an, dass auch eine Portierung auf Android mit vertretbarem Aufwand umsetzbar ist, da sich beide Systeme grundsätzlich ähneln und schätzen der Aufwand mit fünf Personentagen ab.

NSS

NSS unterstützt offiziell Unix/Linux, Windows, OS X, Solaris, OS/2 und eComStation.

Unter Windows konnte NSS erfolgreich mit Visual Studio 2013 getestet werden. Auch unter Linux konnten wir NSS mit Clang 3.6.0 und GCC 4.9.2 erfolgreich kompilieren und die mitgelieferte Testsuite ausführen. iOS wird offiziell nicht unterstützt. Da offiziell OS X unterstützt wird nehmen wir an, dass eine Portierung auf iOS mit vertretbarem Aufwand umsetzbar ist, da sich beide Systeme grundsätzlich ähneln und schätzen den Aufwand mit fünf Personentagen ab. Android wird offiziell unterstützt. Den Aufwand, die Testsuite unter Android zu testen und ggf. Fehler zu beheben schätzen wir mit zwei Personentag ab.

2.2.3 Design und Komplexität

Design und Komplexität sind ein wichtiges Maß für die Qualität einer Bibliothek. Ein gutes, modulares Design erleichtert Erweiterungen und erhöht die Verständlichkeit der Bibliothek. Eine begrenztes Maß an Komplexität verringert die Fehleranfälligkeit der Implementierung. In diesem Abschnitt werden daher die Austauschbarkeit und Blockierung bestimmter Funktionen, z.B. durch Kompilierung, der Integrationsaufwand in Anwendungen, die Modularisierung, die Komplexität der Funktionen und die Unterstützung von Engine- oder ähnlichen Konzepten zur Unterstützung von alternativen Implementierungen, bspw. zur Einbindung von kryptographischer Hardware. Die detaillierten Messergebnisse finden sich im Anhang in Abschnitt 3.1.

2.2.3.1 Botan

Botan bietet unter den drei betrachteten Bibliotheken die größte Flexibilität bzgl. der Konfiguration. Dies ist in Botans großer Anzahl von Modulen begründet³, die über ein Konfigurationsskript sta-

3 adler32 aead aes aes_ni aes_ssse3 alloc aont asm_x86_64 asn1 auto_rng base base64 bcrypt benchmark bigint

tisch aktiviert bzw. deaktiviert werden können. Es ist vergleichbar zu der Flexibilität von OpenSSL, jedoch deutlich übersichtlicher, da z.B. direkt eine Liste von Modulen einsehbar ist. Botan beinhaltet 71675 Code-Zeilen. Die Funktionalitäten und Klassen sind sauber in Pakete verteilt. Die Struktur ist gut nachvollziehbar und gesuchte kryptographische Features können sehr schnell gefunden werden. In den Paketen selbst gibt es keine Makefiles. Es gibt nur eine zentrale Make-Datei im Root-Verzeichnis, so dass es keine Möglichkeit gibt nur an einem Paket zu entwickeln und dieses mit einem lokalen Makefile zu kompilieren. Dies ist auch nicht vorgesehen, da bestimmte Funktionalitäten bei der Kompilierung mit Compiler-Flags entfernt werden können, wie oben beschrieben wurde. Wir gehen davon aus, dass dieses Konzept ausreichend ist und kein Bedarf an Veränderungen besteht.

Der Integrationsaufwand wurde anhand der Anwendbarkeit in Apache und Nginx geschätzt. Bei Botan ist dieser deutlich größer als bei LibreSSL und NSS. So gibt es beispielsweise kein `mod_botan` und dieses müsste vollständig neu geschrieben werden. Auch für Nginx konnte keine einfache Integrationsmöglichkeit gefunden werden.

Botan implementiert ein sogenanntes Provider-Modell. Zur Kompilezeit können alternative Implementierungen registriert werden, die dann über eine Factory zur Laufzeit ausgewählt werden können. Es existiert bereits ein OpenSSL-Provider, der mit dem Konfigurationsflag `–with-openssl` zur Kompilation ausgewählt werden kann. Dieser Provider stellt bereits eine Implementierung der Algorithmen AES, DES, Blowfish, CAST, IDEA, SEED, ECDSA, SHA-1, SHA-2, MD2, MD4, MD5, RIPEMD, RC4 und RSA zur Verfügung.

2.2.3.2 LibreSSL

Von Seiten der Integrationsmöglichkeiten ist LibreSSL recht komfortabel. Das `mod_ssl` kann relativ einfach unter Verwendung LibreSSL (anstelle von `openssl`) erstellt werden. Durch einen Wrapper zu OpenSSL bietet es auch sehr gute Kompatibilität zu diesem. Die Flexible Deaktivierung von Modulen bekannt aus OpenSSL wurde in LibreSSL nahezu vollständig entfernt. Lediglich sehr wenige Compilerflags konnten identifiziert werden⁴.

LibreSSL beinhaltet 264926 Code-Zeilen und besteht aus 7 Modulen. Diese sind sauber getrennt und beinhalten keine zyklische Abhängigkeiten. Wir gehen auch hier davon aus, dass es keinen Bedarf an Veränderungen gibt.

LibreSSL implementiert das gleiche Engine-Konzept wie OpenSSL. Jedoch wurde hier die Mög-

block blowfish camellia cascade cast cbc cbc_mac ccm cfb chacha chacha20poly1305 clmul cmac codec_filt
comb4p compression crc24 crc32 credentials cryptobox ctr curve25519 datastor des dev_random dh dl_algo
dl_group dlies dsa eax ec_gfp ec_group ecb ecc_key ecdh ecdsa egd elgamal eme_oaep eme_pkcs1 eme_raw
emsa1 emsa1_bsi emsa_pkcs1 emsa_pssr emsa_raw emsa_x931 entropy fd_unix ffi filters fpe_fe1 gcm gost_28147
gost_3410 gost_3411 has160 hash hash_id hex hkdf hmac hmac_drbg hmac_rng hres_timer http_util idea idea_sse2
if_algo kasumi kdf rc41 kdf2 keccak keypair lion locking_allocator mac mars mce mceies md2 md4 md5
mdx_hash mgf1 misty1 mode_pad modes mp mp_x86_64 noekeon noekeon_simd nr numbertheory ocb ofb
oid_lookup openpgp par_hash passhash9 pbes2 pbkdf pbkdf1 pbkdf2 pem pk_pad poly1305 prf_tls prf_x942
proc_walk pubkey rc2 rc4 rc5 rc6 rdrand rfc3394 rfc6979 rmd128 rmd160 rng rsa rw safer salsa20 seed serpent
serpent_simd sessions_sql sha1 sha1_sse2 sha1_x86_64 sha2_32 sha2_64 simd simd_sse2 siphash siv skein srp6
stream system_rng tea threefish threefish_avx2 tiger tls tss twofish unix_procs utils whirlpool x509 x919_mac
x931_rng xtea xtea_simd xts

4 OPENSSL_NO_RC4, OPENSSL_NO_ASM, OPENSSL_NO_MD5

lichkeit entfernt, Engines dynamisch nachzuladen.

2.2.3.3 NSS

NSS bietet mit `mod_nss` eine fertige Integrationsmöglichkeit in Apache. Für Nginx müsste ein eigenes Modul geschrieben werden. NSS beinhaltet 671744 Code-Zeilen und besteht aus 32 Modulen, die stark von einander abhängig sind und viele gegenseitige und zyklische Abhängigkeiten beinhalten. Dies macht es schwer, bestimmte Module zu extrahieren und unabhängig einzusetzen. Es ist schwer abzuschätzen, wie viel Aufwand eine Entkopplung der Module bei so einem stark abhängigen Code dauern würde und ob dies überhaupt möglich wäre.

NSS implementiert kein Engine-Konzept. Teile der Implementierung können über PKCS#11-Module ausgetauscht werden.

2.2.3.4 Komplexität der Funktionen

Zur Analyse der Komplexität der Funktionen haben wir CppDepend eingesetzt. Dieses Tool analysiert beispielsweise Funktionen mit zu vielen Parametern oder zu lange Funktionen.

In unsere Analyse haben wir nur die *kritischen* Anmerkungen von CppDepend einfließen lassen. Dazu gehören:

- Komplexe Funktionen:
 - Beinhalten mehr als 20 Entscheidungsbäumen, siehe <http://www.cppdepend.com/Metrics#CC>
 - Beinhalten mehr als 5 Verschachtelungen, z.B. mit `if/for`
- Funktionen mit zu vielen Parametern: Wenn eine Funktion mehr als 8 Parameter beinhaltet, ist sie schwer aufzurufen. Funktionen mit mehreren Parametern verschlechtern auch die Performanz.

Um jede dieser kritischen Anmerkungen zu fixen, schätzen wir 0,5 Personentage ab, inkl. der damit verbundenen Tests.

CppDepend hat folgende Resultate ergeben:

	Botan	LibreSSL	NSS
Komplexe Funktionen	3	29	74
Funktionen mit zu vielen Parameter	16	40	101
Gesamt	19	69	175
Aufwand (Personentage)	10	35	88

2.2.4 Performance

Anhand eines selbstgeschriebenen Testprogramms (siehe Anhang) wurde eine kleine Auswahl Algorithmen einer Geschwindigkeitsmessung unterzogen. Dazu wurden die drei Bibliotheken unter Linux mit GCC 5.2.0 und eingeschalteten Optimierungen kompiliert. Die Tests wurden mittels eines

Intel i5-4670K Prozessors durchgeführt. Getestet wurden MD5, SHA-160, SHA-256, SHA-512, AES-128/GCM, AES-192/GCM, AES-256/GCM und AES-128/CBC ohne Padding jeweils mit unterschiedlichen Eingabelängen.

Hier zeigte sich, dass bei sämtlichen Hashberechnungen LibreSSL schneller als Botan und Botan schneller als NSS ist. Bei AES/GCM ist NSS performanter als LibreSSL und LibreSSL performanter als Botan. Die AES/CBC Tests zeigten, dass LibreSSL vor NSS und NSS vor Botan liegt. Die folgende Tabelle listet die genauen Ergebnisse:

Algorithmus	Botan	LibreSSL	NSS
MD5 16 byte	231.496 ns	118.362 ns	958.38 ns
MD5 64 byte	328 ns	203.522 ns	1047.88 ns
MD5 256 byte	602.058 ns	452.291 ns	1303.96 ns
MD5 1024 byte	1695.49 ns	1453.01 ns	2.40759 µs
MD5 8192 byte	11.9179 µs	10.7778 µs	12.6067 µs
SHA-160 16 byte	275.878 ns	123.694 ns	1021.13 ns
SHA-160 64 byte	399.921 ns	204.064 ns	1153.22 ns
SHA-160 256 byte	761.623 ns	434.614 ns	1547.15 ns
SHA-160 1024 byte	2.21599 µs	1329.17 ns	3.12008 µs
SHA-160 8192 byte	15.7458 µs	9.62546 µs	17.6529 µs
SHA-256 16 byte	443.53 ns	267.479 ns	1228.89 ns
SHA-256 64 byte	753.044 ns	484.428 ns	1575.78 ns
SHA-256 256 byte	1629.53 ns	1110.81 ns	2.45576 µs
SHA-256 1024 byte	5.23236 µs	3.60856 µs	6.1717 µs
SHA-256 8192 byte	37.7826 µs	27.3347 µs	39.8516 µs
SHA-512 16 byte	582.972 ns	336.319 ns	1315.98 ns
SHA-512 64 byte	561.111 ns	335.203 ns	1315 ns
SHA-512 256 byte	1349.6 ns	881.169 ns	2.07593 µs
SHA-512 1024 byte	3.61237 µs	2.43646 µs	4.33583 µs
SHA-512 8192 byte	24.7522 µs	17.5793 µs	25.3719 µs
AES128-GCM 262144 byte	512.978 µs	139.653 µs	77.9664 µs
AES192-GCM 262144 byte	514.12 µs	149.671 µs	91.2841 µs
AES256-GCM 262144 byte	521.471 µs	159.909 µs	95.9189 µs
AES128-CBC 262144 byte	417.651 µs	308.122 µs	314.116 µs

Da jedoch nur eine sehr kleiner Teil der Bibliotheken und nur auf einer Plattform getestet wurde haben diese Werte nur begrenzte Aussagekraft. Dennoch ist geplant diese Ergebnisse an die Botan-Entwickler weiterzuleiten.

2.2.5 Zufallszahlengenerator

In diesem Abschnitt werden die in den Bibliotheken vorgefundenen Zufallszahlengeneratoren untersucht.

Botan

Botan implementiert den HMAC_DRBG aus NIST SP 800-90A sowie die PRNGs aus RFC 5869 und ANSI X9.31. Die API bietet keine Möglichkeit einen festen Zufallszahlengenerator zu definieren. Alle API-Funktionen die Zufall benötigen erwarten eine Instanz eines der Zufallsgeneratoren als Argument. Der von der Dokumentation empfohlene AutoSeeded_RNG benutzt den PRNG aus RFC 5869 mit SHA-512. Initialisiert wird der AutoSeeded_RNG automatisch aus folgenden Quellen:

- Linux: die ersten 4 kB der ersten 2048 Dateien in /proc, 32 Byte aus /dev/random, Ausgabe von Kommandozeilentools wie ls, ps und netstat.
- Windows: Rückgabewerte von Windows-API Funktionen GetTickCount, GetMessagePos, GetMessageTime, GetInputState, GetCurrentProcessId, GetCurrentThreadId, GetSystemInfo, GlobalMemoryStatus, GetCursorPos, GetCaretPos, QueryPerformanceCounter, CreateToolhelp32Snapshot und CryptGenRandom.

Auf allen Plattformen wird zusätzlich der x86 Time Stamp Counter und die Ausgabe von RdRand in den Seed gemischt. Bei einem fork(2) wird **keiner** der PRNGs neu initialisiert.

Eigene PRNGs können implementiert und auch in allen API-Funktionen benutzt werden, da diese immer die Instanz eines Zufallsgenerators als Argument erwarten.

LibreSSL

LibreSSL ersetzt die RAND_* Implementierung von OpenSSL durch arc4random(3). Dieser generiert Zufallszahlen aus dem Schlüsselstrom von ChaCha20. Initialisiert wird arc4random(3) unter GNU/Linux aus folgenden Quellen:

1. getrandom(2)
2. urandom(4)
3. Werte der sysctl Variablen CTL_KERN, KERN_RANDOM und RANDOM_UUID.
4. SHA512 Hash der aktuellen Uhrzeit, Prozess und Session ID und (Session) ID des übergeordneten Prozesses, wartende und blockierte Signale aus sigpending(2) und sigprocmask(2), Inhalt mehrerer Heap Puffer und stat_t des aktuellen und des Root-Verzeichnisses.

Die Quellen werden in der angegebenen Reihenfolge abgearbeitet. Sobald eine Quelle Entropie erfolgreich liefert stoppt der Algorithmus. Die Linux API erlaubt die Nutzung von Hardwarezufallszahlengeneratoren für getrandom(2) und urandom(4). Dies ist aber nicht vorgeschrieben. Unter Linux wird der Zufallszahlengenerator nach einem fork(2) neu initialisiert. Auf Windows-Systemen wird CryptGenRandom verwendet. Die Initialisierung von arc4random(3) geschieht in allen Fällen automatisch. Die in OpenSSL noch vorhandene Möglichkeit den Zufallszahlengenerator mit RAND_set_en-

gine() zu ersetzen wurde in LibreSSL entfernt.

NSS

NSS implementiert den Hash_DRBG aus NIST SP 800-90A. Dieser ist der Standard PRNG. Je nach Plattform wird der Generator beim Aufruf von NSS_Init() aus folgenden Quellen initialisiert:

- Windows: RtlGenRandom, GlobalMemoryStatus, GetLogicalDrives, GetComputerName, GetCurrentProcess, GetCurrentProcessId, GetCurrentThreadId, GetVolumeInformation, GetDiskFreeSpace, CurrentClockTickTime, GetTickCount und time.
- GNU/Linux: /dev/urandom, sysinfo(2), Umgebungsvariablenamen und Werte, optional NSRANDFILE, der Inhalt folgender Dateien: /etc/passwd, /etc/utmp, /tmp, /var/tmp, /usr/tmp und die Ausgabe von "netstat -ni".

Der PRNG wird alle 2^{48} Iterationen neu initialisiert. Bei einem fork(2) wird der PRNG **nicht** neu initialisiert.

2.2.6 PKCS-Unterstützung

In diesem Abschnitt wird die Unterstützung der wichtigsten PKCS-Standards PKCS#1, PKCS#10, PKCS#11 und PKCS#12 untersucht.

Botan

Botan unterstützt bereits PKCS#1 und PKCS#10. Eine Unterstützung von PKCS#11 und PKCS#12 fehlt. Für eine Unterstützung von PKCS#11 in Botan schätzen wir den Aufwand mit 50 Personentagen ab, da eine Neuimplementierung nötig würde. Für eine Unterstützung von PKCS#12 in Botan schätzen wir den Aufwand mit 5 Personentagen ab, da der zugehörige RFC mit 14 Seiten nicht sehr umfangreich ist und ein ASN.1 Encoder/Decoder bereits in Botan vorhanden ist.

LibreSSL

LibreSSL unterstützt PKCS#1, PKCS#10 und PKCS#12. Auch hier fehlt die Unterstützung von PKCS#11. Das OpenSC-Projekt stellt jedoch eine OpenSSL-Engine zur Verfügung, mit der OpenSSL mit Unterstützung für PKCS#11 nachgerüstet werden kann. Teile dieser Engine könnten für LibreSSL wiederverwendet werden. Den Aufwand dafür schätzen wir mit 35 Personentagen ab.

NSS

NSS unterstützt als einzige Bibliothek alle vier Standards PKCS#1, PKCS#10, PKCS#11 und PKCS#12.

2.2.7 Testabdeckung

In diesem Abschnitt wird die Testabdeckung der Bibliotheken untersucht. Dazu werden die Bibliotheken unter Linux mit GCC und dem Compilerflag **-coverage** kompiliert und anschließend die mitgelieferte Testsuite ausgeführt. Dabei erzeugt GCC Datensätze über die aufgerufenen Funktionen. Aus diesen Datensätzen wird mit lcov eine Statistik erstellt, die dann die Abdeckung der im Kompilat vorhandenen Bibliotheksfunktionen durch die ausgeführten Tests aus der Testsuite in den

Metriken Abdeckung der Zeilen in Prozent und Abdeckung der Funktionen in Prozent angibt. In der Literatur finden sich unterschiedliche Angaben für eine ausreichende Testabdeckung [CODEMET, TIOBE], meist reichend von 70% bis 80%. Für eine Kryptobibliothek streben wir eine Testabdeckung von 80% an, sowohl für die Abdeckung der Zeilen als auch der Funktionen. Da der Aufwand, weitere Funktionen mit Tests abzudecken stark von der Art und Weise der Implementierung, der Komplexität und Abhängigkeiten zu anderen Funktionen abhängt und daher für jede Funktion einzeln betrachtet werden muss, wird im folgenden keine Schätzung in Personentagen angegeben. Stattdessen wird nur die Anzahl der Zeilen und Funktionen verglichen, die zusätzlich abgedeckt werden müssten um die angestrebte Testabdeckung von 80% zu erreichen. Die detaillierten Messergebnisse finden sich im Anhang in Abschnitt 3.2.

Botan

Botan's Testsuite deckt 77,2% (21282 von 27578) der Zeilen und 74,5% (4237 von 5690) der Funktionen der gesamten Bibliothek mit Tests ab. Um eine Testabdeckung von 80% zu erreichen müssten 780 weitere Zeilen und 315 weitere Funktionen mit Tests abgedeckt werden.

LibreSSL

LibreSSL's Testsuite deckt 28,3% (19905 von 70213) der Zeilen und 31,3% (1648 von 5270) Funktionen der gesamten Bibliothek mit Tests ab. Um eine Testabdeckung von 80% zu erreichen müssten 36265 weitere Zeilen und 2568 weitere Funktionen mit Tests abgedeckt werden.

NSS

NSS's Testsuite deckt 47,6% (58670 von 123333) der Zeilen und 58,0% (4144 von 7143) der Funktionen der gesamten Bibliothek mit Tests ab. Um eine Testabdeckung von 80% zu erreichen müssten 39996 weitere Zeilen und 1570 weitere Funktionen mit Tests abgedeckt werden.

Bibliothek	Anzahl zusätzlich abzudeckender Funktionen / Faktor
Botan	315 / 1x
LibreSSL	2568 / 8,15x
NSS	1570 / 4,98x

Tabelle 2.1: Anzahl der zusätzlich mit Tests abzudeckenden Funktionen zur Erreichung einer Testabdeckung von 80%

2.2.8 Dokumentation

Die Dokumentation ist ein wichtiger Bestandteil einer Kryptobibliothek. Zunächst muss die Anwenderschnittstelle (API) möglichst vollständig dokumentiert sein, damit der Anwender die Schnittstellen verstehen und anwenden kann. Hierfür hat sich das Doxygen-Format als Standard durchgesetzt. Ein Handbuch mit Hinweisen zur Verwendung von Bibliotheksfunktionen und kommentiertem Beispielcode ist meist erste Anlaufstelle für den Anwender. Nicht zuletzt ist auch die Kommentierung der Implementierung innerhalb der Kryptobibliothek wichtig für das Verständnis und die Lesbarkeit des Quelltextes.

Botan

Die Botan-API ist im Doxygen-Format in den meisten Teilen dokumentiert. In etwa 110 der 260

Headerdateien fanden wir unvollständige oder nicht vorhandene API-Dokumentation vor. Den Aufwand, dies zu beheben schätzen wir mit 11 Personentagen ab. Die Doxygen-API Dokumentation kann lokal erzeugt werden, liegt im HTML-Format für die jeweils aktuellste Version aber auch auf der Botan-Webseite vor⁵. Auch auf der Botan-Webseite steht ein Handbuch⁶ bereit. Es enthält neben Beschreibungen der API auch längere Beschreibungen der zugrundeliegenden Algorithmen oder Verfahren, bspw. zum Themenkomplex Password Hashing. Das Handbuch behandelt die Themen TLS, X.509, Zufallsgeneratoren, MACs, Password Hashing, AEAD, KDF, Cryptobox und SRP und weitere, konzeptionelle Bibliothekseigenheiten. Es enthält kommentierten Beispielcode zur Schlüsselerzeugung, zu CRLs, zu X.509 Zertifikaten, zu CSRs, zu PBKDF, bcrypt, und Format Preserving Encryption (FPE). An einigen Stellen ist das Handbuch nicht mehr aktuell. Während zu einigen Themenkomplexen ausführliche Dokumentation existiert, ist zu anderen Themen keine Dokumentation vorhanden. Den Aufwand, das Handbuch zu verbessern schätzen wir mit 18 Personentagen ab. Die Implementierung ist an vielen Stellen wenig dokumentiert und könnte dahingehend verbessert werden.

LibreSSL

Die LibreSSL-API ist in weiten Teilen nicht dokumentiert. Während die meisten Strukturdefinitionen schon dokumentiert sind, findet sich eine API-Dokumentation der Funktionen lediglich in den Headerdateien für Elliptische Kurven. In etwa 60 der 70 Headerdateien fanden wir unvollständige oder nicht vorhandene API-Dokumentation vor. Dies zu beheben schätzen wir mit 30 Personentagen ab. Es existiert kein Handbuch, lediglich man pages liegen für einige Funktionen vor⁷. Ein Handbuch zu erstellen schätzen wir mit 40 Personentagen ab. Die Implementierung ist an vielen Stellen wenig dokumentiert und könnte dahingehend verbessert werden.

NSS

Die NSS-API ist in den meisten Teilen dokumentiert. Das gewählte Format ist jedoch nicht Doxygen-kompatibel. In allen 327 Headerdateien fanden wir vollständige oder nahezu vollständige API-Dokumentation vor. Die API-Dokumentation zu komplettieren und in das Doxygen-Format zu wandeln schätzen wir aufgrund der großen Menge an Headern und Funktionen mit 14 Personentagen ab. Auf der NSS-Webseite stehen weitere Informationen, wie bspw. eine API-Referenz und Beispielcode bereit. Kurze Codebeispiele sind verfügbar zu den Themen Schlüsselerzeugung und Transport, Symmetrische Verschlüsselung, Hashfunktionen und MACs, PKI Verschlüsselung und NSS Schlüsselspeicher. Oft finden sich jedoch Links zu nicht existierenden Webseiten (Fehler 404) oder die Dokumentation ist doppelt vorhanden, wie bspw. bei den Kompilieranweisungen. Dies zu beheben und fehlende Dokumentation zu ergänzen schätzen wir mit 30 Personentagen ab. Die Implementierung ist an vielen Stellen gut dokumentiert.

2.2.9 Mechanismen zum sicheren Löschen von Geheimnissen

Eine Kryptobibliothek verarbeitet oft sensitive Daten, wie z.B. Schlüsselmaterial, im RAM. Die ei-

5 <http://botan.randombit.net/doxygen/>

6 <http://botan.randombit.net/manual/>

7 <http://www.openbsd.org/cgi-bin/man.cgi/OpenBSD-current/man1/openssl.1?query=openssl&sec=1>,
<http://www.openbsd.org/cgi-bin/man.cgi/OpenBSD-current/man3/crypto.3?query=crypto&sec=3>,
<http://www.openbsd.org/cgi-bin/man.cgi/OpenBSD-current/man3/ssl.3?query=SSL&sec=3>,
http://www.openbsd.org/cgi-bin/man.cgi/OpenBSD-current/man3/tls_accept_fds.3?query=tls_init&sec=3

nem Prozess zugewiesenen Speicherbereiche werden nach dessen Freigabe aus Effizienzgründen nicht gelöscht, das heißt in einen definierten Zustand versetzt. So kann es passieren, dass ein anderer Prozess den freigegebenen Speicher des ersten Prozesses zugewiesen bekommt und geheime Daten, wie Schlüsselmaterial auslesen kann. Um dies zu verhindern muss eine Kryptobibliothek sicherstellen, dass alle geheimen Daten die verarbeitet wurden sicher gelöscht, d.h. überschrieben werden. Dabei muss darauf geachtet werden, dass der Überschreibvorgang nicht durch den Compiler wegoptimiert wird. Es sollte außerdem sichergestellt werden, dass Speicherbereiche die sensitive Daten enthalten nicht in die Auslagerungsdatei (Swap) gelangen.

2.2.9.1 Botan

Botan nutzt für die Verarbeitung von sensitiven Daten die eigenen Typen `secure_vector` und `secure_deque`, deklariert in der Headerdatei `secmem.h`.

```
template<typename T> using secure_vector = std::vector<T,  
secure_allocator<T>>;  
template<typename T> using secure_deque = std::deque<T,  
secure_allocator<T>>;
```

Diese entsprechen den Containerklassen `std::vector` und `std::deque` und erweitern sie lediglich um einen benutzerdefinierten Allokator `secure_allocator`. Der Allokator garantiert, dass auf allen Plattformen beim Zerstören eines solchen Objektes durch den Destruktor der Inhalt des Containers überschrieben wird. Unter Windows wird dazu die API-Funktion `RtlSecureZeroMemory()` aufgerufen, auf allen anderen Plattformen wird `memset()` benutzt. Für `RtlSecureZeroMemory()` ist garantiert, dass ein Aufruf durch den MSVC-Kompiler nicht wegoptimiert wird. Da gleiches für `memset()` nicht gilt und die Funktion `memset_s()`, welche das Linux-Äquivalent zu `RtlSecureZeroMemory()` darstellt, in der glibc derzeit nicht unterstützt wird, stellt Botan durch einen volatile Funktionszeiger auf `memset()` sicher, dass der Aufruf nach `memset()` nicht wegoptimiert wird.

```
void zero_mem(void* ptr, size_t n)  
{  
    #if defined(BOTAN_TARGET_OS_HAS_RTLSECUREZEROMEMORY)  
        ::RtlSecureZeroMemory(ptr, n);  
    #elif defined(BOTAN_USE_VOLATILE_MEMSET_FOR_ZERO) &&  
        (BOTAN_USE_VOLATILE_MEMSET_FOR_ZERO == 1)  
        static void* (*const volatile memset_ptr)(void*, int, size_t) =  
std::memset;  
        (memset_ptr)(ptr, 0, n);  
    #else  
        volatile byte* p = reinterpret_cast<volatile byte*>(ptr);  
  
        for(size_t i = 0; i != n; ++i)  
            p[i] = 0;  
    #endif  
}
```

Auf Plattformen die `mlock()` unterstützen, das sind Linux, FreeBSD, NetBSD, OpenBSD und Solaris, wird sichergestellt, dass Containerinhalte nie auf die Festplatte ausgelagert werden. Unter Windows wird das nicht sichergestellt, obwohl dies mit den Windows API-Funktionen `VirtualLock()` und `VirtualUnlock()` möglich wäre. Den Aufwand hierfür schätzen wir mit einem Personentag ab.

2.2.9.2 LibreSSL

Ähnlich zu OpenSSL definiert LibreSSL in `crypto/mem_clr.c` die Funktion `OPENSSL_cleanse()` zum sicheren Löschen von sensiblen Daten. Diese ruft je nach Plattform lediglich unterschiedliche Implementierungen der Funktion `explicit_bzero()` auf. Unter Linux wird mittels `memset()` der übergebene Speicherbereich mit Nullen überschrieben. Durch die Deklaration der Funktion `__explicit_bzero_hook()` als weak function wird sichergestellt, dass der Compiler den Aufruf von `memset()` nicht durch Optimierung entfernt.

```
__attribute__((weak)) void
__explicit_bzero_hook(void *buf, size_t len)
{
}

void
explicit_bzero(void *buf, size_t len)
{
    memset(buf, 0, len);
    __explicit_bzero_hook(buf, len);
}
```

Unter Windows wird in `explicit_bzero()` die Windows API-Funktion `SecureZeroMemory()` aufgerufen. Diese ist äquivalent zur Funktion `RtlSecureZeroMemory()`, wie sie in Botan verwendet wird.

```
void
explicit_bzero(void *buf, size_t len)
{
    SecureZeroMemory(buf, len);
}
```

Für keine Plattform wird sichergestellt, dass Containerinhalte nicht auf die Festplatte ausgelagert werden. Den Aufwand hierfür schätzen wir mit zwei Personentagen ab.

2.2.9.3 NSS

NSS verwaltet Speicher in Memory Pools namens „Arenas“. Wenn ein solcher Speicherbereich freigegeben werden soll, wird die Funktion `nss_ZFreeIf()` aus `lib/base/arena.c` aufgerufen. Diese ruft wiederum die `nsslibc_memset()` aus `lib/base/libc.c` auf.

```
NSS_IMPLEMENT void *
nsslibc_memset
(
    void *dest,
    PRUint8 byte,
    PRUint32 n
)
{
    #ifdef NSSDEBUG
        if( ((void *)NULL == dest) ) {
            nss_SetError(NSS_ERROR_INVALID_POINTER);
            return (void *)NULL;
        }
    #endif /* NSSDEBUG */
```

```

    return memset(dest, (int)byte, (size_t)n);
}

```

Mittels `memset()` wird der übergebene Speicherbereich mit Nullen überschrieben (die Null wird `nss_ZfreeIf()` übergeben). Da `nsslibc_memset()` sich in einer anderen Kompilationseinheit als `nss_ZfreeIf()` befindet, gehen wir davon aus dass der Kompiler den Aufruf auf `memset()` nicht weg-optimiert.

Für keine Plattform wird sichergestellt, dass Containerinhalte nicht auf die Festplatte ausgelagert werden. Den Aufwand hierfür schätzen wir mit drei Personentagen ab.

2.2.10 Fehlende Algorithmen

In der Vorauswahl wurden die Bibliotheken bereits auf die Unterstützung der geforderten kryptographischen Algorithmen und Protokolle hin untersucht. Zusätzlich muss die Bibliothek die TLS-Erweiterung Encrypt-then-MAC unterstützen.

2.2.10.1 Botan

Von den geforderten Algorithmen und Protokollen fehlen in Botan eine KDF nach NIST SP800-56C, ECIES, ECKDSA, ECGDSA, Merkle-Signaturen und GMAC. Zusätzlich muss die TLS-Erweiterung Encrypt-then-MAC implementiert werden. Wir schätzen den Implementierungsaufwand folgendermaßen ab:

KDF nach NIST SP800-56C	10 PT
ECIES	12 PT
ECKDSA	12 PT
ECGDSA	12 PT
Merkle-Signaturen	25 PT
GMAC	3 PT
TLS-Erweiterung Encrypt-then-MAC	15 PT

2.2.10.2 LibreSSL

Von den geforderten Algorithmen und Protokollen fehlen in LibreSSL eine KDF nach NIST SP800-56C, ECIES, DLIES, ECKDSA, ECGDSA, Merkle-Signaturen, eigene Kurven, GMAC und ein RNG nach NIST SP800-90A. Zusätzlich muss die TLS-Erweiterung Encrypt-then-MAC implementiert werden. Wir schätzen den Implementierungsaufwand folgendermaßen ab:

KDF nach NIST SP800-56C	13 PT
ECIES	16 PT

DLIES	16 PT
ECKDSA	16 PT
ECGDSA	16 PT
Merkle-Signaturen	33 PT
Eigene Kurven GF(p)	33 PT
GMAC	5 PT
RNG nach NIST SP800-90A	7 PT
TLS-Erweiterung Encrypt-then-MAC	10 PT

2.2.10.3 NSS

Von den geforderten Algorithmen und Protokollen fehlen in NSS XTS, eine KDF nach NIST SP800-56C, TLS-DHE, TLS-PSK, ECIES, DLIES, ECKDSA, ECGDSA, Merkle-Signaturen, Brainpool-Kurven, eigene Kurven, GMAC und CMAC. Zusätzlich muss die TLS-Erweiterung Encrypt-then-MAC implementiert werden.

XTS	26 PT
KDF nach NIST SP800-56C	13 PT
TLS-DHE	33 PT
TLS-PSK	47 PT
ECIES	16 PT
DLIES	16 PT
ECKDSA	16 PT
ECGDSA	16 PT
Merkle-Signaturen	33 PT
Brainpool-Kurven	33 PT
Eigene Kurven GF(p)	33 PT
GMAC	5 PT
CMAC	15 PT
TLS-Erweiterung Encrypt-then-MAC	12 PT

2.2.11 Nicht gewünschte Algorithmen

Um die Nutzung von nicht geeigneten Algorithmen zu verhindern soll die Bibliothek auf die notwendigen Funktionalitäten und Schnittstellen beschränkt werden. Dies bedeutet beispielsweise die Entfernung von Algorithmen wie DES, Camellia, IDEA, SEED, DES, RC2, RC4, MD2, MD5 oder Protokollen wie SSLv2 und SSLv3. Der Aufwand dafür ist stark von der Modularisierung der Bibliothek abhängig. Mit steigender Modularität sinkt der Aufwand für die Entfernung der nicht gewünschten Algorithmen und Protokolle. Botan bietet nach der Analyse in 2.2.3 die beste Modularisierung, daher schätzen wir den Aufwand für die Entfernung nicht gewünschter Algorithmen mit zwei Personentagen ab. LibreSSL ist weniger modular, daher schätzen wir den Aufwand für die Entfernung nicht gewünschter Algorithmen mit fünf Personentagen ab. NSS ist am wenigsten modular und enthält zyklische Abhängigkeiten zwischen einzelnen Codeteilen, sodass wir hier den meisten Aufwand, nämlich 12 Personentage, für die Entfernung der nicht gewünschten Algorithmen und Protokolle erwarten.

2.2.12 Fehler in Zertifikatsvalidierung

Die Zertifikatskettenprüfung in X.509 ist nicht nur ein zentraler Bestandteil von TLS und daher kritisch für die Sicherheit der Bibliothek. Daher wird eine Analyse der Zertifikatskettenprüfung der Bibliotheken mittels des Tools *x509test* [x509test] in Version 02187cb59 durchgeführt. *x509test* ist ein in Python 3 geschriebenes Tool zum Testen der Zertifikatskettenprüfung eines SSL/TLS-Clients.

Um einen SSL/TLS-Client zu testen, stellt *x509test* einen TLS-Server bereit, zu dem sich ein SSL/TLS-Client verbindet. Als TLS-Server kommt OpenSSL zum Einsatz. Zum Starten des TLS-Servers erwartet *x509test* zunächst einen Domainnamen (Fully Qualified Domain Name – FQDN) und ein Root CA Zertifikat. Wird kein Root CA Zertifikat angegeben, generiert *x509test* ein self-signed Zertifikat das als Root CA Zertifikat zum Einsatz kommt. Im nächsten Schritt generiert *x509test* eine Menge von Test-Zertifikaten. Einige der Zertifikate sind direkt von der Root CA signiert, andere von zusätzlich erzeugten Intermediate CAs. Die meisten der erzeugten Test-Zertifikate enthalten Fehler.

x509test wartet dann auf den Verbindungsaufbau des zu testenden SSL/TLS-Clients. In jeder Session mit dem Client wird eine andere Zertifikatskette präsentiert. Akzeptiert der Client eine fehlerhafte Zertifikatskette im Handshake oder bricht der Handshake bei einer gültigen Zertifikatskette ab, so wird dies als potentieller Fehler des Clients aufgezeichnet. Enthalten sind Positiv- wie Negativtests zu Basic Certificate Fields wie Name und Validity, Certificate Extensions wie Name Constraints, KeyUsage und ExtendedKeyUsage, zur Hostname Verification, auch mittels Wildcards, und zum Handling von Self-signed Zertifikaten. Die vorliegende Version von *x509test* enthält 45 Tests. Die detaillierten Ausgaben der *x509test*-Durchläufe finden sich im Anhang in Abschnitt 3.3.

2.2.12.1 Botan

Im Test mit dem Botan TLS-Testclient schlugen 7 der 45 Tests fehl. Weitere 4 Tests Negativtests wurden nicht ausgeführt, da bereits der zugehörige Positivtest fehl schlug. Die fehlgeschlagenen Tests betreffen die Bereiche Name Constraints, Key Usage und Hostname Verification nach RFC 6125. Den Aufwand zur Behebung der fehlgeschlagenen Tests schätzen wir bei zwei Personentagen pro fehlgeschlagenem Test mit 14 Personentagen ab.

Name	Typ	Referenz	Verhalten Botan	Schwere
ValidNameConstraint	Positiv	RFC 5280, 4.2.1.10	Abgelehnt	k.A.
InvalidNameConstraint Exclude	Negativ	RFC 5280, 4.2.1.10	Nicht ausgeführt	Mittel
InvalidNameConstraint PermitRight	Negativ	RFC 5280, 4.2.1.10	Nicht ausgeführt	Mittel
InvalidNameConstraint PermitThenExclude	Negativ	RFC 5280, 4.2.1.10	Nicht ausgeführt	Mittel
InvalidNameConstraint Permit	Negativ	RFC 5280, 4.2.1.10	Nicht ausgeführt	Mittel
InvalidExtendedKeyUsage	Negativ	RFC 5280, 4.2.1.12	Akzeptiert	Niedrig
InvalidKeyUsage	Negativ	RFC 5280, 4.2.1.3	Akzeptiert	Niedrig
InvalidWildcardLeft	Negativ	RFC 6125, 6.4.3	Akzeptiert	Mittel
InvalidWildcardLeftAltName	Negativ	RFC 6125, 6.4.3	Akzeptiert	Mittel
InvalidNameAltName WithSubj	Negativ	RFC 6125, 6.4.4	Akzeptiert	Niedrig
InvalidNameNullAltName WithSubj	Negativ	RFC 6125, 6.4.4	Akzeptiert	Niedrig

Tabelle 2.2: Fehlgeschlagene bzw. nicht ausgeführte Tests mit dem Botan TLS-Testclient

2.2.12.2 LibreSSL

Im Test mit dem LibreSSL TLS-Testclient schlugen 19 der 45 durchgeführten Tests fehl. Vier Negativtests wurden nicht ausgeführt, da bereits der zugehörige Positivtest fehlschlug. Dies entspricht exakt dem Testergebnis von OpenSSL, was im Rahmen der OpenSSL-Studie angefertigt wurde. Demnach betreffen 17 der fehlgeschlagenen Tests die Hostname Verification nach RFC 6125. Auch LibreSSL führt derzeit keine Hostname Verification durch. Die übrigen zwei fehlgeschlagenen Tests betreffen die Prüfungen der Certificate Extensions und der Name Constraints. Den Aufwand zur Behebung der fehlgeschlagenen Tests schätzen wir bei zwei Personentagen pro fehlgeschlagenem Test mit 38 Personentagen ab.

Name	Typ	Referenz	Verhalten OpenSSL	Schwere
ValidNameConstraint	Positiv	RFC 5280, 4.2.1.10	Abgelehnt: permitted subtree violation	k.A.
InvalidNameConstraint Exclude	Negativ	RFC 5280, 4.2.1.10	Nicht ausgeführt	Mittel
InvalidNameConstraint PermitRight	Negativ	RFC 5280, 4.2.1.10	Nicht ausgeführt	Mittel
InvalidNameConstraint PermitThenExclude	Negativ	RFC 5280, 4.2.1.10	Nicht ausgeführt	Mittel
InvalidNameConstraint Permit	Negativ	RFC 5280, 4.2.1.10	Nicht ausgeführt	Mittel
InvalidNameNull	Negativ	RFC 6125, 6.4	Akzeptiert	Mittel
InvalidName	Negativ	RFC 6125, 6.4	Akzeptiert	Mittel
InvalidNameAltName	Negativ	RFC 6125, 6.4	Akzeptiert	Mittel
MissingIntCABasicConstraintWithCertSign	Negativ	RFC 5280, 4.2.1.3	Akzeptiert	Hoch
InvalidWildcardLeft	Negativ	RFC 6125, 6.4.3	Akzeptiert	Mittel
InvalidWildcardMid	Negativ	RFC 6125, 6.4.3	Akzeptiert	Mittel
InvalidWildcardSingle	Negativ	RFC 6125, 6.4.3	Akzeptiert	Mittel
InvalidWildcardMidMixed	Negativ	RFC 6125, 6.4.3	Akzeptiert	Mittel
InvalidWildcardAll	Negativ	RFC 6125, 6.4.3	Akzeptiert	Mittel
InvalidWildcardLeftAltName	Negativ	RFC 6125, 6.4.3	Akzeptiert	Mittel
InvalidWildcardMidAltName	Negativ	RFC 6125, 6.4.3	Akzeptiert	Mittel
InvalidWildcardSingleAltName	Negativ	RFC 6125, 6.4.3	Akzeptiert	Mittel
InvalidWildcardMidMixedAltName	Negativ	RFC 6125, 6.4.3	Akzeptiert	Mittel
InvalidWildcardAllAltName	Negativ	RFC 6125, 6.4.3	Akzeptiert	Mittel
InvalidNameNullAltName	Negativ	RFC 6125, 6.4.4	Akzeptiert	Mittel
InvalidNameNullAltNameAndSubj	Negativ	RFC 6125, 6.4.4	Akzeptiert	Mittel
InvalidNameAltNameWithSubj	Negativ	RFC 6125, 6.4.4	Akzeptiert	Niedrig
InvalidNameNullAltNameWithSubj	Negativ	RFC 6125, 6.4.4	Akzeptiert	Niedrig

Tabelle 2.3: Fehlgeschlagene bzw. nicht ausgeführte Tests mit dem LibreSSL TLS-Client

2.2.12.3 NSS

Im Test mit dem NSS TLS-Testclient wurden alle 45 Tests erfolgreich ausgeführt.

2.2.13 Fehler in TLS-Protokollvalidierung

Im letzten Jahr wurden mehrere Schwachstellen in den TLS-Protokoll-Workflows berichtet. Die wissenschaftlichen Artikel haben aber nicht die untersuchten Bibliotheken analysiert (bzw. haben ältere Versionen dieser Bibliotheken untersucht).

In unserer Evaluierung haben wir die TLS Server-Implementierungen und ihre State Machines untersucht. Dafür wurde das Tool von Joeri de Ruiter verwendet (Joeri de Ruiter, Erik Poll: Protocol State Fuzzing of TLS Implementations). Die kompletten Resultate sind im Anhang in Abschnitt 3.4 zu finden.

Die Analyse hat gezeigt, dass der Botan-Server die einfachste State Machine hat. Sie beinhaltet nur 6 States. Die Übergänge sind sehr sauber implementiert und bei einer invaliden Nachricht antwortet der Server immer mit einem Alert. Zu einem validen Austausch von ApplicationData-Nachrichten führt nur ein Pfad.

Die untersuchte LibreSSL-Bibliothek beinhaltet die meisten Zustände (8). Jedoch ist der Übergang in den finalen Zustand, wo die Daten ausgetauscht werden, nur über einen Pfad möglich.

Das Modell von NSS ist sehr ähnlich zu dem Modell im wissenschaftlichen Artikel (NSS 3.17.4). Zu einem validen Austausch von Daten führt hier auch nur ein Pfad. Es ist interessant, dass man aus dem State 0 in den State 3 mit einer ApplicationData- oder ChangeCipherSpec-Nachricht kommen kann. Wenn man in diesem Zustand weiter ist, antwortet der Server auf keine der gesendeten Nachrichten. Die Verbindung wird aber aufrecht erhalten.

Die Analyse zeigt, dass die TLS State Machines korrekt implementiert sind, jedoch einige Unterschiede zeigen. Es ist interessant, dass jeder Server in einigen Zuständen leere ApplicationData-Nachrichten akzeptiert hat.

Eine Quelltextanalyse der TLS-Clients und ggf. Fixes der State Machines sollten aber berücksichtigt werden. Diese schätzen wir in allen Bibliotheken mit jeweils 10 Personentagen ab. Darüber hinaus empfehlen wir eine TLS-Fuzzing-Analyse auszuführen, in welcher zusätzlich zu den State Machines Bearbeitung invalider Nachrichten analysiert wird. Wir schätzen den Aufwand auf 20 Personentage ab, womit sich insgesamt 30 Personentage ergeben.

2.2.14 Statische Analyse

Mittels der statischen Analyse soll ein Erfahrungswert für die allgemeine Codequalität gewonnen werden. Dazu werden generierte Compiler-Warnungen, die Ausgabe des Tools Cppcheck und die Ergebnisse von Clang-Analyzer untersucht.

2.2.14.1 Warnungen

Kompilerfehler deuten auf Fehler im Code hin, während Compilerwarnungen auf mögliche Fehler hinweisen die erst zur Laufzeit auftreten und zu unerwartetem Verhalten führen können. Das Ziel ist

es daher, den Code ohne Compilerwarnungen übersetzen zu können. Bei der Auswahl der Compilerflags wurde sich an C++ Best Practices [CPPBESTPR] orientiert.

Die drei Bibliotheken wurden unter Linux x64 mit GCC 5.2.0 und den Flags "-Wall -Wextra -Wshadow -Wpedantic -Wunreachable-code" kompiliert. Bei der Kompilierung von Botan wurden 605 Warnungen generiert, bei LibreSSL 541 und bei NSS 4715 Warnungen.

Bei Botan setzen sich die 605 Warnungen aus den Typen Wpedantic (258x) und Wshadow (347x) zusammen. Der überwiegende Teil der Wpedantic Warnungen wurde durch ein überflüssiges Semikolon am Ende einer Deklaration erzeugt.

Diese Warnungen sind nicht kritisch und einfach zu beheben. Die Wshadow-Warnungen sollten um Programmierfehlern vorzubeugen behoben werden, sind jedoch auch als nicht kritisch einzustufen. Auf 118 Zeilen Code kommt eine Warnung bei Botan (insgesamt 71675 Zeilen Code).

Die 541 LibreSSL Warnungen setzen sich wie folgt zusammen:

- Wunused-parameter (418x)
- Wpedantic (73x)
- Wmissing-field-initializers (2x)
- Wsign-compare (44x)
- Wformat= (2x)
- Wempty-body (1x)
- Wshadow (1x)

Die 418x Wunused-parameter Warnungen sind nicht kritisch, jedoch ungünstig für die Lesbarkeit des Codes. Vermutlich wurden sie aus Gründen der API-Kompatibilität beibehalten. Die Wpedantic Warnungen wurden erzeugt, da sich an einigen Stellen nicht an die ISO C-Standard gehalten wurde. Dies kann zu Problemen mit der Portabilität des Codes führen. Die restlichen Warnungen sind schwerwiegender und müssen behoben werden. Bei LibreSSL wird alle 489 Zeilen Code eine Warnung erzeugt (insgesamt 264926 Zeilen Code).

Bei NSS wurden die folgenden 4715 Warnungen generiert:

- Wunused-parameter (1739x)
- Wmissing-field-initializers (1036x)
- Wshadow (636x)
- Wpedantic (605x)
- Wsign-compare (291x)
- Wunused-but-set-variable (156x)

- Wswitch (56x)
- Wmaybe-uninitialized (45x)
- Wstrict-aliasing (39x)
- Wpointer-to-int-cast (26x)
- Wtype-limits (18x)
- Wincompatible-pointer-types (9x)
- Wunused-function (8x)
- Wempty-body (7x)
- Woverflow (5x)
- Wold-style-declaration (4x)
- Wformat (4x)
- Wunused-value (4x)
- Wunused-variable (2x)
- Wdiscarded-qualifiers (2x)
- Wparentheses (2x)
- Wunused-but-set-parameter (1x)
- Wint-to-pointer-cast (1x)

Hier sind nicht nur die meisten, sondern auch die schwerwiegendsten Warnungen zu finden. Alle 142 Zeilen Code wird eine Warnung in NSS erzeugt (insgesamt 671744 Zeilen Code).

Bei 5 Minuten pro Warnung schätzen wir den Aufwand mit 7 Personentagen für Botan ab. Die von LibreSSL sind etwas aufwändiger zu beheben, da gerade beim Beheben von Wsign-compare Vor-sicht geboten ist. Da es jedoch insgesamt weniger Warnungen als bei Botan sind schätzen wir den Aufwand mit 6 Personentagen ab. Den mit Abstand meisten Aufwand bereitet NSS mit 50 Personentagen.

2.2.14.2 Cppcheck

Cppcheck hat bei Botan einen Fehler, 44 Stilwarnungen und eine Performance-Warnung erzeugt. LibreSSL enthält 20 Fehler, 34 Warnungen, 595 Stilwarnungen, 6 Portabilitätswarnungen und 4 Performance-Warnungen. Bei NSS hat Cppcheck 85 Fehler, 544 Warnungen, 2162 Stilwarnungen, 75 Portabilitätswarnungen und 219 Performance-Warnungen gefunden. Unter allen drei Analysen können sich False-Positives befinden.

Hier zeigt sich ein ähnliches Bild wie bei der Analyse der Compiler-Warnungen. Das beste Ergebnis erzielt Botan vor LibreSSL und NSS. Den Aufwand zur Behebung schätzen wir bei 5 Minuten pro

Fix mit jeweils einem, sieben und 33 Personentagen ab.

2.2.14.3 Clang-Analyzer

Mit Clang-Analyzer wurden ebenfalls alle drei Bibliotheken analysiert. In Botan wurden 5 Fehler gefunden, in LibreSSL wurden 119 Fehler gefunden. Darunter 90 "tote Zuweisungen", die mit geringem Aufwand zu beheben sind. 631 Fehler wurden in NSS gefunden, darunter 311 "tote Zuweisungen", aber auch 184 Nullpointer-Dereferenzierungen sowie weitere Fehler wie beispielsweise Memory-Leaks. Unter allen drei Analysen können sich hier ebenfalls False-Positives befinden.

Auch hier platziert sich Botan vor LibreSSL und NSS. Den Aufwand zur Behebung schätzen wir bei 15 Minuten pro Fix mit jeweils einem, vier und 20 Personentagen ab.

2.2.15 Continuous Integration

Continuous Integration (CI) bezeichnet den Prozess des regelmäßigen Zusammenfügens von Komponenten zu einer Anwendung zum Zweck der Erhöhung der Softwarequalität (Quelle: Wikipedia). In der modernen Ausprägung beinhaltet dies meist die automatisierte Kompilation auf einem oder mehreren Referenzsystemen, die Ausführung der Testsuite, die Analyse der Testabdeckung und auch eine statische Analyse des Codes, idealerweise bei jeder Änderung. All dies ermöglicht es, Fehler möglichst früh in der Entwicklungsphase zu finden und zu beheben und damit die Qualität sicherzustellen.

2.2.15.1 Botan

Bei Botan kommt bereits Continuous Integration zum Einsatz. Bei einem neuen Commit werden über Travis-CI 12 verschiedene Build-Jobs gestartet. Dabei erfolgt die Kompilierung unter Linux mit Clang sowie GCC und unter OS X sowie iOS mit Clang. Jeder dieser Kompilierungsvorgänge erfolgt einmal zur Generierung einer statischen und einmal zur Generierung einer dynamischen Bibliothek. Zwei weitere Jobs existieren zum Bestimmen der Code-Coverage und einen zur dynamischen Analyse mittels Address-Sanitizer. Bei jedem dieser Jobs werden außerdem die Tests ausgeführt. Die automatisierte Kompilierung für Windows erfolgt auf der Plattform Appveyor mittels Visual C++ 2013. Dort existiert ein Job für x86 und einer für x64. Die Testsuite wird ebenfalls bei beiden Jobs ausgeführt. Eine statische Analyse erfolgt mittels Coverity. Es existieren weitere Build-Jobs auf den Plattformen "Circle CI" und "Kullo CI". Den Aufwand, diese gesamte Umgebung während der Entwicklung des Projektes zur Verfügung zu stellen schätzen mit vier Personentagen ab.

2.2.15.2 LibreSSL

Für LibreSSL gibt es auf der Travis-CI Plattform ebenfalls Build-Jobs für die Konfigurationen: OS X mit Clang, OS X mit GCC, Linux mit Clang, Linux mit GCC sowie zwei Cross-Kompilier-MinGW Jobs für x86 und X64. Bis auf den beiden Cross-Kompilier-Jobs wird die Testsuite auf allen Jobs ausgeführt. Auf Windows folglich jedoch nicht. Die Code Coverage kann über ein Shellskript überprüft werden (gen-coverage-report.sh). Dies erfolgt jedoch nicht automatisch. Eine automatisierte statische oder dynamische Analyse kommt ebenfalls nicht zum Einsatz. Den Aufwand, diese gesamte Umgebung während der Entwicklung des Projektes zur Verfügung zu stellen und um automatisiertes Test Coverage Reporting und eine automatisierte statische Analyse zu ergänzen schätzen

mit sechs Personentagen ab.

2.2.15.3 NSS

Das Projekt nutzt die Mozilla-eigene Treeherder CI Infrastruktur, die auf Buildbot basiert. Sie umfasst nur die automatisierte Kompilation und Tests. Den Aufwand, diese gesamte Umgebung während der Entwicklung des Projektes zur Verfügung zu stellen und um automatisiertes Test Coverage Reporting und eine automatisierte statische Analyse zu ergänzen schätzen mit 10 Personentagen ab.

2.2.16 Kompatibilität mit der OpenSSL-Schnittstelle

Im Rahmen der durchgeführten Herstellerabfrage wurde von mehreren VS-Herstellern der Wunsch einer OpenSSL-/LibreSSL-kompatiblen Schnittstelle geäußert. Einzig LibreSSL bietet dies. Der genaue Aufwand für die anderen Bibliotheken ist unserer Ansicht nach schwer abzuschätzen. Desweiteren halten wir eine Abbildung der OpenSSL-API auf Botan oder NSS für nicht sinnvoll. Die OpenSSL-API ist ein häufig genannter Anlass für Kritik an OpenSSL, da sie nicht sehr anwenderfreundlich ist. Wir schlagen stattdessen für Botan und NSS vor, für häufig genutzte Funktionen ein entsprechendes Kapitel im Benutzerhandbuch zu ergänzen, das den Umstieg von der OpenSSL-API auf die API von Botan oder NSS erleichtert. Für beide Bibliotheken schätzen wir dafür jeweils 3 Personentage ab.

2.3 Auswertung

In Botan sollten Verbesserungen beim RSA-Blinding, bei der Resistenz gegen Bleichenbacher-Angriffe, bei der Resistenz gegen Small Subgroup-Angriffe, und bei der Resistenz gegen Padding Oracle-Angriffe auf CBC implementiert werden. Invalid Curve Angriffe sind hingegen nicht möglich. Die Unterstützung von Linux und Windows ist gegeben. iOS und Android werden grundsätzlich auch unterstützt, jedoch ist der Status der Testsuite auf beiden Plattformen unklar. Botan ist sehr gut modularisiert. Bei der Kompilation kann man einzelne der über 50 Module deaktivieren, diese sind dann nicht Teil des Kompilats. Eine Analyse mit dem Tool cppdepend hat 19 als komplex eingestufte Funktionen ergeben. Botan implementiert einen NIST SP 800-90A kompatiblen Zufallsgenerator, der im Falle des fork(2) eines Prozess neu initialisiert werden sollte. Die Unterstützung von PKCS#11 und PKCS#12 muss ergänzt werden. Die Testsuite deckt bereits ca. 75% des Codes mit Tests ab. Eine Doxygen-kompatible Dokumentation ist vorhanden, aber nicht vollständig. Auch ein Handbuch steht zur Verfügung, sollte aber aktualisiert und um weitere Themen ergänzt werden. Der Mechanismus zum sicheren Löschen von Geheimnissen sollte um das Locken in den RAM unter Windows ergänzt werden. Da Botan sehr modular aufgebaut ist, ist der Aufwand zur Entfernung der nicht gewünschten Algorithmen gering. Die 7 mit x509test gefundenen Fehler in der Zertifikatskettenprüfung des TLS-Handshakes sollten, genau wie die Compilerwarnungen, die Meldungen von cppcheck und des clang-analyzer, behoben werden. Eine Continuous Integration Infrastruktur ist vorhanden. Insgesamt schätzen wir den Gesamtaufwand zum Erreichen der gesteckten Ziele mit der Botan-Bibliothek mit 309 Personentagen ab.

In LibreSSL sollten Verbesserungen bei der Resistenz gegen Bleichenbacher-Angriffe implementiert werden. Invalid-Curve Angriffe, Small Subgroup-Angriffe und Padding Oracle-Angriffe auf CBC sind hingegen nicht möglich. Das RSA-Blinding ist korrekt implementiert. iOS und Android

werden offiziell nicht unterstützt, jedoch werden die verwandten Plattformen OS X und Linux unterstützt. LibreSSL ist weniger gut modularisiert und bietet nur wenige Compilerflags an. Es ist unklar, ob das Deaktivieren von bestimmten Funktionen überhaupt möglich ist. Eine Analyse mit dem Tool `cppdepend` hat 69 als komplex eingestufte Funktionen ergeben. LibreSSL implementiert keinen NIST SP 800-90A kompatiblen Zufallsgenerator. Dieser müsste komplett neu implementiert werden. Die Unterstützung von PKCS#11 muss ergänzt werden. Die Testsuite deckt lediglich ca. 30% des Codes mit Tests ab. Eine Doxygen-kompatible Dokumentation ist nicht vorhanden. Ein Handbuch steht nicht zur Verfügung und müsste komplett neu geschrieben werden. Der Mechanismus zum sicheren Löschen von Geheimnissen sollte um das Locken in den RAM auf allen Plattformen ergänzt werden. Da LibreSSL weniger modular aufgebaut ist, ist der Aufwand zur Entfernung der nicht gewünschten Algorithmen höher als bei Botan. Die 19 mit `x509test` gefundenen Fehler in der Zertifikatskettenprüfung des TLS-Handshakes sollten, genau wie die Compilerwarnungen, die Meldungen von `cppcheck` und des `clang-analyzer`, behoben werden. LibreSSL erzeugt etwas weniger Compilerwarnungen als Botan, dafür aber um den Faktor 14 mehr Meldungen bei `cppcheck` und um den Faktor 24 mehr Meldungen bei `clang-analyzer`. Eine Continuous Integration Infrastruktur zur Kompilation und automatisierten Ausführung der Testsuite ist vorhanden, ein automatisiertes Test Coverage Reporting und statische Analyse sollte aber noch ergänzt werden. Insgesamt schätzen wir den Gesamtaufwand zum Erreichen der gesteckten Ziele mit der LibreSSL-Bibliothek mit 504 Personentagen ab.

In NSS sollten wie bei LibreSSL Verbesserungen bei der Resistenz gegen Bleichenbacher-Angriffe implementiert werden. Invalid-Curve Angriffe, Small Subgroup-Angriffe und Padding Oracle-Angriffe auf CBC sind wie bei LibreSSL nicht möglich. Auch das RSA-Blinding ist korrekt implementiert. iOS wird offiziell nicht unterstützt, jedoch wird mit OS X eine stark verwandte Plattform unterstützt. Android wird offiziell unterstützt. NSS ist am wenigsten gut modularisiert und bietet keine Konfiguration der enthaltenen Funktionen über Compilerflags an. Die einzelnen Module haben untereinander viele Abhängigkeiten. Daher ist unklar, ob das Deaktivieren von bestimmten Funktionen überhaupt möglich ist. Eine Analyse mit dem Tool `cppdepend` hat 175 als komplex eingestufte Funktionen ergeben. NSS implementiert einen NIST SP 800-90A kompatiblen Zufallsgenerator. Auch die Neuinitialisierung bei `fork(2)` muss ergänzt werden. PKCS#11 wird von NSS unterstützt. Die Testsuite deckt ca. 50% des Codes mit Tests ab. Die API-Dokumentation ist nahezu vollständig, jedoch nicht Doxygen-kompatibel. Ein eingeschränktes Handbuch steht zur Verfügung, sollte aber von toten Verweisen bereinigt, aktualisiert und um weitere Themen ergänzt werden. Der Mechanismus zum sicheren Löschen von Geheimnissen sollte um das Sperren des Auslagerns auf die Festplatte auf allen Plattformen ergänzt werden. Da NSS am wenigsten modular aufgebaut ist, ist der Aufwand zur Entfernung der nicht gewünschten Algorithmen im Vergleich am höchsten. Im Test mit `x509test` traten im Gegensatz zu Botan und LibreSSL keine Fehler auf. NSS erzeugt wesentlich mehr Meldungen bei der statischen Analyse, nämlich um den Faktor 7 mehr Compilerwarnungen als Botan und LibreSSL, um den Faktor 65 mehr Meldungen bei `cppcheck` als Botan und um den Faktor 126 mehr Meldungen bei `clang-analyzer` als Botan. Eine Continuous Integration Infrastruktur zur Kompilation und automatisierten Ausführung der Testsuite ist zwar vorhanden, jedoch sehr komplex. Daher schätzen wir den Aufwand zur Bereitstellung der Infrastruktur im Rahmen des Projektes etwas höher ein als bei Botan und LibreSSL. In jedem Fall müsste die CI noch

um ein automatisiertes Test Coverage Reporting und eine automatisierte statische Analyse ergänzt werden. Insgesamt schätzen wir den Gesamtaufwand zum Erreichen der gesteckten Ziele mit der NSS-Bibliothek mit 880 Personentagen ab.

Unserer Einschätzung nach erreicht man das Ziel einer sicheren und allgemeinen Kryptobibliothek mit dem geringsten Aufwand mit der Botan-Bibliothek. Botan bietet im Vergleich zu LibreSSL und NSS eine gute Modularisierung bei geringer Komplexität und guter Lesbarkeit des Quelltextes, einen hohen Grad an Dokumentation und die mit Abstand beste Testabdeckung. Die statische Analyse hat zudem gezeigt, dass die allgemeine Codequalität als gut zu bewerten ist. Dies liegt nicht zuletzt auch an der vorhandenen Continuous Integration Infrastruktur. Vom geforderten Funktionsumfang her ist Botan mit LibreSSL und NSS vergleichbar. Davon ausgenommen ist lediglich die PKCS#11-Unterstützung, die nicht vorhanden ist.

3 Anhang: Messergebnisse

3.1 Komplexität

Für das Messen der Komplexität wurde u.a. mit Sloccount [sloccount] die Anzahl der Codezeilen berechnet. Die Messergebnisse sind hier im Detail dargestellt.

3.1.1 Botan

SLOC	Directory	SLOC-by-Language (Sorted)
62478	lib	cpp=61231,asm=1247
5225	tests	cpp=5225
3427	cmd	cpp=3427
940	scripts	python=851,sh=55,lisp=34
634	contrib	cpp=309,ansic=236,perl=89
442	python	python=442
173	ocaml	ml=172,sh=1
72	build-data	python=72

Totals grouped by language (dominant language first):

cpp:	70192 (95.64%)
python:	1365 (1.86%)
asm:	1247 (1.70%)
ansic:	236 (0.32%)
ml:	172 (0.23%)
perl:	89 (0.12%)
sh:	56 (0.08%)
lisp:	34 (0.05%)

3.1.2 LibreSSL

SLOC	Directory	SLOC-by-Language (Sorted)
172483	crypto	ansic=124523,asm=47960
28326	apps	ansic=28326
26028	ssl	ansic=26028

22022	include	ansic=22022
15147	tests	ansic=14889,sh=258
11325	top_dir	sh=11325
1178	tls	ansic=1178
41	scripts	sh=41
0	m4	(none)
0	man	(none)

Totals grouped by language (dominant language first):

ansic: 216966 (78.45%)

asm: 47960 (17.34%)

sh: 11624 (4.20%)

3.1.3 NSS

SLOC	Directory	SLOC-by-Language (Sorted)
------	-----------	---------------------------

578644	nss	ansic=479304,asm=48936,cpp=27833,sh=16200, python=3923,perl=2031,lex=306,yacc=79,ada=16,csh=10,sed=6
--------	-----	---

119643	nspr	ansic=112043,sh=3544,cpp=2802,asm=826,perl=404,python=24
--------	------	--

Totals grouped by language (dominant language first):

ansic: 591347 (84.69%)

asm: 49762 (7.13%)

cpp: 30635 (4.39%)

sh: 19744 (2.83%)

python: 3947 (0.57%)

perl: 2435 (0.35%)

lex: 306 (0.04%)

yacc: 79 (0.01%)

ada: 16 (0.00%)

csh: 10 (0.00%)

sed: 6 (0.00%)

3.2 Testabdeckung

Die Analyse der Testabdeckung wurde mit gcov [GCV] durchgeführt. Die Messergebnisse sind im folgenden dargestellt. Sie zeigen die Testabdeckung der jeweiligen Bibliothek in Prozent für Codezeilen (Lines) und Funktionen (Functions). Eine Messung der Testabdeckung für Verzweigungen (Branches) ist für C/C++ Code nicht verfügbar, daher werden hier die Werte 0 angezeigt.

3.2.1 Botan

Summary coverage rate:

lines.....: 77.2% (21282 of 27578 lines)

functions..: 74.5% (4237 of 5690 functions)

branches...: no data found

Filename	Lines		Functions		Branches	
	Rate	Num	Rate	Num	Rate	Num
=====						
[Botan-1.11.18/]						
build/include/botan/adler32.h	66.7%	6	71.4%	7	-	0
build/include/botan/aead.h	100%	9	85.7%	7	-	0
build/include/botan/aes.h	33.3%	9	60.0%	15	-	0
build/include/botan/aes_ni.h	83.3%	12	88.9%	18	-	0
build/include/botan/aes_ssse3.h	33.3%	9	60.0%	15	-	0
build/include/botan/alg_id.h	100%	2	66.7%	6	-	0
build/include/botan/asn1_alt_name.h	100%	1	66.7%	6	-	0
build/include/botan/asn1_attribute.h	100%	2	66.7%	3	-	0
build/include/botan/asn1_obj.h	60.0%	5	64.3%	14	-	0
build/include/botan/asn1_oid.h	100%	3	87.5%	8	-	0
build/include/botan/asn1_str.h	100%	1	80.0%	5	-	0
build/include/botan/asn1_time.h	100%	1	66.7%	6	-	0
build/include/botan/auto_rng.h	40.0%	10	33.3%	9	-	0
build/include/botan/b64_filt.h	50.0%	4	66.7%	6	-	0
build/include/botan/base64.h	100%	2	50.0%	2	-	0

build/include/botan/basefilt.h	50.0% 4 50.0% 6 - 0
build/include/botan/ber_dec.h	68.0% 50 45.5% 22 - 0
build/include/botan/bigint.h	94.7% 94 90.5% 42 - 0
build/include/botan/blinding.h	100% 2 100% 3 - 0
build/include/botan/block_cipher.h	100% 20 81.5% 92 - 0
build/include/botan/blowfish.h	66.7% 3 80.0% 5 - 0
build/include/botan/bswap.h	100% 12 100% 5 - 0
build/include/botan/buf_comp.h	100% 39 88.9% 18 - 0
build/include/botan/buf_filt.h	100% 1 50.0% 2 - 0
build/include/botan/calendar.h	100% 2 100% 1 - 0
build/include/botan/camellia.h	33.3% 9 60.0% 15 - 0
build/include/botan/cascade.h	100% 5 100% 4 - 0
build/include/botan/cast128.h	33.3% 3 60.0% 5 - 0
build/include/botan/cast256.h	33.3% 3 60.0% 5 - 0
build/include/botan/cbc.h	100% 17 94.4% 18 - 0
build/include/botan/cbc_mac.h	100% 4 100% 4 - 0
build/include/botan/ccm.h	63.2% 19 70.6% 17 - 0
build/include/botan/certstor.h	100% 9 80.0% 10 - 0
build/include/botan/cfb.h	100% 11 91.7% 12 - 0
build/include/botan/chacha.h	71.4% 7 71.4% 7 - 0
build/include/botan/chacha20poly1305.h	75.0% 16 70.6% 17 - 0
build/include/botan/cipher_mode.h	33.3% 3 40.0% 5 - 0
build/include/botan/cmac.h	100% 4 100% 4 - 0
build/include/botan/code_based_util.h	100% 15 100% 7 - 0
build/include/botan/comb4p.h	42.9% 7 60.0% 5 - 0
build/include/botan/comp_filter.h	0.0% 3 0.0% 6 - 0
build/include/botan/compression.h	0.0% 8 0.0% 6 - 0
build/include/botan/cpuid.h	54.5% 33 50.0% 16 - 0
build/include/botan/crc24.h	66.7% 6 71.4% 7 - 0
build/include/botan/crc32.h	66.7% 6 71.4% 7 - 0

build/include/botan/credentials_manager.h	100% 2 66.7% 3 - 0
build/include/botan/crl_ent.h	75.0% 4 62.5% 8 - 0
build/include/botan/ctr.h	71.4% 7 80.0% 5 - 0
build/include/botan/curve25519.h	62.5% 8 72.7% 11 - 0
build/include/botan/curve_gfp.h	100% 38 95.2% 21 - 0
build/include/botan/data_snk.h	0.0% 4 0.0% 5 - 0
build/include/botan/data_src.h	75.0% 12 63.6% 11 - 0
build/include/botan/datastor.h	100% 1 100% 5 - 0
build/include/botan/der_enc.h	100% 16 92.3% 13 - 0
build/include/botan/des.h	66.7% 6 80.0% 10 - 0
build/include/botan/desx.h	33.3% 3 60.0% 5 - 0
build/include/botan/dh.h	87.5% 8 90.0% 10 - 0
build/include/botan/dl_algo.h	90.0% 10 90.9% 11 - 0
build/include/botan/dl_group.h	100% 1 100% 3 - 0
build/include/botan/dlies.h	100% 2 50.0% 4 - 0
build/include/botan/dsa.h	90.9% 11 92.3% 13 - 0
build/include/botan/eax.h	58.8% 17 66.7% 15 - 0
build/include/botan/ec_group.h	100% 9 100% 10 - 0
build/include/botan/ecb.h	100% 9 90.0% 10 - 0
build/include/botan/ecc_key.h	100% 9 100% 7 - 0
build/include/botan/ecdh.h	70.6% 17 71.4% 14 - 0
build/include/botan/ecdsa.h	85.7% 14 92.3% 13 - 0
build/include/botan/elgamal.h	88.9% 9 90.9% 11 - 0
build/include/botan/eme.h	100% 2 66.7% 3 - 0
build/include/botan/eme_pkcs.h	100% 1 100% 3 - 0
build/include/botan/eme_raw.h	100% 2 100% 3 - 0
build/include/botan/emsa.h	100% 2 66.7% 3 - 0
build/include/botan/emsal.h	100% 3 100% 4 - 0
build/include/botan/emsal_bsi.h	100% 2 100% 3 - 0
build/include/botan/emsal_pkcs1.h	100% 2 100% 5 - 0

build/include/botan/emsa_raw.h	100% 1 100% 3 - 0
build/include/botan/emsa_x931.h	100% 1 100% 2 - 0
build/include/botan/entropy_src.h	100% 12 83.3% 12 - 0
build/include/botan/exceptn.h	24.1% 54 21.7% 46 - 0
build/include/botan/filter.h	77.8% 18 64.7% 17 - 0
build/include/botan/filters.h	34.8% 23 36.8% 19 - 0
build/include/botan/gcm.h	57.9% 19 68.4% 19 - 0
build/include/botan/get_byte.h	100% 2 100% 5 - 0
build/include/botan/gf2m_small_m.h	100% 41 100% 21 - 0
build/include/botan/gost_28147.h	40.0% 5 60.0% 5 - 0
build/include/botan/gost_3410.h	75.0% 16 78.6% 14 - 0
build/include/botan/gost_3411.h	60.0% 5 66.7% 6 - 0
build/include/botan/has160.h	66.7% 6 66.7% 6 - 0
build/include/botan/hash.h	50.0% 2 50.0% 4 - 0
build/include/botan/hex.h	100% 2 100% 2 - 0
build/include/botan/hex_filt.h	0.0% 4 0.0% 6 - 0
build/include/botan/hkdf.h	50.0% 4 60.0% 5 - 0
build/include/botan/hmac.h	100% 4 100% 4 - 0
build/include/botan/hmac_drbg.h	100% 1 100% 2 - 0
build/include/botan/hmac_rng.h	100% 1 100% 2 - 0
build/include/botan/http_util.h	0.0% 11 0.0% 8 - 0
build/include/botan/idea.h	60.0% 5 71.4% 7 - 0
build/include/botan/idea_sse2.h	66.7% 3 80.0% 5 - 0
build/include/botan/if_algo.h	100% 14 100% 14 - 0
build/include/botan/internal/algo_registry.h	93.3% 75 99.6% 254 - 0
build/include/botan/internal/binary_matrix.h	100% 9 100% 4 - 0
build/include/botan/internal/bit_ops.h	87.1% 31 90.0% 10 - 0
build/include/botan/internal/curve_nistp.h	100% 33 97.1% 35 - 0
build/include/botan/internal/def_powm.h	66.7% 6 75.0% 8 - 0
build/include/botan/internal/dev_random.h	0.0% 1 0.0% 1 - 0

build/include/botan/internal/donna128.h	100%	4 100%	2 - 0
build/include/botan/internal/es_egd.h	50.0%	2 66.7%	3 - 0
build/include/botan/internal/hres_timer.h	50.0%	2 75.0%	4 - 0
build/include/botan/internal/mode_utils.h	80.0%	15 100%	14 - 0
build/include/botan/internal/mp_asmi.h	100%	36 100%	12 - 0
build/include/botan/internal/mp_madd.h	100%	6 100%	2 - 0
build/include/botan/internal/pk_ops_impl.h	0.0%	4 0.0%	2 - 0
build/include/botan/internal/pk_utils.h	75.0%	4 94.7%	19 - 0
build/include/botan/internal/proc_walk.h	50.0%	6 42.9%	7 - 0
build/include/botan/internal/rdrand.h	50.0%	2 75.0%	4 - 0
build/include/botan/internal/rounding.h	50.0%	10 50.0%	2 - 0
build/include/botan/internal/semaphore.h	0.0%	2 0.0%	2 - 0
build/include/botan/internal/simd_sse2.h	100%	65 100%	23 - 0
build/include/botan/internal/stl_util.h	77.3%	22 88.9%	9 - 0
build/include/botan/internal/tls_extensions.h	80.2%	101 76.8%	95 - 0
build/include/botan/in...l/tls_handshake_hash.h	60.0%	5 60.0%	5 - 0
build/include/botan/internal/tls_handshake_io.h	92.9%	14 88.2%	17 - 0
build/include/botan/in.../tls_handshake_state.h	90.6%	32 94.4%	18 - 0
build/include/botan/internal/tls_heartbeats.h	0.0%	3 0.0%	3 - 0
build/include/botan/internal/tls_messages.h	70.5%	129 79.3%	82 - 0
build/include/botan/internal/tls_reader.h	75.3%	93 73.3%	30 - 0
build/include/botan/internal/tls_record.h	100%	12 100%	11 - 0
build/include/botan/internal/tls_seq_numbers.h	87.0%	46 88.0%	25 - 0
build/include/botan/internal/tls_session_key.h	100%	9 100%	10 - 0
build/include/botan/internal/unix_procs.h	16.7%	12 45.5%	11 - 0
build/include/botan/internal/xor_buf.h	96.2%	26 100%	5 - 0
build/include/botan/kasumi.h	33.3%	3 60.0%	5 - 0
build/include/botan/kdf.h	100%	13 88.9%	9 - 0
build/include/botan/kdf1.h	50.0%	4 60.0%	5 - 0
build/include/botan/kdf2.h	50.0%	4 60.0%	5 - 0

build/include/botan/keccak.h	66.7% 3 75.0% 4 - 0
build/include/botan/key_filt.h	33.3% 3 50.0% 4 - 0
build/include/botan/key_spec.h	100% 14 100% 5 - 0
build/include/botan/lion.h	100% 6 100% 6 - 0
build/include/botan/loadstor.h	98.6% 138 100% 41 - 0
build/include/botan/lookup.h	100% 2 100% 1 - 0
build/include/botan/mac.h	100% 1 66.7% 3 - 0
build/include/botan/mars.h	33.3% 3 60.0% 5 - 0
build/include/botan/mce_kem.h	100% 2 100% 2 - 0
build/include/botan/mceliece.h	88.6% 44 73.3% 15 - 0
build/include/botan/mceliece_key.h	73.1% 26 76.0% 25 - 0
build/include/botan/md2.h	85.7% 7 85.7% 7 - 0
build/include/botan/md4.h	83.3% 6 83.3% 6 - 0
build/include/botan/md5.h	83.3% 6 83.3% 6 - 0
build/include/botan/mdx_hash.h	100% 2 66.7% 3 - 0
build/include/botan/mem_ops.h	100% 14 92.9% 14 - 0
build/include/botan/misty1.h	33.3% 3 60.0% 5 - 0
build/include/botan/mode_pad.h	62.5% 16 64.0% 25 - 0
build/include/botan/noekeon.h	60.0% 5 71.4% 7 - 0
build/include/botan/noekeon_simd.h	66.7% 3 80.0% 5 - 0
build/include/botan/nr.h	62.5% 8 75.0% 12 - 0
build/include/botan/oaep.h	100% 1 100% 2 - 0
build/include/botan/ocb.h	53.3% 15 66.7% 12 - 0
build/include/botan/ocsp.h	0.0% 5 0.0% 4 - 0
build/include/botan/ocsp_types.h	0.0% 7 0.0% 12 - 0
build/include/botan/ofb.h	71.4% 7 80.0% 5 - 0
build/include/botan/par_hash.h	100% 2 100% 3 - 0
build/include/botan/pbkdf.h	100% 6 80.0% 5 - 0
build/include/botan/pbkdf1.h	33.3% 6 60.0% 5 - 0
build/include/botan/pbkdf2.h	33.3% 6 60.0% 5 - 0

build/include/botan/pem.h	100% 4 100% 2 - 0
build/include/botan/pipe.h	37.5% 8 40.0% 5 - 0
build/include/botan/pk_keys.h	88.9% 9 85.7% 14 - 0
build/include/botan/pk_ops.h	81.0% 21 67.7% 31 - 0
build/include/botan/pkcs10.h	100% 1 50.0% 2 - 0
build/include/botan/pkcs8.h	0.0% 3 0.0% 3 - 0
build/include/botan/point_gfp.h	94.2% 52 84.6% 26 - 0
build/include/botan/poly1305.h	66.7% 6 71.4% 7 - 0
build/include/botan/polyn_gf2m.h	100% 26 100% 14 - 0
build/include/botan/pow_mod.h	100% 10 83.3% 18 - 0
build/include/botan/prf_tls.h	42.9% 7 55.6% 9 - 0
build/include/botan/prf_x942.h	33.3% 3 50.0% 4 - 0
build/include/botan/pssr.h	100% 1 100% 2 - 0
build/include/botan/pubkey.h	88.9% 36 75.0% 28 - 0
build/include/botan/rc2.h	66.7% 3 80.0% 5 - 0
build/include/botan/rc4.h	75.0% 4 75.0% 4 - 0
build/include/botan/rc5.h	100% 2 100% 3 - 0
build/include/botan/rc6.h	66.7% 3 80.0% 5 - 0
build/include/botan/reducer.h	100% 8 87.5% 8 - 0
build/include/botan/rfc6979.h	100% 1 100% 1 - 0
build/include/botan/rmd128.h	83.3% 6 83.3% 6 - 0
build/include/botan/rmd160.h	83.3% 6 83.3% 6 - 0
build/include/botan/rng.h	62.5% 16 42.9% 14 - 0
build/include/botan/rotate.h	100% 8 100% 7 - 0
build/include/botan/rsa.h	100% 14 100% 12 - 0
build/include/botan/rw.h	100% 14 100% 12 - 0
build/include/botan/safer_sk.h	100% 1 100% 2 - 0
build/include/botan/salsa20.h	83.3% 6 83.3% 6 - 0
build/include/botan/scan_name.h	83.3% 6 80.0% 5 - 0
build/include/botan/secmem.h	97.0% 66 90.5% 74 - 0

build/include/botan/secqueue.h	33.3% 3 50.0% 4 - 0
build/include/botan/seed.h	33.3% 3 60.0% 5 - 0
build/include/botan/serp_simd.h	100% 3 100% 5 - 0
build/include/botan/serpent.h	60.0% 5 66.7% 6 - 0
build/include/botan/sha160.h	90.9% 11 85.7% 7 - 0
build/include/botan/sha1_sse2.h	66.7% 3 75.0% 4 - 0
build/include/botan/sha1_x86_64.h	50.0% 2 75.0% 4 - 0
build/include/botan/sha2_32.h	83.3% 12 83.3% 12 - 0
build/include/botan/sha2_64.h	82.4% 17 83.3% 18 - 0
build/include/botan/siphhash.h	100% 5 100% 5 - 0
build/include/botan/siv.h	61.1% 18 68.8% 16 - 0
build/include/botan/skein_512.h	66.7% 3 75.0% 4 - 0
build/include/botan/srp6.h	100% 1 100% 2 - 0
build/include/botan/srp6_files.h	0.0% 4 0.0% 4 - 0
build/include/botan/stream_cipher.h	58.3% 12 62.5% 8 - 0
build/include/botan/stream_mode.h	78.3% 23 73.3% 15 - 0
build/include/botan/sym_algo.h	94.1% 17 88.9% 9 - 0
build/include/botan/symkey.h	100% 7 100% 9 - 0
build/include/botan/tea.h	33.3% 3 60.0% 5 - 0
build/include/botan/threefish.h	40.0% 5 57.1% 7 - 0
build/include/botan/threefish_avx2.h	0.0% 2 0.0% 4 - 0
build/include/botan/tiger.h	50.0% 4 75.0% 4 - 0
build/include/botan/tls_alert.h	100% 5 100% 4 - 0
build/include/botan/tls_blocking.h	0.0% 3 0.0% 4 - 0
build/include/botan/tls_channel.h	87.5% 8 87.5% 8 - 0
build/include/botan/tls_ciphersuite.h	100% 13 100% 15 - 0
build/include/botan/tls_client.h	100% 1 50.0% 2 - 0
build/include/botan/tls_exceptn.h	0.0% 7 0.0% 7 - 0
build/include/botan/tls_handshake_msg.h	100% 2 66.7% 3 - 0
build/include/botan/tls_policy.h	50.0% 2 25.0% 4 - 0

build/include/botan/tls_server.h	100% 2 66.7% 3 - 0
build/include/botan/tls_server_info.h	44.4% 18 72.7% 11 - 0
build/include/botan/tls_session.h	47.1% 17 40.0% 15 - 0
build/include/botan/tls_session_manager.h	60.0% 5 50.0% 6 - 0
build/include/botan/tls_session_manager_sql.h	0.0% 3 0.0% 3 - 0
build/include/botan/tls_version.h	76.5% 17 80.0% 10 - 0
build/include/botan/transform.h	82.6% 23 75.0% 16 - 0
build/include/botan/transform_filter.h	100% 3 100% 4 - 0
build/include/botan/tss.h	100% 4 80.0% 5 - 0
build/include/botan/twofish.h	66.7% 3 80.0% 5 - 0
build/include/botan/whrlpool.h	83.3% 6 83.3% 6 - 0
build/include/botan/x509_crl.h	25.0% 4 42.9% 7 - 0
build/include/botan/x509_dn.h	100% 1 50.0% 6 - 0
build/include/botan/x509_ext.h	54.5% 77 58.1% 86 - 0
build/include/botan/x509_obj.h	100% 2 75.0% 4 - 0
build/include/botan/x509cert.h	100% 2 66.7% 6 - 0
build/include/botan/x509path.h	92.3% 13 90.0% 10 - 0
build/include/botan/x509self.h	100% 1 100% 1 - 0
build/include/botan/x919_mac.h	100% 4 100% 4 - 0
build/include/botan/x931_rng.h	100% 1 100% 2 - 0
build/include/botan/xtea.h	50.0% 4 66.7% 6 - 0
build/include/botan/xtea_simd.h	66.7% 3 80.0% 5 - 0
build/include/botan/xts.h	100% 7 90.0% 10 - 0
src/lib/alloc/locking_.../locking_allocator.cpp	87.8% 115 100% 10 - 0
src/lib/asn1/alg_id.cpp	93.8% 48 100% 9 - 0
src/lib/asn1/asn1_alt_name.cpp	52.4% 103 72.7% 11 - 04
src/lib/asn1/asn1_attribute.cpp	82.6% 23 75.0% 4 - 0
src/lib/asn1/asn1_obj.cpp	65.0% 20 50.0% 6 - 0
src/lib/asn1/asn1_oid.cpp	82.6% 86 80.0% 10 - 0
src/lib/asn1/asn1_str.cpp	85.1% 47 100% 8 - 0

src/lib/asn1/asn1_time.cpp	85.9% 156 83.3% 18 - 0
src/lib/asn1/ber_dec.cpp	65.0% 257 80.6% 36 - 0
src/lib/asn1/der_enc.cpp	84.2% 146 90.9% 33 - 0
src/lib/asn1/oid_lookup/default.cpp	100% 2 100% 1 - 0
src/lib/asn1/oid_lookup/oids.cpp	60.2% 83 61.1% 18 - 0
src/lib/asn1/x509_dn.cpp	69.4% 144 66.7% 18 - 0
src/lib/base/init.cpp	0.0% 4 0.0% 2 - 0
src/lib/base/lookup.cpp	87.2% 39 100% 14 - 0
src/lib/base/scan_name.cpp	75.5% 102 76.9% 13 - 0
src/lib/base/symkey.cpp	55.6% 36 60.0% 10 - 0
src/lib/block/aes/aes.cpp	91.7% 217 82.4% 17 - 0
src/lib/block/aes_ni/aes_ni.cpp	100% 528 100% 17 - 0
src/lib/block/aes_ssse3/aes_ssse3.cpp	95.9% 294 87.0% 23 - 0
src/lib/block/blowfish/blowfish.cpp	97.0% 99 100% 9 - 0
src/lib/block/camellia/camellia.cpp	96.0% 226 87.0% 23 - 0
src/lib/block/cascade/cascade.cpp	80.4% 51 75.0% 12 - 0
src/lib/block/cast/cast128.cpp	96.6% 119 91.7% 12 - 0
src/lib/block/cast/cast256.cpp	96.8% 124 88.9% 9 - 0
src/lib/block/des/des.cpp	100% 182 100% 13 - 0
src/lib/block/des/desx.cpp	82.1% 28 83.3% 6 - 0
src/lib/block/gost_28147/gost_28147.cpp	87.0% 69 90.0% 10 - 0
src/lib/block/idea/idea.cpp	95.1% 82 88.9% 9 - 0
src/lib/block/idea_sse2/idea_sse2.cpp	100% 128 100% 8 - 0
src/lib/block/kasumi/kasumi.cpp	96.4% 84 85.7% 7 - 0
src/lib/block/lion/lion.cpp	87.0% 77 80.0% 10 - 0
src/lib/block/mars/mars.cpp	98.4% 187 90.9% 11 - 0
src/lib/block/misty1/misty1.cpp	95.4% 87 85.7% 7 - 0
src/lib/block/noekeon/noekeon.cpp	96.4% 112 88.9% 9 - 0
src/lib/block/noekeon_simd/noekeon_simd.cpp	100% 72 100% 4 - 0
src/lib/block/rc2/rc2.cpp	92.8% 69 85.7% 7 - 0

src/lib/block/rc5/rc5.cpp	93.9% 66 87.5% 8 - 0
src/lib/block/rc6/rc6.cpp	100% 81 100% 6 - 0
src/lib/block/safer/safer_sk.cpp	95.9% 73 88.9% 9 - 0
src/lib/block/seed/seed.cpp	95.9% 73 85.7% 7 - 0
src/lib/block/serpent/serpent.cpp	97.7% 131 87.5% 8 - 0
src/lib/block/serpent_simd/serp_simd.cpp	100% 110 100% 6 - 0
src/lib/block/tea/tea.cpp	91.7% 36 83.3% 6 - 0
src/lib/block/threefish/threefish.cpp	92.7% 110 87.5% 8 - 0
src/lib/block/threefish_avx2/threefish_avx2.cpp	1.6% 126 33.3% 6 - 0
src/lib/block/twofish/twofish.cpp	100% 149 100% 7 - 0
src/lib/block/xtea/xtea.cpp	96.0% 75 87.5% 8 - 0
src/lib/block/xtea_simd/xtea_simd.cpp	100% 72 100% 6 - 0
src/lib/cert/x509/certstor.cpp	74.6% 59 72.7% 11 - 0
src/lib/cert/x509/crl_ent.cpp	77.8% 45 66.7% 6 - 0
src/lib/cert/x509/key_constraint.cpp	42.9% 28 50.0% 2 - 0
src/lib/cert/x509/ocsp.cpp	0.0% 113 0.0% 8 - 0
src/lib/cert/x509/ocsp_types.cpp	0.0% 54 0.0% 7 - 0
src/lib/cert/x509/pkcs10.cpp	74.4% 82 78.6% 14 - 0
src/lib/cert/x509/x509_ca.cpp	93.3% 104 88.9% 9 - 0
src/lib/cert/x509/x509_crl.cpp	87.0% 77 90.9% 11 - 0
src/lib/cert/x509/x509_ext.cpp	66.0% 294 68.3% 60 - 0
src/lib/cert/x509/x509_obj.cpp	74.7% 99 93.8% 16 - 0
src/lib/cert/x509/x509cert.cpp	49.0% 259 64.1% 39 - 0
src/lib/cert/x509/x509opt.cpp	63.6% 44 50.0% 8 - 0
src/lib/cert/x509/x509path.cpp	64.4% 163 68.8% 16 - 0
src/lib/cert/x509/x509self.cpp	93.8% 64 100% 3 - 0
src/lib/codec/base64/base64.cpp	87.1% 93 87.5% 8 - 0
src/lib/codec/hex/hex.cpp	80.0% 60 100% 9 - 0
src/lib/compression/compression.cpp	11.4% 114 18.8% 16 - 0
src/lib/credentials/credentials_manager.cpp	52.4% 42 38.5% 13 - 0

src/lib/entropy/dev_random/dev_random.cpp	91.4% 35 100% 4	- 0
src/lib/entropy/egd/es_egd.cpp	23.8% 63 62.5% 8	- 0
src/lib/entropy/entropy_srcs.cpp	95.2% 21 100% 2	- 0
src/lib/entropy/hres_timer/hres_timer.cpp	100% 14 100% 1	- 0
src/lib/entropy/proc_walk/proc_walk.cpp	0.0% 61 0.0% 6	- 0
src/lib/entropy/rdrand/rdrand.cpp	27.3% 11 100% 1	- 0
src/lib/entropy/unix_p.../unix_proc_sources.cpp	0.0% 39 0.0% 1	- 0
src/lib/entropy/unix_procs/unix_procs.cpp	16.8% 119 37.5% 8	- 0
src/lib/ffi/ffi.cpp	0.0% 440 0.0% 208	- 0
src/lib/filters/algo_filt.cpp	29.5% 44 30.0% 10	- 0
src/lib/filters/basefilt.cpp	14.8% 27 14.3% 7	- 0
src/lib/filters/buf_filt.cpp	55.8% 43 100% 3	- 0
src/lib/filters/codec_filt/b64_filt.cpp	97.4% 77 100% 8	- 0
src/lib/filters/codec_filt/hex_filt.cpp	0.0% 78 0.0% 8	- 0
src/lib/filters/comp_filter.cpp	0.0% 37 0.0% 8	- 0
src/lib/filters/data_snk.cpp	0.0% 19 0.0% 5	- 0
src/lib/filters/data_src.cpp	71.8% 71 73.3% 15	- 0
src/lib/filters/fd_unix/fd_unix.cpp	0.0% 20 0.0% 2	- 0
src/lib/filters/filter.cpp	84.5% 58 88.9% 9	- 0
src/lib/filters/key_filt.cpp	100% 14 100% 3	- 0
src/lib/filters/out_buf.cpp	79.2% 48 80.0% 10	- 0
src/lib/filters/pipe.cpp	51.4% 138 44.4% 27	- 0
src/lib/filters/pipe_io.cpp	5.9% 17 50.0% 4	- 0
src/lib/filters/pipe_rw.cpp	58.3% 60 50.0% 16	- 0
src/lib/filters/secqueue.cpp	54.8% 104 58.8% 17	- 0
src/lib/filters/threaded_fork.cpp	0.0% 56 0.0% 11	- 0
src/lib/filters/transform_filter.cpp	66.7% 60 71.4% 14	- 0
src/lib/hash/checksum/adler32/adler32.cpp	92.9% 42 100% 5	- 0
src/lib/hash/checksum/crc24/crc24.cpp	100% 33 100% 4	- 0
src/lib/hash/checksum/crc32/crc32.cpp	100% 33 100% 4	- 0

src/lib/hash/comb4p/comb4p.cpp	83.3% 54 88.9% 9 - 0
src/lib/hash/gost_3411/gost_3411.cpp	100% 157 100% 7 - 0
src/lib/hash/has160/has160.cpp	100% 98 100% 9 - 0
src/lib/hash/keccak/keccak.cpp	94.8% 115 77.8% 9 - 0
src/lib/hash/md2/md2.cpp	100% 45 100% 6 - 0
src/lib/hash/md4/md4.cpp	100% 59 100% 8 - 0
src/lib/hash/md5/md5.cpp	100% 71 100% 9 - 0
src/lib/hash/mdx_hash/mdx_hash.cpp	95.7% 46 100% 5 - 0
src/lib/hash/par_hash/par_hash.cpp	75.0% 48 80.0% 10 - 0
src/lib/hash/rmd128/rmd128.cpp	100% 108 100% 9 - 0
src/lib/hash/rmd160/rmd160.cpp	100% 137 100% 10 - 0
src/lib/hash/sha1/sha160.cpp	100% 91 100% 9 - 0
src/lib/hash/sha1_sse2/sha1_sse2.cpp	100% 141 100% 7 - 0
src/lib/hash/sha1_x86_64/sha1_x86_64.cpp	100% 7 100% 3 - 0
src/lib/hash/sha2_32/sha2_32.cpp	100% 137 100% 11 - 0
src/lib/hash/sha2_64/sha2_64.cpp	100% 171 100% 14 - 0
src/lib/hash/skein/skein_512.cpp	85.1% 87 83.3% 12 - 0
src/lib/hash/tiger/tiger.cpp	94.4% 108 100% 9 - 0
src/lib/hash/whirlpool/whirlpool.cpp	100% 96 100% 5 - 0
src/lib/kdf/hkdf/hkdf.cpp	90.9% 22 100% 4 - 0
src/lib/kdf/kdf.cpp	71.4% 7 100% 1 - 0
src/lib/kdf/kdf1/kdf1.cpp	72.7% 11 100% 3 - 0
src/lib/kdf/kdf2/kdf2.cpp	100% 15 100% 3 - 0
src/lib/kdf/prf_tls/prf_tls.cpp	85.7% 42 100% 7 - 0
src/lib/kdf/prf_x942/prf_x942.cpp	100% 40 100% 5 - 0
src/lib/mac/cbc_mac/cbc_mac.cpp	77.8% 45 70.0% 10 - 0
src/lib/mac/cmac/cmac.cpp	86.5% 89 81.8% 11 - 0
src/lib/mac/hmac/hmac.cpp	72.7% 44 80.0% 10 - 0
src/lib/mac/mac.cpp	0.0% 5 0.0% 1 - 0
src/lib/mac/poly1305/poly1305.cpp	95.6% 114 88.9% 9 - 0

src/lib/mac/siphash/siphash.cpp	89.4% 66 66.7% 9 - 0
src/lib/mac/x919_mac/x919_mac.cpp	77.8% 45 66.7% 9 - 0
src/lib/math/bigint/big_code.cpp	62.7% 75 100% 6 - 0
src/lib/math/bigint/big_io.cpp	3.7% 27 50.0% 4 - 0
src/lib/math/bigint/big_ops2.cpp	99.0% 98 100% 8 - 0
src/lib/math/bigint/big_ops3.cpp	95.6% 90 100% 8 - 0
src/lib/math/bigint/big_rand.cpp	86.7% 15 100% 2 - 0
src/lib/math/bigint/bigint.cpp	82.7% 127 90.0% 20 - 0
src/lib/math/bigint/divide.cpp	95.4% 65 100% 3 - 0
src/lib/math/ec_gfp/curve_gfp.cpp	97.2% 72 93.3% 15 - 0
src/lib/math/ec_gfp/curve_nistp.cpp	99.5% 412 100% 17 - 0
src/lib/math/ec_gfp/point_gfp.cpp	91.1% 292 100% 17 - 0
src/lib/math/mp/mp_asm.cpp	100% 79 100% 9 - 0
src/lib/math/mp/mp_comba.cpp	100% 903 100% 10 - 0
src/lib/math/mp/mp_karat.cpp	93.2% 147 100% 6 - 0
src/lib/math/mp/mp_misc.cpp	93.3% 30 100% 3 - 0
src/lib/math/mp/mp_monty.cpp	100% 34 100% 3 - 0
src/lib/math/mp/mp_mulop.cpp	100% 24 100% 2 - 0
src/lib/math/mp/mp_shift.cpp	100% 60 100% 4 - 0
src/lib/math/numbertheory/dsa_gen.cpp	0.0% 53 0.0% 7 - 0
src/lib/math/numbertheory/jacobi.cpp	88.0% 25 100% 1 - 0
src/lib/math/numbertheory/make_prm.cpp	61.5% 52 50.0% 2 - 0
src/lib/math/numbertheory/mp_numth.cpp	89.3% 28 100% 3 - 0
src/lib/math/numbertheory/numthry.cpp	94.2% 156 100% 10 - 0
src/lib/math/numbertheory/pow_mod.cpp	92.7% 82 92.9% 14 - 0
src/lib/math/numbertheory/powm_fw.cpp	96.2% 26 100% 4 - 0
src/lib/math/numbertheory/powm_mnt.cpp	98.1% 54 100% 4 - 0
src/lib/math/numbertheory/reducer.cpp	91.4% 35 100% 2 - 0
src/lib/math/numbertheory/ressol.cpp	85.4% 41 100% 1 - 0
src/lib/misc/aont/package.cpp	0.0% 49 0.0% 2 - 0

src/lib/misc/benchmark/benchmark.cpp	0.0% 74 0.0% 17 - 0
src/lib/misc/cryptobox/cryptobox.cpp	94.3% 53 100% 3 - 0
src/lib/misc/fpe_fe1/fpe_fe1.cpp	0.0% 73 0.0% 7 - 0
src/lib/misc/openpgp/openpgp.cpp	0.0% 105 0.0% 4 - 0
src/lib/misc/pbes2/pbes2.cpp	88.9% 72 100% 3 - 0
src/lib/misc/pem/pem.cpp	87.3% 71 100% 6 - 0
src/lib/misc/rfc3394/rfc3394.cpp	89.8% 49 100% 2 - 0
src/lib/misc/srp6/srp6.cpp	83.3% 60 85.7% 7 - 0
src/lib/misc/srp6/srp6_files.cpp	0.0% 32 0.0% 2 - 0
src/lib/misc/tss/tss.cpp	80.4% 102 75.0% 8 - 0
src/lib/modes/aead/aead.cpp	100% 6 100% 1 - 0
src/lib/modes/aead/ccm/ccm.cpp	84.3% 140 73.7% 19 - 0
src/lib/modes/aead/cha...5/chacha20poly1305.cpp	82.4% 91 84.6% 13 - 0
src/lib/modes/aead/eax/eax.cpp	75.9% 83 73.3% 15 - 0
src/lib/modes/aead/gcm/clmul/clmul.cpp	100% 43 100% 1 - 0
src/lib/modes/aead/gcm/gcm.cpp	74.0% 150 79.2% 24 - 0
src/lib/modes/aead/ocb/ocb.cpp	86.8% 219 79.3% 29 - 0
src/lib/modes/aead/siv/siv.cpp	80.6% 93 75.0% 16 - 0
src/lib/modes/cbc/cbc.cpp	85.8% 162 82.1% 28 - 0
src/lib/modes/cfb/cfb.cpp	83.3% 78 82.4% 17 - 0
src/lib/modes/cipher_mode.cpp	91.4% 35 100% 1 - 0
src/lib/modes/ecb/ecb.cpp	81.7% 82 81.0% 21 - 0
src/lib/modes/mode_pad/mode_pad.cpp	59.6% 52 71.4% 7 - 0
src/lib/modes/xts/xts.cpp	91.4% 162 81.8% 22 - 0
src/lib/passhash/bcrypt/bcrypt.cpp	95.2% 42 100% 5 - 0
src/lib/passhash/passhash9/passhash9.cpp	82.0% 61 100% 5 - 0
src/lib/pbkdf/pbkdf.cpp	94.1% 17 100% 4 - 0
src/lib/pbkdf/pbkdf1/pbkdf1.cpp	73.9% 23 100% 3 - 0
src/lib/pbkdf/pbkdf2/pbkdf2.cpp	88.0% 50 100% 5 - 0
src/lib/pk_pad/eme.cpp	50.0% 8 50.0% 4 - 0

src/lib/pk_pad/eme_oaep/oaep.cpp	94.9% 59 100% 7 - 0
src/lib/pk_pad/eme_pkcs1/eme_pkcs.cpp	76.7% 30 80.0% 5 - 0
src/lib/pk_pad/eme_raw/eme_raw.cpp	70.0% 10 80.0% 5 - 0
src/lib/pk_pad/ems1/ems1.cpp	86.4% 44 100% 7 - 0
src/lib/pk_pad/ems1_bsi/ems1_bsi.cpp	87.5% 8 100% 3 - 0
src/lib/pk_pad/ems1_pkcs1/ems1_pkcs1.cpp	84.6% 52 100% 13 - 0
src/lib/pk_pad/ems1_pssr/pssr.cpp	85.0% 80 88.9% 9 - 0
src/lib/pk_pad/ems1_raw/ems1_raw.cpp	75.0% 24 83.3% 6 - 0
src/lib/pk_pad/ems1_x931/ems1_x931.cpp	86.1% 36 100% 8 - 0
src/lib/pk_pad/get_pk_pad.cpp	75.0% 12 100% 2 - 0
src/lib/pk_pad/hash_id/hash_id.cpp	91.2% 34 100% 2 - 0
src/lib/pk_pad/mgf1/mgf1.cpp	100% 13 100% 1 - 0
src/lib/pubkey/blinding.cpp	88.9% 18 100% 3 - 0
src/lib/pubkey/curve25519/curve25519.cpp	95.0% 60 81.0% 21 - 0
src/lib/pubkey/curve25519/donna.cpp	100% 241 100% 14 - 0
src/lib/pubkey/dh/dh.cpp	95.2% 42 81.2% 16 - 0
src/lib/pubkey/dl_algo/dl_algo.cpp	81.1% 37 70.0% 10 - 0
src/lib/pubkey/dl_group/dl_group.cpp	54.0% 161 76.5% 17 - 0
src/lib/pubkey/dl_group/named.cpp	69.4% 36 100% 1 - 0
src/lib/pubkey/dlies/dlies.cpp	91.4% 58 100% 6 - 0
src/lib/pubkey/dsa/dsa.cpp	91.4% 81 80.8% 26 - 0
src/lib/pubkey/ec_group/ec_group.cpp	87.5% 72 80.0% 5 - 0
src/lib/pubkey/ec_group/named.cpp	98.3% 60 100% 1 - 0
src/lib/pubkey/ecc_key/ecc_key.cpp	80.3% 66 60.0% 15 - 0
src/lib/pubkey/ecdh/ecdh.cpp	100% 13 100% 6 - 0
src/lib/pubkey/ecdsa/ecdsa.cpp	91.4% 58 94.4% 18 - 0
src/lib/pubkey/elgamal/elgamal.cpp	89.0% 73 81.0% 21 - 0
src/lib/pubkey/gost_3410/gost_3410.cpp	65.6% 96 54.5% 22 - 0
src/lib/pubkey/if_algo/if_algo.cpp	94.7% 75 75.0% 12 - 0
src/lib/pubkey/keypair/keypair.cpp	76.0% 25 100% 2 - 0

src/lib/pubkey/mce/binary_matrix.cpp	97.4% 38 100% 3 - 0
src/lib/pubkey/mce/code_based_key_gen.cpp	95.8% 72 100% 3 - 0
src/lib/pubkey/mce/gf2m_rootfind_dcmp.cpp	98.4% 127 100% 11 - 0
src/lib/pubkey/mce/gf2m_small_m.cpp	58.3% 48 57.1% 7 - 0
src/lib/pubkey/mce/goppa_code.cpp	96.2% 79 100% 3 - 0
src/lib/pubkey/mce/mce_kem.cpp	100% 29 100% 4 - 0
src/lib/pubkey/mce/mceliece.cpp	100% 70 100% 8 - 0
src/lib/pubkey/mce/mceliece_key.cpp	85.5% 152 66.7% 18 - 0
src/lib/pubkey/mce/polyn_gf2m.cpp	89.8% 374 90.3% 31 - 0
src/lib/pubkey/mce/workfactor.cpp	0.0% 44 0.0% 7 - 0
src/lib/pubkey/mceies/mceies.cpp	91.3% 46 100% 3 - 0
src/lib/pubkey/nr/nr.cpp	88.1% 84 69.2% 26 - 0
src/lib/pubkey/pk_algs.cpp	88.0% 50 100% 2 - 0
src/lib/pubkey/pk_keys.cpp	50.0% 16 75.0% 4 - 0
src/lib/pubkey/pk_ops.cpp	89.2% 65 75.0% 24 - 0
src/lib/pubkey/pkcs8.cpp	82.6% 86 95.0% 20 - 0
src/lib/pubkey/pubkey.cpp	86.7% 90 95.5% 22 - 0
src/lib/pubkey/rfc6979/rfc6979.cpp	96.3% 27 100% 4 - 0
src/lib/pubkey/rsa/rsa.cpp	93.1% 101 97.3% 37 - 0
src/lib/pubkey/rw/rw.cpp	90.9% 77 94.4% 18 - 0
src/lib/pubkey/workfactor.cpp	71.4% 7 50.0% 2 - 0
src/lib/pubkey/x509_key.cpp	82.9% 35 83.3% 6 - 0
src/lib/rng/hmac_drbg/hmac_drbg.cpp	91.1% 56 87.5% 8 - 0
src/lib/rng/hmac_rng/hmac_rng.cpp	83.3% 72 80.0% 10 - 0
src/lib/rng/rng.cpp	100% 6 100% 1 - 0
src/lib/rng/system_rng/system_rng.cpp	69.0% 29 50.0% 10 - 0
src/lib/rng/x931_rng/x931_rng.cpp	73.2% 56 66.7% 9 - 0
src/lib/stream/chacha/chacha.cpp	93.3% 90 85.7% 7 - 0
src/lib/stream/ctr/ctr.cpp	83.3% 60 80.0% 10 - 0
src/lib/stream/ofb/ofb.cpp	78.0% 41 77.8% 9 - 0

src/lib/stream/rc4/rc4.cpp	87.9% 58 88.9% 9 - 0
src/lib/stream/salsa20/salsa20.cpp	93.7% 127 77.8% 9 - 0
src/lib/tls/msg_cert_req.cpp	91.7% 72 100% 5 - 0
src/lib/tls/msg_cert_verify.cpp	0.0% 31 0.0% 4 - 0
src/lib/tls/msg_certificate.cpp	88.6% 35 100% 3 - 0
src/lib/tls/msg_client_hello.cpp	57.4% 108 54.5% 11 - 0
src/lib/tls/msg_client_kex.cpp	30.0% 170 66.7% 3 - 0
src/lib/tls/msg_finished.cpp	100% 22 100% 5 - 0
src/lib/tls/msg_hello_verify.cpp	0.0% 29 0.0% 3 - 0
src/lib/tls/msg_server_hello.cpp	64.5% 76 85.7% 7 - 0
src/lib/tls/msg_server_kex.cpp	59.5% 126 87.5% 8 - 0
src/lib/tls/msg_session_ticket.cpp	75.0% 20 75.0% 4 - 0
src/lib/tls/sessions_s...ession_manager_sql.cpp	0.0% 84 0.0% 7 - 0
src/lib/tls/tls_alert.cpp	7.8% 51 33.3% 3 - 0
src/lib/tls/tls_blocking.cpp	0.0% 29 0.0% 6 - 0
src/lib/tls/tls_channel.cpp	62.8% 325 75.0% 36 - 0
src/lib/tls/tls_ciphersuite.cpp	47.4% 97 72.7% 11 - 0
src/lib/tls/tls_client.cpp	65.5% 206 90.0% 10 - 0
src/lib/tls/tls_extensions.cpp	67.4% 316 79.3% 29 - 0
src/lib/tls/tls_handshake_hash.cpp	90.9% 11 100% 3 - 0
src/lib/tls/tls_handshake_io.cpp	72.2% 187 90.5% 21 - 0
src/lib/tls/tls_handshake_state.cpp	74.3% 175 86.7% 30 - 0
src/lib/tls/tls_heartbeats.cpp	0.0% 40 0.0% 5 - 0
src/lib/tls/tls_policy.cpp	59.2% 152 60.0% 35 - 0
src/lib/tls/tls_record.cpp	90.5% 274 100% 12 - 0
src/lib/tls/tls_server.cpp	61.6% 271 91.7% 12 - 0
src/lib/tls/tls_session.cpp	38.7% 106 37.5% 8 - 0
src/lib/tls/tls_session_key.cpp	97.1% 34 100% 1 - 0
src/lib/tls/tls_session_manager_memory.cpp	39.5% 43 50.0% 6 - 0
src/lib/tls/tls_suite_info.cpp	100% 158 100% 1 - 0

src/lib/tls/tls_version.cpp	69.2% 39 87.5% 8 - 0
src/lib/utls/assert.cpp	90.0% 10 100% 1 - 0
src/lib/utls/calendar.cpp	100% 7 100% 2 - 0
src/lib/utls/charset.cpp	55.6% 90 77.8% 9 - 0
src/lib/utls/cpuid.cpp	46.7% 45 50.0% 2 - 0
src/lib/utls/datastor/datastor.cpp	70.3% 64 76.9% 13 - 0
src/lib/utls/fs.cpp	95.8% 24 100% 1 - 0
src/lib/utls/http_util/http_util.cpp	0.0% 93 0.0% 8 - 0
src/lib/utls/parsing.cpp	61.4% 153 64.3% 14 - 0
src/lib/utls/read_cfg.cpp	24.1% 29 50.0% 2 - 0
src/lib/utls/semaphore.cpp	0.0% 17 0.0% 3 - 0
src/lib/utls/ta_utils.cpp	0.0% 19 0.0% 6 - 0
src/lib/utls/version.cpp	0.0% 8 0.0% 6 - 0
src/lib/utls/zero_mem.cpp	100% 3 100% 1 - 0

=====

Total:|77.2% 27578|74.5% 5k| - 0

3.2.2 LibreSSL

Summary coverage rate:

lines.....: 28.3% (19905 of 70213 lines)

functions...: 31.3% (1648 of 5270 functions)

branches....: no data found

	Lines	Functions	Branches	
Filename	Rate Num	Rate Num	Rate Num	

=====

[libressl-2.2.2/]

crypto/aes/aes_cbc.c	100% 5 100% 1 - 0
crypto/aes/aes_core.c	98.6% 294 100% 4 - 0
crypto/aes/aes_ige.c	100% 77 100% 1 - 0

crypto/aes/aes_misc.c	0.0%	2 0.0%	1 - 0
crypto/aes/aes_wrap.c	77.6%	49 100%	2 - 0
crypto/asn1/a_bitstr.c	36.5%	115 33.3%	6 - 0
crypto/asn1/a_bool.c	0.0%	30 0.0%	2 - 0
crypto/asn1/a_bytes.c	0.0%	148 0.0%	4 - 0
crypto/asn1/a_d2i_fp.c	0.0%	117 0.0%	5 - 0
crypto/asn1/a_digest.c	77.8%	9 100%	1 - 0
crypto/asn1/a_dup.c	34.6%	26 50.0%	2 - 0
crypto/asn1/a_enum.c	0.0%	74 0.0%	4 - 0
crypto/asn1/a_gentm.c	0.0%	91 0.0%	5 - 0
crypto/asn1/a_i2d_fp.c	0.0%	54 0.0%	4 - 0
crypto/asn1/a_int.c	37.3%	220 44.4%	9 - 0
crypto/asn1/a_mbstr.c	41.8%	201 63.6%	11 - 0
crypto/asn1/a_object.c	24.9%	209 33.3%	9 - 0
crypto/asn1/a_octet.c	0.0%	6 0.0%	3 - 0
crypto/asn1/a_print.c	0.0%	44 0.0%	2 - 0
crypto/asn1/a_set.c	0.0%	90 0.0%	3 - 0
crypto/asn1/a_sign.c	0.0%	64 0.0%	2 - 0
crypto/asn1/a_strex.c	4.6%	306 7.1%	14 - 0
crypto/asn1/a_strnid.c	0.0%	91 0.0%	12 - 0
crypto/asn1/a_time.c	8.5%	71 10.0%	10 - 0
crypto/asn1/a_type.c	21.4%	42 50.0%	4 - 0
crypto/asn1/a_utctm.c	25.2%	103 33.3%	6 - 0
crypto/asn1/a_utf8.c	98.8%	86 100%	2 - 0
crypto/asn1/a_verify.c	0.0%	50 0.0%	1 - 0
crypto/asn1/ameth_lib.c	18.9%	190 40.0%	20 - 0
crypto/asn1/asn1_err.c	100%	5 100%	1 - 0
crypto/asn1/asn1_gen.c	0.0%	330 0.0%	9 - 0
crypto/asn1/asn1_lib.c	57.9%	247 61.5%	26 - 0
crypto/asn1/asn1_par.c	0.0%	212 0.0%	5 - 0

crypto/asn1/asn_mime.c	5.4%	504 8.0%	25 -	0
crypto/asn1/asn_moid.c	6.1%	49 25.0%	4 -	0
crypto/asn1/asn_pack.c	0.0%	72 0.0%	6 -	0
crypto/asn1/bio_asn1.c	0.0%	194 0.0%	18 -	0
crypto/asn1/bio_ndef.c	0.0%	77 0.0%	5 -	0
crypto/asn1/d2i_pr.c	15.7%	51 50.0%	2 -	0
crypto/asn1/evp_asn1.c	18.4%	76 50.0%	4 -	0
crypto/asn1/f_int.c	0.0%	94 0.0%	2 -	0
crypto/asn1/f_string.c	0.0%	88 0.0%	2 -	0
crypto/asn1/i2d_pr.c	0.0%	10 0.0%	1 -	0
crypto/asn1/n_pkey.c	0.0%	151 0.0%	13 -	0
crypto/asn1/nsseq.c	0.0%	14 0.0%	5 -	0
crypto/asn1/p5_pbe.c	0.0%	50 0.0%	6 -	0
crypto/asn1/p5_pbev2.c	0.0%	121 0.0%	11 -	0
crypto/asn1/p8_pkey.c	0.0%	56 0.0%	7 -	0
crypto/asn1/t_crl.c	0.0%	45 0.0%	2 -	0
crypto/asn1/t_pkey.c	64.7%	34 100%	1 -	0
crypto/asn1/t_req.c	0.0%	119 0.0%	3 -	0
crypto/asn1/t_spki.c	0.0%	23 0.0%	1 -	0
crypto/asn1/t_x509.c	0.0%	290 0.0%	12 -	0
crypto/asn1/t_x509a.c	0.0%	37 0.0%	1 -	0
crypto/asn1/tasn_dec.c	55.7%	546 69.2%	13 -	0
crypto/asn1/tasn_enc.c	80.2%	268 80.0%	10 -	0
crypto/asn1/tasn_fre.c	87.5%	104 100%	5 -	0
crypto/asn1/tasn_new.c	68.9%	151 100%	8 -	0
crypto/asn1/tasn_prn.c	0.0%	283 0.0%	21 -	0
crypto/asn1/tasn_typ.c	11.2%	179 11.2%	80 -	0
crypto/asn1/tasn_utl.c	83.5%	91 100%	10 -	0
crypto/asn1/x_algor.c	32.8%	58 27.3%	11 -	0
crypto/asn1/x_attrib.c	41.9%	31 33.3%	6 -	0

crypto/asn1/x_bignum.c	84.4% 32 100% 4 - 0
crypto/asn1/x_crl.c	0.0% 239 0.0% 30 - 0
crypto/asn1/x_exten.c	0.0% 15 0.0% 7 - 0
crypto/asn1/x_info.c	0.0% 28 0.0% 2 - 0
crypto/asn1/x_long.c	29.6% 54 20.0% 5 - 0
crypto/asn1/x_name.c	59.7% 243 59.1% 22 - 0
crypto/asn1/x_nx509.c	0.0% 9 0.0% 4 - 0
crypto/asn1/x_pkey.c	0.0% 37 0.0% 2 - 0
crypto/asn1/x_pubkey.c	14.8% 182 17.6% 17 - 0
crypto/asn1/x_req.c	0.0% 27 0.0% 10 - 0
crypto/asn1/x_sig.c	77.8% 9 75.0% 4 - 0
crypto/asn1/x_spki.c	0.0% 18 0.0% 8 - 0
crypto/asn1/x_val.c	0.0% 9 0.0% 4 - 0
crypto/asn1/x_x509.c	40.5% 74 26.7% 15 - 0
crypto/asn1/x_x509a.c	0.0% 94 0.0% 17 - 0
crypto/bf/bf_cfb64.c	100% 35 100% 1 - 0
crypto/bf/bf_ecb.c	83.3% 12 50.0% 2 - 0
crypto/bf/bf_enc.c	100% 114 100% 3 - 0
crypto/bf/bf_ofb64.c	100% 30 100% 1 - 0
crypto/bf/bf_skey.c	100% 31 100% 1 - 0
crypto/bio/b_dump.c	0.0% 54 0.0% 8 - 0
crypto/bio/b_posix.c	18.2% 11 33.3% 3 - 0
crypto/bio/b_print.c	46.2% 26 50.0% 4 - 0
crypto/bio/b_sock.c	0.0% 212 0.0% 8 - 0
crypto/bio/bf_buff.c	22.5% 284 55.6% 9 - 0
crypto/bio/bf_nbio.c	0.0% 88 0.0% 9 - 0
crypto/bio/bio_cb.c	0.0% 48 0.0% 1 - 0
crypto/bio/bio_err.c	100% 5 100% 1 - 0
crypto/bio/bio_lib.c	48.1% 291 47.4% 38 - 0
crypto/bio/bss_acpt.c	0.0% 194 0.0% 12 - 0

crypto/bio/bss_conn.c	0.0% 286 0.0% 13 - 0
crypto/bio/bss_dgram.c	0.0% 281 0.0% 13 - 0
crypto/bio/bss_fd.c	0.0% 96 0.0% 11 - 0
crypto/bio/bss_file.c	54.6% 119 100% 10 - 0
crypto/bio/bss_mem.c	73.5% 132 88.9% 9 - 0
crypto/bio/bss_null.c	69.0% 29 62.5% 8 - 0
crypto/bio/bss_sock.c	0.0% 80 0.0% 10 - 0
crypto/bn/bn_add.c	91.7% 133 100% 4 - 0
crypto/bn/bn_asm.c	88.9% 423 81.8% 11 - 0
crypto/bn/bn_blind.c	45.0% 131 46.2% 13 - 0
crypto/bn/bn_const.c	0.0% 16 0.0% 8 - 0
crypto/bn/bn_ctx.c	73.6% 140 81.2% 16 - 0
crypto/bn/bn_div.c	87.7% 114 100% 1 - 0
crypto/bn/bn_err.c	100% 5 100% 1 - 0
crypto/bn/bn_exp.c	77.9% 443 100% 9 - 0
crypto/bn/bn_exp2.c	70.7% 116 100% 1 - 0
crypto/bn/bn_gcd.c	38.7% 266 50.0% 4 - 0
crypto/bn/bn_gf2m.c	77.4% 523 90.5% 21 - 0
crypto/bn/bn_kron.c	75.9% 58 100% 1 - 0
crypto/bn/bn_lib.c	66.0% 368 76.7% 30 - 0
crypto/bn/bn_mod.c	59.6% 89 72.7% 11 - 0
crypto/bn/bn_mont.c	85.5% 165 100% 9 - 0
crypto/bn/bn_mul.c	37.7% 562 55.6% 9 - 0
crypto/bn/bn_nist.c	91.7% 424 100% 12 - 0
crypto/bn/bn_prime.c	69.0% 226 100% 8 - 0
crypto/bn/bn_print.c	29.6% 189 37.5% 8 - 0
crypto/bn/bn_rand.c	75.6% 82 100% 7 - 0
crypto/bn/bn_recip.c	73.8% 107 85.7% 7 - 0
crypto/bn/bn_shift.c	92.3% 91 100% 4 - 0
crypto/bn/bn_sqr.c	85.6% 97 100% 3 - 0

crypto/bn/bn_sqrt.c	54.0% 174 100% 1 - 0
crypto/bn/bn_word.c	74.4% 90 100% 5 - 0
crypto/buffer/buf_err.c	100% 5 100% 1 - 0
crypto/buffer/buffer.c	71.6% 74 80.0% 5 - 0
crypto/camellia/camellia.c	0.0% 147 0.0% 5 - 0
crypto/camellia/cmll_cbc.c	0.0% 5 0.0% 1 - 0
crypto/camellia/cmll_cfb.c	0.0% 9 0.0% 3 - 0
crypto/camellia/cmll_ecb.c	0.0% 5 0.0% 1 - 0
crypto/camellia/cmll_misc.c	0.0% 13 0.0% 3 - 0
crypto/camellia/cmll_ofb.c	0.0% 3 0.0% 1 - 0
crypto/cast/c_cfb64.c	100% 35 100% 1 - 0
crypto/cast/c_ecb.c	100% 10 100% 1 - 0
crypto/cast/c_enc.c	81.8% 110 100% 3 - 0
crypto/cast/c_ofb64.c	100% 30 100% 1 - 0
crypto/cast/c_skey.c	100% 68 100% 1 - 0
crypto/chacha/chacha-merged.c	96.1% 155 100% 3 - 0
crypto/chacha/chacha.c	92.3% 26 100% 4 - 0
crypto/cmac/cm_ameth.c	0.0% 6 0.0% 2 - 0
crypto/cmac/cm_pmeth.c	0.0% 74 0.0% 9 - 0
crypto/cmac/cmac.c	0.0% 122 0.0% 10 - 0
crypto/compat/arc4random.c	90.4% 73 100% 8 - 0
crypto/compat/arc4random_linux.h	54.5% 22 50.0% 4 - 0
crypto/compat/chacha_private.h	94.0% 117 100% 3 - 0
crypto/compat/explicit_bzero.c	100% 6 100% 2 - 0
crypto/compat/getentropy_linux.c	7.1% 196 28.6% 7 - 0
crypto/compat/reallocarray.c	50.0% 6 100% 1 - 0
crypto/compat/strlcat.c	0.0% 17 0.0% 1 - 0
crypto/compat/strlcpy.c	100% 12 100% 1 - 0
crypto/compat/timingsafe_bcmp.c	100% 6 100% 1 - 0
crypto/compat/timingsafe_memcmp.c	100% 10 100% 1 - 0

crypto/conf/conf_api.c	77.0% 113 93.3% 15 - 0
crypto/conf/conf_def.c	58.4% 365 60.0% 20 - 0
crypto/conf/conf_err.c	100% 5 100% 1 - 0
crypto/conf/conf_lib.c	14.5% 138 18.2% 22 - 0
crypto/conf/conf_mall.c	100% 4 100% 1 - 0
crypto/conf/conf_mod.c	20.0% 240 25.0% 24 - 0
crypto/conf/conf_sap.c	0.0% 17 0.0% 2 - 0
crypto/cpt_err.c	100% 5 100% 1 - 0
crypto/cryptlib.c	17.9% 196 26.5% 34 - 0
crypto/cversion.c	0.0% 10 0.0% 2 - 0
crypto/des/cbc_cksm.c	100% 25 100% 1 - 0
crypto/des/cfb64ede.c	38.7% 111 50.0% 2 - 0
crypto/des/cfb64enc.c	100% 35 100% 1 - 0
crypto/des/cfb_enc.c	89.5% 57 100% 1 - 0
crypto/des/des_enc.c	100% 193 100% 5 - 0
crypto/des/des_locl.h	100% 2 100% 1 - 0
crypto/des/ecb3_enc.c	100% 15 100% 1 - 0
crypto/des/ecb_enc.c	52.6% 19 50.0% 2 - 0
crypto/des/ede_cbcm_enc.c	100% 82 100% 1 - 0
crypto/des/fcrypt.c	100% 33 100% 2 - 0
crypto/des/fcrypt_b.c	100% 36 100% 1 - 0
crypto/des/ncbc_enc.c	100% 51 100% 1 - 0
crypto/des/ofb64ede.c	100% 30 100% 1 - 0
crypto/des/ofb64enc.c	100% 30 100% 1 - 0
crypto/des/ofb_enc.c	75.0% 44 100% 1 - 0
crypto/des/pcbc_enc.c	100% 39 100% 1 - 0
crypto/des/qud_cksm.c	96.0% 25 100% 1 - 0
crypto/des/rand_key.c	0.0% 5 0.0% 1 - 0
crypto/des/set_key.c	75.8% 62 57.1% 7 - 0
crypto/des/xcbc_enc.c	100% 58 100% 1 - 0

crypto/dh/dh_ameth.c	0.0%	208 0.0%	19 -	0
crypto/dh/dh_asn1.c	0.0%	17 0.0%	4 -	0
crypto/dh/dh_check.c	71.1%	45 100%	2 -	0
crypto/dh/dh_err.c	100%	5 100%	1 -	0
crypto/dh/dh_gen.c	58.5%	53 100%	2 -	0
crypto/dh/dh_key.c	76.2%	101 100%	8 -	0
crypto/dh/dh_lib.c	54.3%	94 45.5%	11 -	0
crypto/dh/dh_pmeth.c	0.0%	92 0.0%	8 -	0
crypto/dsa/dsa_ameth.c	12.3%	301 18.2%	22 -	0
crypto/dsa/dsa_asn1.c	43.9%	66 38.5%	13 -	0
crypto/dsa/dsa_err.c	100%	5 100%	1 -	0
crypto/dsa/dsa_gen.c	64.4%	174 100%	2 -	0
crypto/dsa/dsa_key.c	78.4%	37 100%	2 -	0
crypto/dsa/dsa_lib.c	39.2%	125 41.7%	12 -	0
crypto/dsa/dsa_ossl.c	64.5%	169 100%	6 -	0
crypto/dsa/dsa_pmeth.c	0.0%	122 0.0%	9 -	0
crypto/dsa/dsa_prn.c	20.0%	30 25.0%	4 -	0
crypto/dsa/dsa_sign.c	94.1%	17 100%	4 -	0
crypto/dsa/dsa_vrf.c	100%	2 100%	1 -	0
crypto/dso/dso_err.c	100%	5 100%	1 -	0
crypto/dso/dso_lib.c	0.0%	211 0.0%	21 -	0
crypto/dso/dso_null.c	0.0%	2 0.0%	1 -	0
crypto/dso/dso_openssl.c	0.0%	2 0.0%	1 -	0
crypto/ec/ec2_mult.c	66.5%	221 100%	7 -	0
crypto/ec/ec2_oct.c	36.8%	193 66.7%	3 -	0
crypto/ec/ec2_smpl.c	59.9%	372 96.3%	27 -	0
crypto/ec/ec_ameth.c	0.0%	288 0.0%	22 -	0
crypto/ec/ec_asn1.c	0.0%	641 0.0%	31 -	0
crypto/ec/ec_check.c	59.0%	39 100%	1 -	0
crypto/ec/ec_curve.c	58.8%	97 60.0%	5 -	0

crypto/ec/ec_cvt.c	67.7% 31 100% 2 - 0
crypto/ec/ec_err.c	100% 5 100% 1 - 0
crypto/ec/ec_key.c	40.9% 264 50.0% 26 - 0
crypto/ec/ec_lib.c	55.4% 516 81.0% 58 - 0
crypto/ec/ec_mult.c	70.3% 404 87.5% 8 - 0
crypto/ec/ec_oct.c	44.2% 52 75.0% 4 - 0
crypto/ec/ec_pmeth.c	0.0% 127 0.0% 10 - 0
crypto/ec/ec_print.c	0.0% 66 0.0% 4 - 0
crypto/ec/eck_prn.c	0.0% 181 0.0% 7 - 0
crypto/ec/ecp_mont.c	64.8% 108 90.9% 11 - 0
crypto/ec/ecp_nist.c	75.8% 62 100% 5 - 0
crypto/ec/ecp_oct.c	51.7% 211 100% 3 - 0
crypto/ec/ecp_smpl.c	61.7% 759 85.7% 28 - 0
crypto/ecdh/ech_err.c	100% 5 100% 1 - 0
crypto/ecdh/ech_key.c	80.0% 5 100% 1 - 0
crypto/ecdh/ech_lib.c	44.0% 75 45.5% 11 - 0
crypto/ecdh/ech_ossl.c	62.3% 61 100% 2 - 0
crypto/ecdsa/ecs_asn1.c	100% 9 100% 4 - 0
crypto/ecdsa/ecs_err.c	100% 5 100% 1 - 0
crypto/ecdsa/ecs_lib.c	52.6% 97 50.0% 12 - 0
crypto/ecdsa/ecs_ossl.c	56.2% 201 100% 4 - 0
crypto/ecdsa/ecs_sign.c	72.7% 22 80.0% 5 - 0
crypto/ecdsa/ecs_vrf.c	91.3% 23 100% 2 - 0
crypto/engine/eng_all.c	100% 4 100% 1 - 0
crypto/engine/eng_cnf.c	3.1% 96 16.7% 6 - 0
crypto/engine/eng_ctrl.c	0.0% 156 0.0% 8 - 0
crypto/engine/eng_err.c	100% 5 100% 1 - 0
crypto/engine/eng_fat.c	4.0% 75 20.0% 5 - 0
crypto/engine/eng_init.c	0.0% 43 0.0% 4 - 0
crypto/engine/eng_lib.c	47.4% 135 40.0% 30 - 0

crypto/engine/eng_list.c	49.4% 166 58.3% 12 - 0
crypto/engine/eng_pkey.c	0.0% 65 0.0% 9 - 0
crypto/engine/eng_table.c	4.4% 137 5.6% 18 - 0
crypto/engine/tb_asnmth.c	6.4% 78 15.4% 13 - 0
crypto/engine/tb_cipher.c	5.7% 35 11.1% 9 - 0
crypto/engine/tb_dh.c	8.0% 25 12.5% 8 - 0
crypto/engine/tb_digest.c	14.3% 35 22.2% 9 - 0
crypto/engine/tb_dsa.c	8.0% 25 12.5% 8 - 0
crypto/engine/tb_ecdh.c	8.0% 25 12.5% 8 - 0
crypto/engine/tb_ecdsa.c	8.0% 25 12.5% 8 - 0
crypto/engine/tb_pkmeth.c	11.9% 42 20.0% 10 - 0
crypto/engine/tb_rand.c	0.0% 25 0.0% 8 - 0
crypto/engine/tb_rsa.c	8.0% 25 12.5% 8 - 0
crypto/err/err.c	56.5% 428 57.6% 59 - 0
crypto/err/err_all.c	100% 29 100% 1 - 0
crypto/err/err_prn.c	50.0% 22 60.0% 5 - 0
crypto/evp/bio_b64.c	74.2% 291 75.0% 8 - 0
crypto/evp/bio_enc.c	66.8% 187 75.0% 8 - 0
crypto/evp/bio_md.c	55.7% 106 75.0% 8 - 0
crypto/evp/c_all.c	97.4% 156 75.0% 4 - 0
crypto/evp/digest.c	62.2% 143 84.6% 13 - 0
crypto/evp/e_aes.c	11.2% 484 58.9% 56 - 0
crypto/evp/e_aes_cbc_hmac_sha1.c	100% 4 100% 2 - 0
crypto/evp/e_bf.c	100% 4 100% 9 - 0
crypto/evp/e_camellia.c	60.0% 15 48.6% 37 - 0
crypto/evp/e_cast.c	100% 4 100% 9 - 0
crypto/evp/e_chacha.c	18.2% 11 33.3% 3 - 0
crypto/evp/e_chacha20poly1305.c	0.0% 86 0.0% 6 - 0
crypto/evp/e_des.c	45.0% 80 85.7% 14 - 0
crypto/evp/e_des3.c	47.4% 97 76.2% 21 - 0

crypto/evp/e_gost2814789.c	22.5%	89 80.0%	10	-	0
crypto/evp/e_idea.c	5.0%	20 44.4%	9	-	0
crypto/evp/e_null.c	0.0%	8 0.0%	3	-	0
crypto/evp/e_rc2.c	20.3%	64 75.0%	16	-	0
crypto/evp/e_rc4.c	100%	10 100%	4	-	0
crypto/evp/e_rc4_hmac_md5.c	2.9%	69 25.0%	4	-	0
crypto/evp/e_xcbc_d.c	71.4%	21 100%	3	-	0
crypto/evp/encode.c	91.4%	185 100%	8	-	0
crypto/evp/evp_aead.c	0.0%	47 0.0%	9	-	0
crypto/evp/evp_enc.c	46.1%	319 54.2%	24	-	0
crypto/evp/evp_err.c	100%	5 100%	1	-	0
crypto/evp/evp_key.c	52.3%	88 20.0%	5	-	0
crypto/evp/evp_lib.c	56.6%	113 58.1%	31	-	0
crypto/evp/evp_pbe.c	4.2%	95 10.0%	10	-	0
crypto/evp/evp_pkey.c	0.0%	75 0.0%	13	-	0
crypto/evp/m_dss.c	25.0%	8 25.0%	4	-	0
crypto/evp/m_dss1.c	25.0%	8 25.0%	4	-	0
crypto/evp/m_ecdsa.c	25.0%	8 25.0%	4	-	0
crypto/evp/m_gost2814789.c	92.9%	14 100%	5	-	0
crypto/evp/m_gostr341194.c	100%	8 100%	4	-	0
crypto/evp/m_md4.c	100%	8 100%	4	-	0
crypto/evp/m_md5.c	100%	8 100%	4	-	0
crypto/evp/m_ripemd.c	100%	8 100%	4	-	0
crypto/evp/m_sha.c	100%	8 100%	4	-	0
crypto/evp/m_sha1.c	100%	32 100%	16	-	0
crypto/evp/m_sigver.c	52.1%	73 60.0%	5	-	0
crypto/evp/m_streebog.c	25.0%	16 25.0%	8	-	0
crypto/evp/m_wp.c	25.0%	8 25.0%	4	-	0
crypto/evp/names.c	49.3%	75 45.5%	11	-	0
crypto/evp/p5_crpt.c	0.0%	58 0.0%	2	-	0

crypto/evp/p5_crpt2.c	28.6% 126 25.0% 4 - 0
crypto/evp/p_lib.c	38.7% 194 41.9% 31 - 0
crypto/evp/p_sign.c	0.0% 41 0.0% 1 - 0
crypto/evp/p_verify.c	44.4% 36 100% 1 - 0
crypto/evp/pmeth_fn.c	25.5% 153 61.5% 13 - 0
crypto/evp/pmeth_gn.c	27.6% 87 30.0% 10 - 0
crypto/evp/pmeth_lib.c	29.6% 267 34.2% 38 - 0
crypto/ex_data.c	49.8% 215 65.4% 26 - 0
crypto/gost/gost2814789.c	79.2% 207 92.9% 14 - 0
crypto/gost/gost89_keywrap.c	0.0% 40 0.0% 3 - 0
crypto/gost/gost89_params.c	94.1% 34 100% 3 - 0
crypto/gost/gost89imit_ameth.c	37.5% 8 50.0% 2 - 0
crypto/gost/gost89imit_pmeth.c	59.0% 83 87.5% 8 - 0
crypto/gost/gost_asn1.c	0.0% 45 0.0% 20 - 0
crypto/gost/gost_err.c	100% 5 100% 1 - 0
crypto/gost/gostr341001.c	0.0% 234 0.0% 7 - 0
crypto/gost/gostr341001_ameth.c	0.0% 389 0.0% 19 - 0
crypto/gost/gostr341001_key.c	0.0% 139 0.0% 13 - 0
crypto/gost/gostr341001_params.c	0.0% 20 0.0% 4 - 0
crypto/gost/gostr341001_pmeth.c	0.0% 370 0.0% 17 - 0
crypto/gost/gostr341194.c	86.6% 112 80.0% 10 - 0
crypto/gost/streebog.c	0.0% 166 0.0% 14 - 0
crypto/hmac/hm_ameth.c	0.0% 38 0.0% 5 - 0
crypto/hmac/hm_pmeth.c	0.0% 99 0.0% 9 - 0
crypto/hmac/hmac.c	65.7% 102 88.9% 9 - 0
crypto/idea/i_cbc.c	100% 77 100% 2 - 0
crypto/idea/i_cfb64.c	100% 35 100% 1 - 0
crypto/idea/i_ecb.c	80.0% 10 50.0% 2 - 0
crypto/idea/i_ofb64.c	0.0% 30 0.0% 1 - 0
crypto/idea/i_skey.c	98.4% 64 100% 3 - 0

crypto/lhash/lh_stats.c	0.0% 65 0.0% 6 - 0
crypto/lhash/lhash.c	89.1% 193 100% 13 - 0
crypto/malloc-wrapper.c	2.2% 89 5.0% 20 - 0
crypto/md32_common.h	93.3% 60 65.2% 23 - 0
crypto/md4/md4_dgst.c	100% 69 100% 2 - 0
crypto/md5/md5_dgst.c	100% 84 100% 2 - 0
crypto/mem_clr.c	100% 3 100% 1 - 0
crypto/mem_dbg.c	30.8% 26 30.8% 13 - 0
crypto/modes/cbc128.c	64.5% 62 100% 2 - 0
crypto/modes/ccm128.c	0.0% 227 0.0% 10 - 0
crypto/modes/cfb128.c	91.7% 72 100% 4 - 0
crypto/modes/ctr128.c	0.0% 77 0.0% 5 - 0
crypto/modes/cts128.c	100% 123 100% 8 - 0
crypto/modes/gcm128.c	60.1% 411 64.3% 14 - 0
crypto/modes/ofb128.c	87.0% 23 100% 1 - 0
crypto/modes/xts128.c	0.0% 49 0.0% 1 - 0
crypto/o_init.c	100% 2 100% 1 - 0
crypto/o_time.c	0.0% 45 0.0% 3 - 0
crypto/objects/o_names.c	49.7% 163 57.9% 19 - 0
crypto/objects/obj_dat.c	19.1% 408 45.9% 37 - 0
crypto/objects/obj_err.c	100% 5 100% 1 - 0
crypto/objects/obj_lib.c	53.5% 43 100% 2 - 0
crypto/objects/obj_xref.c	5.1% 78 8.3% 12 - 0
crypto/ocsp/ocsp_asn.c	0.0% 135 0.0% 60 - 0
crypto/ocsp/ocsp_cl.c	0.0% 146 0.0% 12 - 0
crypto/ocsp/ocsp_err.c	100% 5 100% 1 - 0
crypto/ocsp/ocsp_ext.c	0.0% 205 0.0% 45 - 0
crypto/ocsp/ocsp_ht.c	0.0% 193 0.0% 7 - 0
crypto/ocsp/ocsp_lib.c	0.0% 110 0.0% 6 - 0
crypto/ocsp/ocsp_prn.c	0.0% 152 0.0% 7 - 0

crypto/ocsp/ocsp_srv.c	0.0% 116 0.0% 9 - 0
crypto/ocsp/ocsp_vfy.c	0.0% 190 0.0% 9 - 0
crypto/pem/pem_all.c	1.5% 67 3.1% 65 - 0
crypto/pem/pem_err.c	100% 5 100% 1 - 0
crypto/pem/pem_info.c	0.0% 198 0.0% 3 - 0
crypto/pem/pem_lib.c	35.7% 487 50.0% 16 - 0
crypto/pem/pem_oth.c	83.3% 12 100% 1 - 0
crypto/pem/pem_pk8.c	0.0% 85 0.0% 20 - 0
crypto/pem/pem_pkey.c	18.0% 100 16.7% 6 - 0
crypto/pem/pem_x509.c	0.0% 1 0.0% 4 - 0
crypto/pem/pem_xaux.c	50.0% 2 12.5% 8 - 0
crypto/pem/pvkfmt.c	0.0% 499 0.0% 28 - 0
crypto/pkcs12/p12_add.c	0.0% 95 0.0% 10 - 0
crypto/pkcs12/p12_asn.c	0.0% 36 0.0% 16 - 0
crypto/pkcs12/p12_attr.c	0.0% 38 0.0% 7 - 0
crypto/pkcs12/p12_crpt.c	0.0% 29 0.0% 2 - 0
crypto/pkcs12/p12_crt.c	0.0% 166 0.0% 7 - 0
crypto/pkcs12/p12_decr.c	0.0% 58 0.0% 3 - 0
crypto/pkcs12/p12_init.c	0.0% 18 0.0% 1 - 0
crypto/pkcs12/p12_key.c	0.0% 88 0.0% 2 - 0
crypto/pkcs12/p12_kiss.c	0.0% 124 0.0% 4 - 0
crypto/pkcs12/p12_mutl.c	0.0% 76 0.0% 4 - 0
crypto/pkcs12/p12_p8d.c	0.0% 2 0.0% 1 - 0
crypto/pkcs12/p12_p8e.c	0.0% 21 0.0% 1 - 0
crypto/pkcs12/p12_utl.c	0.0% 51 0.0% 10 - 0
crypto/pkcs12/pk12err.c	100% 5 100% 1 - 0
crypto/pkcs7/pk7_asn1.c	26.7% 120 23.9% 46 - 0
crypto/pkcs7/pk7_attr.c	54.5% 55 83.3% 6 - 0
crypto/pkcs7/pk7_doit.c	48.5% 686 70.8% 24 - 0
crypto/pkcs7/pk7_lib.c	44.1% 340 75.0% 20 - 0

crypto/pkcs7/pk7_mime.c	0.0%	13 0.0%	4 -	0
crypto/pkcs7/pk7_smime.c	52.8%	284 90.0%	10 -	0
crypto/pkcs7/pkcs7err.c	100%	5 100%	1 -	0
crypto/poly1305/poly1305-donna.c	100%	212 100%	6 -	0
crypto/poly1305/poly1305.c	100%	9 100%	3 -	0
crypto/rand/rand_err.c	100%	5 100%	1 -	0
crypto/rand/rand_lib.c	23.1%	26 18.2%	11 -	0
crypto/rc2/rc2_cbc.c	85.8%	113 100%	3 -	0
crypto/rc2/rc2_ecb.c	100%	10 100%	1 -	0
crypto/rc2/rc2_skey.c	100%	25 100%	1 -	0
crypto/rc2/rc2cfb64.c	100%	35 100%	1 -	0
crypto/rc2/rc2ofb64.c	100%	30 100%	1 -	0
crypto/rc4/rc4_enc.c	93.8%	64 100%	1 -	0
crypto/rc4/rc4_skey.c	85.7%	14 50.0%	2 -	0
crypto/ripemd/rmd_dgst.c	100%	182 100%	2 -	0
crypto/rsa/rsa_ameth.c	9.1%	330 25.0%	20 -	0
crypto/rsa/rsa_asn1.c	34.4%	32 27.3%	11 -	0
crypto/rsa/rsa_chk.c	0.0%	97 0.0%	1 -	0
crypto/rsa/rsa_crpt.c	36.0%	75 60.0%	10 -	0
crypto/rsa/rsa_eay.c	57.3%	422 90.9%	11 -	0
crypto/rsa/rsa_err.c	100%	5 100%	1 -	0
crypto/rsa/rsa_gen.c	0.0%	112 0.0%	2 -	0
crypto/rsa/rsa_lib.c	33.0%	103 27.3%	11 -	0
crypto/rsa/rsa_none.c	0.0%	16 0.0%	2 -	0
crypto/rsa/rsa_oaep.c	0.0%	106 0.0%	4 -	0
crypto/rsa/rsa_pk1.c	71.8%	78 100%	4 -	0
crypto/rsa/rsa_pmeth.c	25.6%	285 69.2%	13 -	0
crypto/rsa/rsa_prn.c	0.0%	15 0.0%	2 -	0
crypto/rsa/rsa_pss.c	0.0%	124 0.0%	4 -	0
crypto/rsa/rsa_sign.c	58.2%	110 100%	3 -	0

crypto/rsa/rsa_ssl.c	0.0% 46 0.0% 2 - 0
crypto/rsa/rsa_x931.c	0.0% 49 0.0% 3 - 0
crypto/sha/sha1_one.c	87.5% 8 100% 1 - 0
crypto/sha/sha256.c	73.1% 67 42.9% 7 - 0
crypto/sha/sha512.c	86.4% 162 50.0% 10 - 0
crypto/sha/sha_locl.h	100% 110 100% 4 - 0
crypto/stack/stack.c	60.0% 145 68.2% 22 - 0
crypto/ts/ts_asn1.c	0.0% 177 0.0% 64 - 0
crypto/ts/ts_conf.c	0.0% 244 0.0% 21 - 0
crypto/ts/ts_err.c	100% 5 100% 1 - 0
crypto/ts/ts_lib.c	0.0% 40 0.0% 5 - 0
crypto/ts/ts_req_print.c	0.0% 21 0.0% 1 - 0
crypto/ts/ts_req_utils.c	0.0% 84 0.0% 24 - 0
crypto/ts/ts_rsp_print.c	0.0% 99 0.0% 5 - 0
crypto/ts/ts_rsp_sign.c	0.0% 489 0.0% 34 - 0
crypto/ts/ts_rsp_utils.c	0.0% 178 0.0% 39 - 0
crypto/ts/ts_rsp_verify.c	0.0% 295 0.0% 17 - 0
crypto/ts/ts_verify_ctx.c	0.0% 54 0.0% 5 - 0
crypto/txt_db/txt_db.c	0.0% 208 0.0% 6 - 0
crypto/ui/ui_err.c	100% 5 100% 1 - 0
crypto/ui/ui_lib.c	6.5% 399 10.9% 55 - 0
crypto/ui/ui_openssl.c	0.0% 127 0.0% 12 - 0
crypto/whrlpool/wp_block.c	0.0% 62 0.0% 1 - 0
crypto/whrlpool/wp_dgst.c	0.0% 112 0.0% 5 - 0
crypto/x509/by_dir.c	0.0% 178 0.0% 9 - 0
crypto/x509/by_file.c	0.0% 126 0.0% 5 - 0
crypto/x509/by_mem.c	0.0% 32 0.0% 2 - 0
crypto/x509/x509_att.c	0.0% 186 0.0% 19 - 0
crypto/x509/x509_cmp.c	31.1% 132 36.4% 22 - 0
crypto/x509/x509_d2.c	0.0% 36 0.0% 3 - 0

crypto/x509/x509_def.c	16.7%	12 16.7%	6	-	0
crypto/x509/x509_err.c	100%	5 100%	1	-	0
crypto/x509/x509_ext.c	11.1%	54 11.1%	27	-	0
crypto/x509/x509_lu.c	18.6%	365 25.0%	32	-	0
crypto/x509/x509_obj.c	57.0%	79 100%	1	-	0
crypto/x509/x509_req.c	0.0%	134 0.0%	18	-	0
crypto/x509/x509_set.c	0.0%	49 0.0%	7	-	0
crypto/x509/x509_trs.c	16.3%	123 29.4%	17	-	0
crypto/x509/x509_txt.c	0.0%	58 0.0%	1	-	0
crypto/x509/x509_v3.c	10.8%	130 26.7%	15	-	0
crypto/x509/x509_vfy.c	19.9%	1117 25.8%	66	-	0
crypto/x509/x509_vpm.c	30.6%	144 34.8%	23	-	0
crypto/x509/x509cset.c	0.0%	58 0.0%	7	-	0
crypto/x509/x509name.c	0.0%	180 0.0%	18	-	0
crypto/x509/x509rset.c	0.0%	12 0.0%	3	-	0
crypto/x509/x509spki.c	0.0%	37 0.0%	4	-	0
crypto/x509/x509type.c	0.0%	37 0.0%	1	-	0
crypto/x509/x_all.c	1.1%	187 1.3%	78	-	0
crypto/x509v3/pcy_cache.c	0.0%	116 0.0%	7	-	0
crypto/x509v3/pcy_data.c	0.0%	37 0.0%	2	-	0
crypto/x509v3/pcy_lib.c	0.0%	47 0.0%	9	-	0
crypto/x509v3/pcy_map.c	0.0%	35 0.0%	1	-	0
crypto/x509v3/pcy_node.c	0.0%	73 0.0%	7	-	0
crypto/x509v3/pcy_tree.c	0.0%	304 0.0%	14	-	0
crypto/x509v3/v3_akey.c	0.0%	74 0.0%	2	-	0
crypto/x509v3/v3_akeya.c	0.0%	9 0.0%	4	-	0
crypto/x509v3/v3_alt.c	0.0%	299 0.0%	13	-	0
crypto/x509v3/v3_bcons.c	9.4%	32 16.7%	6	-	0
crypto/x509v3/v3_bitst.c	0.0%	25 0.0%	2	-	0
crypto/x509v3/v3_conf.c	0.0%	232 0.0%	29	-	0

crypto/x509v3/v3_cpols.c	0.0% 274 0.0% 28 - 0
crypto/x509v3/v3_crl.d.c	0.0% 305 0.0% 29 - 0
crypto/x509v3/v3_enum.c	0.0% 6 0.0% 1 - 0
crypto/x509v3/v3_extku.c	0.0% 36 0.0% 6 - 0
crypto/x509v3/v3_genn.c	0.0% 107 0.0% 23 - 0
crypto/x509v3/v3_ia5.c	0.0% 22 0.0% 2 - 0
crypto/x509v3/v3_info.c	0.0% 79 0.0% 11 - 0
crypto/x509v3/v3_int.c	0.0% 2 0.0% 1 - 0
crypto/x509v3/v3_lib.c	28.6% 133 50.0% 14 - 0
crypto/x509v3/v3_ncons.c	0.0% 211 0.0% 15 - 0
crypto/x509v3/v3_ocsp.c	0.0% 93 0.0% 11 - 0
crypto/x509v3/v3_pci.c	0.0% 173 0.0% 3 - 0
crypto/x509v3/v3_pcia.c	0.0% 18 0.0% 8 - 0
crypto/x509v3/v3_pcons.c	0.0% 32 0.0% 4 - 0
crypto/x509v3/v3_pku.c	0.0% 20 0.0% 5 - 0
crypto/x509v3/v3_pmaps.c	0.0% 50 0.0% 4 - 0
crypto/x509v3/v3_prn.c	0.0% 93 0.0% 5 - 0
crypto/x509v3/v3_purp.c	22.4% 379 38.9% 36 - 0
crypto/x509v3/v3_skey.c	0.0% 36 0.0% 3 - 0
crypto/x509v3/v3_sxnet.c	0.0% 109 0.0% 16 - 0
crypto/x509v3/v3_utl.c	0.0% 456 0.0% 32 - 0
crypto/x509v3/v3err.c	100% 5 100% 1 - 0
ssl/bio_ssl.c	0.0% 293 0.0% 13 - 0
ssl/bs_ber.c	76.4% 89 100% 6 - 0
ssl/bs_cbb.c	85.8% 197 100% 19 - 0
ssl/bs_cbs.c	78.9% 213 87.9% 33 - 0
ssl/d1_both.c	0.0% 610 0.0% 29 - 0
ssl/d1_clnt.c	0.4% 544 12.5% 8 - 0
ssl/d1_enc.c	0.0% 44 0.0% 1 - 0
ssl/d1_lib.c	33.7% 199 37.5% 16 - 0

ssl/d1_meth.c	33.3% 6 50.0% 2 - 0
ssl/d1_pkt.c	0.0% 609 0.0% 17 - 0
ssl/d1_srtp.c	0.0% 149 0.0% 11 - 0
ssl/d1_srvr.c	0.3% 623 9.1% 11 - 0
ssl/pqueue.c	14.8% 61 27.3% 11 - 0
ssl/s23_clnt.c	0.9% 229 12.5% 8 - 0
ssl/s23_lib.c	5.3% 38 25.0% 4 - 0
ssl/s23_meth.c	11.1% 18 25.0% 4 - 0
ssl/s23_pkt.c	0.0% 30 0.0% 2 - 0
ssl/s23_srvr.c	0.8% 243 14.3% 7 - 0
ssl/s3_both.c	2.6% 312 12.5% 16 - 0
ssl/s3_cbc.c	0.0% 247 0.0% 12 - 0
ssl/s3_clnt.c	0.2% 1287 5.3% 19 - 0
ssl/s3_enc.c	2.1% 374 12.5% 16 - 0
ssl/s3_lib.c	17.0% 466 37.9% 29 - 0
ssl/s3_meth.c	33.3% 6 50.0% 2 - 0
ssl/s3_pkt.c	0.0% 611 0.0% 9 - 0
ssl/s3_srvr.c	0.2% 1299 6.2% 16 - 0
ssl/ssl_algs.c	100% 40 100% 1 - 0
ssl/ssl_asn1.c	71.2% 410 100% 2 - 0
ssl/ssl_cert.c	19.9% 286 21.7% 23 - 0
ssl/ssl_ciph.c	52.7% 702 58.3% 24 - 0
ssl/ssl_err.c	100% 5 100% 1 - 0
ssl/ssl_err2.c	100% 4 100% 1 - 0
ssl/ssl_lib.c	18.7% 1404 10.0% 160 - 0
ssl/ssl_rsa.c	0.0% 368 0.0% 23 - 0
ssl/ssl_sess.c	3.5% 434 4.0% 50 - 0
ssl/ssl_stat.c	0.0% 430 0.0% 8 - 0
ssl/ssl_txt.c	0.0% 61 0.0% 2 - 0
ssl/t1_clnt.c	42.9% 14 75.0% 4 - 0

ssl/t1_enc.c	0.0% 601 0.0% 15 - 0
ssl/t1_lib.c	1.4% 1079 12.1% 33 - 0
ssl/t1_meth.c	42.9% 14 75.0% 4 - 0
ssl/t1_reneg.c	0.0% 84 0.0% 4 - 0
ssl/t1_srvr.c	42.9% 14 75.0% 4 - 0

=====

Total:|28.3% 70213|31.3% 5270| - 0

3.2.3 NSS

Summary coverage rate:

lines.....: 47.6% (58670 of 123333 lines)

functions..: 58.0% (4144 of 7143 functions)

branches...: no data found

	Lines	Functions	Branches
Filename	Rate Num	Rate Num	Rate Num

=====

[nss-3.19.2/nss/]

coreconf/nsinstall/nsinstall.c	47.2% 214 40.0% 5 - 0
coreconf/nsinstall/pathsub.c	29.4% 136 50.0% 10 - 0
lib/base/arena.c	58.0% 300 72.7% 22 - 0
lib/base/error.c	83.1% 59 100% 7 - 0
lib/base/hash.c	75.3% 93 80.0% 15 - 0
lib/base/hashops.c	100% 13 100% 4 - 0
lib/base/item.c	52.8% 53 60.0% 5 - 0
lib/base/libc.c	66.7% 15 75.0% 4 - 0
lib/base/list.c	83.8% 198 85.7% 21 - 0
lib/base/tracker.c	56.7% 97 100% 7 - 0
lib/base/utf8.c	26.4% 182 55.6% 9 - 0
lib/certdb/alg1485.c	57.3% 696 65.8% 38 - 0
lib/certdb/certdb.c	51.7% 1413 66.3% 95 - 0

lib/certdb/certv3.c	65.1% 86 81.8% 11 - 0
lib/certdb/certxutl.c	84.7% 196 100% 15 - 0
lib/certdb/crl.c	60.0% 1357 82.9% 70 - 0
lib/certdb/genname.c	59.0% 764 70.5% 44 - 0
lib/certdb/polcyxtn.c	35.7% 314 52.4% 21 - 0
lib/certdb/secname.c	71.6% 359 87.0% 23 - 0
lib/certdb/stanpcertdb.c	59.9% 501 70.0% 30 - 0
lib/certdb/xauthkid.c	70.8% 48 100% 2 - 0
lib/certdb/xbsconst.c	64.2% 53 100% 2 - 0
lib/certdb/xconst.c	41.8% 91 50.0% 10 - 0
lib/certhigh/certhigh.c	13.0% 522 17.4% 23 - 0
lib/certhigh/cerhtml.c	8.6% 174 50.0% 2 - 0
lib/certhigh/certreq.c	60.5% 119 87.5% 8 - 0
lib/certhigh/certvfy.c	45.9% 761 62.5% 24 - 0
lib/certhigh/certvfypkix.c	74.1% 671 76.0% 25 - 0
lib/certhigh/crlv2.c	38.8% 67 55.6% 9 - 0
lib/certhigh/ocsp.c	54.7% 2185 79.3% 116 - 0
lib/certhigh/ocspsig.c	60.9% 220 100% 7 - 0
lib/certhigh/xcrldist.c	71.6% 88 100% 2 - 0
lib/ckfw/builtins/bfind.c	75.9% 108 100% 6 - 0
lib/ckfw/builtins/binst.c	100% 14 100% 6 - 0
lib/ckfw/builtins/bobject.c	47.6% 42 37.5% 8 - 0
lib/ckfw/builtins/bsession.c	76.9% 13 100% 2 - 0
lib/ckfw/builtins/bslot.c	100% 10 100% 5 - 0
lib/ckfw/builtins/btoken.c	100% 16 100% 8 - 0
lib/ckfw/crypto.c	0.0% 59 0.0% 12 - 0
lib/ckfw/find.c	66.0% 100 62.5% 8 - 0
lib/ckfw/hash.c	61.5% 78 80.0% 10 - 0
lib/ckfw/instance.c	46.9% 324 57.1% 35 - 0
lib/ckfw/mechanism.c	0.0% 270 0.0% 31 - 0

lib/ckfw/mutex.c	53.4% 58 60.0% 10 - 0
lib/ckfw/object.c	23.9% 268 32.0% 25 - 0
lib/ckfw/session.c	13.6% 641 27.1% 48 - 0
lib/ckfw/sessobj.c	17.0% 224 26.3% 19 - 0
lib/ckfw/slot.c	59.6% 171 71.4% 21 - 0
lib/ckfw/token.c	51.7% 458 74.1% 54 - 0
lib/ckfw/wrap.c	14.2% 1933 20.6% 68 - 0
lib/crmf/asn1cmn.c	56.5% 46 100% 5 - 0
lib/crmf/challcli.c	0.0% 116 0.0% 7 - 0
lib/crmf/cmmfasn1.c	46.7% 45 60.0% 5 - 0
lib/crmf/cmmfchal.c	0.0% 150 0.0% 7 - 0
lib/crmf/cmmfrec.c	45.9% 146 42.9% 14 - 0
lib/crmf/cmmfresp.c	65.0% 137 88.9% 9 - 0
lib/crmf/crmfcont.c	42.5% 565 58.5% 41 - 0
lib/crmf/crmfdec.c	57.5% 174 76.9% 13 - 0
lib/crmf/crmfenc.c	38.5% 13 33.3% 3 - 0
lib/crmf/crmfget.c	60.8% 237 82.4% 17 - 0
lib/crmf/crmfpop.c	51.3% 279 72.2% 18 - 0
lib/crmf/crmfreq.c	63.7% 311 90.0% 30 - 0
lib/crmf/encutil.c	90.9% 11 100% 2 - 0
lib/crmf/respcli.c	25.4% 59 14.3% 7 - 0
lib/crmf/respcmn.c	55.2% 203 68.4% 19 - 0
lib/crmf/servget.c	17.4% 483 21.6% 51 - 0
lib/cryptohi/dsautil.c	74.1% 108 75.0% 8 - 0
lib/cryptohi/sechash.c	29.4% 143 34.5% 29 - 0
lib/cryptohi/seckey.c	38.0% 789 57.4% 54 - 0
lib/cryptohi/secsign.c	65.7% 201 90.0% 10 - 0
lib/cryptohi/secvfy.c	50.8% 309 71.4% 21 - 0
lib/dbm/src/db.c	20.0% 25 25.0% 4 - 0
lib/dbm/src/h_bigkey.c	0.0% 314 0.0% 9 - 0

lib/dbm/src/h_func.c	100% 16 100% 1 - 0
lib/dbm/src/h_log2.c	100% 4 100% 1 - 0
lib/dbm/src/h_page.c	60.6% 485 81.2% 16 - 0
lib/dbm/src/hash.c	70.2% 497 85.0% 20 - 0
lib/dbm/src/hash_buf.c	58.8% 131 100% 5 - 0
lib/dbm/src/mktemp.c	0.0% 44 0.0% 3 - 0
lib/dev/ckhelper.c	79.3% 261 83.3% 12 - 0
lib/dev/devslot.c	76.2% 101 80.0% 15 - 0
lib/dev/devtoken.c	64.4% 693 74.4% 43 - 0
lib/dev/devutil.c	16.2% 444 32.4% 34 - 0
lib/freebl/aeskeywrap.c	0.0% 157 0.0% 9 - 0
lib/freebl/alg2268.c	89.4% 170 91.7% 12 - 0
lib/freebl/alghmac.c	56.0% 84 85.7% 7 - 0
lib/freebl/arcfive.c	0.0% 12 0.0% 4 - 0
lib/freebl/arcfour.c	77.9% 136 87.5% 8 - 0
lib/freebl/camellia.c	92.4% 739 88.9% 18 - 0
lib/freebl/ctr.c	67.1% 85 100% 6 - 0
lib/freebl/cts.c	82.2% 90 100% 4 - 0
lib/freebl/des.c	100% 65 100% 2 - 0
lib/freebl/desblapi.c	91.2% 136 91.7% 12 - 0
lib/freebl/dh.c	0.0% 260 0.0% 6 - 0
lib/freebl/drbg.c	77.6% 299 100% 18 - 0
lib/freebl/dsa.c	80.9% 299 84.6% 13 - 0
lib/freebl/ec.c	73.6% 398 90.9% 11 - 0
lib/freebl/ecdecode.c	46.4% 220 100% 5 - 0
lib/freebl/ecl/ec_naf.c	100% 27 100% 2 - 0
lib/freebl/ecl/ecl.c	82.4% 176 87.5% 8 - 0
lib/freebl/ecl/ecl_curve.c	93.8% 48 100% 3 - 0
lib/freebl/ecl/ecl_gf.c	13.4% 486 28.0% 25 - 0
lib/freebl/ecl/ecl_mult.c	29.0% 145 50.0% 4 - 0

lib/freebl/ecl/ecp_256.c	98.1% 160 100% 4 - 0
lib/freebl/ecl/ecp_256_32.c	98.9% 696 100% 30 - 0
lib/freebl/ecl/ecp_384.c	0.0% 116 0.0% 4 - 0
lib/freebl/ecl/ecp_521.c	89.7% 58 100% 5 - 0
lib/freebl/ecl/ecp_aff.c	75.7% 115 83.3% 6 - 0
lib/freebl/ecl/ecp_jac.c	93.1% 216 100% 7 - 0
lib/freebl/ecl/ecp_jm.c	94.5% 128 100% 3 - 0
lib/freebl/ecl/ecp_mont.c	90.2% 61 100% 7 - 0
lib/freebl/gcm.c	71.5% 260 100% 15 - 0
lib/freebl/genload.c	97.6% 41 100% 3 - 0
lib/freebl/hmacct.c	0.0% 106 0.0% 5 - 0
lib/freebl/jpake.c	0.0% 282 0.0% 7 - 0
lib/freebl/ldvector.c	100% 4 100% 1 - 0
lib/freebl/loader.c	50.9% 872 59.4% 212 - 0
lib/freebl/md2.c	78.1% 137 63.6% 11 - 0
lib/freebl/md5.c	83.1% 160 57.1% 14 - 0
lib/freebl/mpi/mp_gf2m.c	51.0% 255 50.0% 14 - 0
lib/freebl/mpi/mpcpucache.c	74.2% 97 90.0% 10 - 0
lib/freebl/mpi/mpi.c	59.2% 1616 70.9% 103 - 0
lib/freebl/mpi/mplogic.c	39.7% 141 46.2% 13 - 0
lib/freebl/mpi/mpmontg.c	62.7% 343 77.8% 9 - 0
lib/freebl/mpi/mpprime.c	53.6% 220 41.7% 12 - 0
lib/freebl/nsslowhash.c	28.7% 122 53.3% 15 - 0
lib/freebl/pqg.c	46.4% 799 73.3% 30 - 0
lib/freebl/rawhash.c	15.0% 20 14.3% 7 - 0
lib/freebl/rijndael.c	80.3% 365 83.3% 18 - 0
lib/freebl/rsa.c	59.5% 751 83.3% 24 - 0
lib/freebl/rsapkcs.c	61.0% 584 80.0% 25 - 0
lib/freebl/seed.c	78.8% 212 90.9% 11 - 0
lib/freebl/sha512.c	81.0% 457 56.4% 55 - 0

lib/freebl/sha_fast.c	87.3% 173 57.1% 14 - 0
lib/freebl/sha_fast.h	100% 5 100% 2 - 0
lib/freebl/shvfy.c	81.5% 124 100% 6 - 0
lib/freebl/stubs.c	56.6% 219 85.0% 40 - 0
lib/freebl/sysrand.c	0.0% 9 0.0% 1 - 0
lib/freebl/tlsprfalg.c	90.3% 72 100% 2 - 0
lib/freebl/unix_rand.c	40.5% 205 57.1% 14 - 0
lib/jar/jar-ds.c	100% 16 100% 4 - 0
lib/jar/jar.c	57.4% 329 68.8% 16 - 0
lib/jar/jarfile.c	59.0% 400 77.8% 18 - 0
lib/jar/jarint.c	87.5% 16 100% 1 - 0
lib/jar/jarsign.c	45.8% 96 33.3% 6 - 0
lib/jar/jarver.c	69.3% 449 86.4% 22 - 0
lib/libpkix/pkix/certsel/pkix_certselector.c	72.1% 419 90.9% 22 - 0
lib/libpkix/pkix/certs...kix_comcertselparams.c	64.6% 359 70.5% 44 - 0
lib/libpkix/pkix/check...icconstraintschecker.c	15.9% 82 20.0% 5 - 0
lib/libpkix/pkix/check...kix_certchainchecker.c	62.6% 91 60.0% 10 - 0
lib/libpkix/pkix/checker/pkix_crlchecker.c	92.3% 117 100% 5 - 0
lib/libpkix/pkix/checker/pkix_ekuchecker.c	8.1% 86 20.0% 5 - 0
lib/libpkix/pkix/check...ix_expirationchecker.c	90.5% 21 100% 2 - 0
lib/libpkix/pkix/check..._namechainingchecker.c	0.0% 25 0.0% 2 - 0
lib/libpkix/pkix/check...meconstraintschecker.c	94.5% 73 100% 5 - 0
lib/libpkix/pkix/checker/pkix_ocspchecker.c	90.7% 129 100% 6 - 0
lib/libpkix/pkix/checker/pkix_policychecker.c	80.2% 727 94.4% 18 - 0
lib/libpkix/pkix/check...ix_revocationchecker.c	84.5% 181 85.7% 7 - 0
lib/libpkix/pkix/check...kix_revocationmethod.c	53.3% 15 50.0% 2 - 0
lib/libpkix/pkix/check...kix_signaturechecker.c	94.3% 105 100% 5 - 0
lib/libpkix/pkix/check...ix_targetcertchecker.c	80.9% 141 100% 5 - 0
lib/libpkix/pkix/crlsel/pkix_comcrlselparams.c	26.9% 245 27.3% 22 - 0
lib/libpkix/pkix/crlsel/pkix_crlselector.c	26.5% 260 33.3% 15 - 0

lib/libpkix/pkix/params/pkix_procparams.c	50.5%	438 68.3%	41	-	0
lib/libpkix/pkix/params/pkix_resourcelimits.c	9.7%	134 6.2%	16	-	0
lib/libpkix/pkix/params/pkix_trustanchor.c	46.6%	148 54.5%	11	-	0
lib/libpkix/pkix/params/pkix_valparams.c	13.1%	99 12.5%	8	-	0
lib/libpkix/pkix/results/pkix_buildresult.c	46.6%	103 62.5%	8	-	0
lib/libpkix/pkix/results/pkix_policynode.c	35.0%	340 45.5%	22	-	0
lib/libpkix/pkix/results/pkix_valresult.c	31.9%	138 44.4%	9	-	0
lib/libpkix/pkix/results/pkix_verifynode.c	32.6%	298 41.2%	17	-	0
lib/libpkix/pkix/store/pkix_store.c	86.4%	132 80.0%	15	-	0
lib/libpkix/pkix/top/pkix_build.c	77.6%	1297 90.9%	22	-	0
lib/libpkix/pkix/top/pkix_lifecycle.c	72.2%	54 100%	2	-	0
lib/libpkix/pkix/top/pkix_validate.c	29.2%	438 44.4%	9	-	0
lib/libpkix/pkix/util/pkix_error.c	25.9%	166 27.3%	11	-	0
lib/libpkix/pkix/util/pkix_errpaths.c	34.8%	46 66.7%	3	-	0
lib/libpkix/pkix/util/pkix_list.c	74.7%	525 85.7%	28	-	0
lib/libpkix/pkix/util/pkix_logger.c	3.9%	335 5.6%	18	-	0
lib/libpkix/pkix/util/pkix_tools.c	69.8%	295 64.7%	17	-	0
lib/libpkix/pkix_pl_nss/module/pkix_pl_aiaMgr.c	60.8%	250 55.6%	9	-	0
lib/libpkix/pkix_pl_ns...pkix_pl_colcertstore.c	3.6%	361 5.9%	17	-	0
lib/libpkix/pkix_pl_ns...kix_pl_httpcertstore.c	22.9%	345 43.8%	16	-	0
lib/libpkix/pkix_pl_ns...pl_httpdefaultclient.c	69.7%	561 69.2%	26	-	0
lib/libpkix/pkix_pl_ns...kix_pl_ldapcertstore.c	0.0%	350 0.0%	10	-	0
lib/libpkix/pkix_pl_ns...pl_ldapdefaultclient.c	2.0%	644 3.2%	31	-	0
lib/libpkix/pkix_pl_ns.../pkix_pl_ldaprequest.c	5.6%	231 11.1%	9	-	0
lib/libpkix/pkix_pl_ns...pkix_pl_ldapresponse.c	6.0%	217 7.7%	13	-	0
lib/libpkix/pkix_pl_ns...kix_pl_ldaptemplates.c	0.0%	1 0.0%	1	-	0
lib/libpkix/pkix_pl_ns...e/pkix_pl_nsscontext.c	52.5%	80 40.0%	10	-	0
lib/libpkix/pkix_pl_ns...kix_pl_pk11certstore.c	82.1%	442 100%	10	-	0
lib/libpkix/pkix_pl_nss/module/pkix_pl_socket.c	25.5%	502 38.5%	26	-	0
lib/libpkix/pkix_pl_ns..._pl_basicconstraints.c	34.7%	118 62.5%	8	-	0

lib/libpkix/pkix_pl_nss/pki/pkix_pl_cert.c	70.0%	1315 83.9%	56	-	0
lib/libpkix/pkix_pl_ns...ix_pl_certpolicyinfo.c	45.6%	103 62.5%	8	-	0
lib/libpkix/pkix_pl_ns...kix_pl_certpolicymap.c	42.7%	110 55.6%	9	-	0
lib/libpkix/pkix_pl_ns..._certpolicyqualifier.c	33.3%	105 37.5%	8	-	0
lib/libpkix/pkix_pl_nss/pki/pkix_pl_crl.c	15.1%	370 22.2%	18	-	0
lib/libpkix/pkix_pl_nss/pki/pkix_pl_crl_dp.c	57.4%	61 100%	3	-	0
lib/libpkix/pkix_pl_nss/pki/pkix_pl_crlentry.c	4.8%	271 8.3%	12	-	0
lib/libpkix/pkix_pl_nss/pki/pkix_pl_date.c	54.4%	125 66.7%	12	-	0
lib/libpkix/pkix_pl_ns.../pkix_pl_generalname.c	46.9%	213 70.0%	10	-	0
lib/libpkix/pkix_pl_ns...i/pkix_pl_infoaccess.c	28.9%	298 50.0%	14	-	0
lib/libpkix/pkix_pl_ns...x_pl_nameconstraints.c	33.3%	387 40.0%	15	-	0
lib/libpkix/pkix_pl_ns...i/pkix_pl_ocspcertid.c	94.8%	58 100%	5	-	0
lib/libpkix/pkix_pl_ns.../pkix_pl_ocsprequest.c	64.1%	142 71.4%	7	-	0
lib/libpkix/pkix_pl_ns...pkix_pl_ocspreponse.c	69.3%	384 72.7%	11	-	0
lib/libpkix/pkix_pl_nss/pki/pkix_pl_publickey.c	66.9%	151 77.8%	9	-	0
lib/libpkix/pkix_pl_nss/pki/pkix_pl_x500name.c	62.7%	142 58.3%	12	-	0
lib/libpkix/pkix_pl_nss/system/pkix_pl_bigint.c	10.5%	124 12.5%	8	-	0
lib/libpkix/pkix_pl_ns...em/pkix_pl_bytearray.c	52.4%	164 70.0%	10	-	0
lib/libpkix/pkix_pl_nss/system/pkix_pl_common.c	23.2%	319 45.5%	11	-	0
lib/libpkix/pkix_pl_nss/system/pkix_pl_error.c	100%	2 100%	1	-	0
lib/libpkix/pkix_pl_ns...em/pkix_pl_hashtable.c	78.5%	93 83.3%	6	-	0
lib/libpkix/pkix_pl_ns...em/pkix_pl_lifecycle.c	90.2%	112 100%	4	-	0
lib/libpkix/pkix_pl_nss/system/pkix_pl_mem.c	76.4%	55 100%	4	-	0
lib/libpkix/pkix_pl_ns.../pkix_pl_monitorlock.c	72.1%	43 60.0%	5	-	0
lib/libpkix/pkix_pl_nss/system/pkix_pl_mutex.c	93.3%	45 100%	5	-	0
lib/libpkix/pkix_pl_nss/system/pkix_pl_object.c	71.6%	331 80.0%	20	-	0
lib/libpkix/pkix_pl_nss/system/pkix_pl_oid.c	79.4%	102 77.8%	9	-	0
lib/libpkix/pkix_pl_ns...tem/pkix_pl_primhash.c	60.8%	143 62.5%	8	-	0
lib/libpkix/pkix_pl_nss/system/pkix_pl_rwlock.c	17.8%	73 11.1%	9	-	0
lib/libpkix/pkix_pl_nss/system/pkix_pl_string.c	52.6%	213 50.0%	10	-	0

lib/nss/nssinit.c	46.0% 569 60.7% 28 - 0
lib/nss/utilwrap.c	0.0% 248 0.0% 108 - 0
lib/pk11wrap/debug_module.c	0.0% 1473 0.0% 83 - 0
lib/pk11wrap/dev3hack.c	71.3% 122 78.6% 14 - 0
lib/pk11wrap/pk11akey.c	51.4% 1109 58.5% 41 - 0
lib/pk11wrap/pk11auth.c	52.2% 345 53.6% 28 - 0
lib/pk11wrap/pk11cert.c	58.7% 1263 67.7% 62 - 0
lib/pk11wrap/pk11cxt.c	37.7% 470 56.0% 25 - 0
lib/pk11wrap/pk11err.c	15.9% 69 100% 1 - 0
lib/pk11wrap/pk11kea.c	80.4% 56 100% 2 - 0
lib/pk11wrap/pk11list.c	100% 27 100% 8 - 0
lib/pk11wrap/pk11load.c	55.1% 276 46.2% 13 - 0
lib/pk11wrap/pk11mech.c	26.7% 572 66.7% 27 - 0
lib/pk11wrap/pk11merge.c	73.7% 579 95.5% 22 - 0
lib/pk11wrap/pk11nobj.c	76.4% 369 100% 12 - 0
lib/pk11wrap/pk11obj.c	48.0% 989 56.4% 55 - 0
lib/pk11wrap/pk11pars.c	37.0% 514 42.3% 26 - 0
lib/pk11wrap/pk11pbe.c	66.3% 584 72.2% 36 - 0
lib/pk11wrap/pk11pk12.c	0.0% 277 0.0% 15 - 0
lib/pk11wrap/pk11pqg.c	0.0% 226 0.0% 14 - 0
lib/pk11wrap/pk11sdr.c	96.7% 153 100% 7 - 0
lib/pk11wrap/pk11skey.c	33.0% 1271 46.5% 71 - 0
lib/pk11wrap/pk11slot.c	62.6% 1085 74.7% 91 - 0
lib/pk11wrap/pk11util.c	51.1% 689 72.0% 50 - 0
lib/pkcs12/p12creat.c	0.0% 101 0.0% 6 - 0
lib/pkcs12/p12d.c	47.7% 1584 76.5% 68 - 0
lib/pkcs12/p12dec.c	0.0% 267 0.0% 13 - 0
lib/pkcs12/p12e.c	58.8% 859 86.7% 30 - 0
lib/pkcs12/p12local.c	10.5% 448 22.2% 18 - 0
lib/pkcs12/p12plcy.c	72.7% 44 75.0% 4 - 0

lib/pkcs12/p12tmpl.c	44.3% 70 75.0% 4 - 0
lib/pkcs7/certread.c	40.0% 200 75.0% 8 - 0
lib/pkcs7/p7common.c	23.0% 309 50.0% 10 - 0
lib/pkcs7/p7create.c	58.2% 479 76.2% 21 - 0
lib/pkcs7/p7decode.c	57.9% 646 72.0% 25 - 0
lib/pkcs7/p7encode.c	70.7% 440 92.3% 13 - 0
lib/pkcs7/p7local.c	73.8% 386 100% 15 - 0
lib/pkcs7/secmime.c	0.0% 268 0.0% 14 - 0
lib/pki/asymmkey.c	0.0% 102 0.0% 34 - 0
lib/pki/certdecode.c	35.7% 14 50.0% 2 - 0
lib/pki/certificate.c	54.3% 411 38.6% 57 - 0
lib/pki/cryptocontext.c	25.7% 268 16.4% 67 - 0
lib/pki/pki3hack.c	60.2% 731 73.3% 45 - 0
lib/pki/pkibase.c	73.1% 557 89.4% 47 - 0
lib/pki/pkistore.c	69.1% 291 82.1% 28 - 0
lib/pki/symmkey.c	0.0% 63 0.0% 21 - 0
lib/pki/tdcache.c	65.2% 468 80.0% 30 - 0
lib/pki/trustdomain.c	49.9% 479 32.8% 61 - 0
lib/smime/cmsarray.c	87.5% 56 80.0% 5 - 0
lib/smime/cmsasn1.c	59.1% 22 33.3% 3 - 0
lib/smime/cmsattr.c	64.8% 145 90.9% 11 - 0
lib/smime/cmscinfo.c	72.0% 150 86.4% 22 - 0
lib/smime/cmscipher.c	70.7% 232 100% 7 - 0
lib/smime/cmsdecode.c	56.7% 282 83.3% 12 - 0
lib/smime/cmsdigdata.c	0.0% 60 0.0% 9 - 0
lib/smime/cmsdigest.c	69.0% 84 66.7% 6 - 0
lib/smime/cmsencdata.c	77.8% 90 100% 9 - 0
lib/smime/cmsencode.c	53.6% 278 80.0% 10 - 0
lib/smime/cmsenvdata.c	73.7% 152 100% 10 - 0
lib/smime/cmsmessage.c	50.5% 99 53.8% 13 - 0

lib/smime/cmspubkey.c	65.0% 40 60.0% 5 - 0
lib/smime/cmsrecinfo.c	27.8% 316 46.7% 15 - 0
lib/smime/cmsreclist.c	55.6% 81 75.0% 4 - 0
lib/smime/cmssigdata.c	60.2% 470 86.2% 29 - 0
lib/smime/cmssiginfo.c	44.4% 387 68.0% 25 - 0
lib/smime/cmsudf.c	12.4% 178 22.2% 18 - 0
lib/smime/cmsutil.c	56.3% 142 81.8% 11 - 0
lib/smime/smimeutil.c	32.3% 251 35.3% 17 - 0
lib/softoken/fipsaudt.c	0.0% 100 0.0% 15 - 0
lib/softoken/fipstest.c	78.4% 459 100% 18 - 0
lib/softoken/fipstokn.c	75.1% 563 93.3% 75 - 0
lib/softoken/jpakesftk.c	0.0% 146 0.0% 8 - 0
lib/softoken/legacydb/dbmshim.c	40.0% 260 45.5% 22 - 0
lib/softoken/legacydb/keydb.c	65.9% 973 90.0% 50 - 0
lib/softoken/legacydb/lgattr.c	81.1% 815 95.3% 43 - 0
lib/softoken/legacydb/lgcreate.c	76.4% 475 100% 13 - 0
lib/softoken/legacydb/lgdestroy.c	73.8% 61 100% 1 - 0
lib/softoken/legacydb/lgfind.c	82.6% 432 100% 18 - 0
lib/softoken/legacydb/lginit.c	71.1% 311 85.7% 28 - 0
lib/softoken/legacydb/lgutl.c	88.5% 165 100% 20 - 0
lib/softoken/legacydb/lowcert.c	78.5% 358 94.4% 18 - 0
lib/softoken/legacydb/lowkey.c	74.6% 138 90.9% 11 - 0
lib/softoken/legacydb/pcertdb.c	65.4% 2513 90.2% 133 - 0
lib/softoken/legacydb/pk11db.c	81.3% 347 100% 12 - 0
lib/softoken/lgglue.c	64.1% 198 91.7% 12 - 0
lib/softoken/lowkey.c	20.4% 221 69.2% 13 - 0
lib/softoken/lowpbe.c	66.9% 640 85.0% 20 - 0
lib/softoken/padbuf.c	76.9% 13 100% 1 - 0
lib/softoken/pkcs11.c	78.1% 2131 96.3% 82 - 0
lib/softoken/pkcs11c.c	50.4% 3697 77.0% 113 - 0

lib/softoken/pkcs11u.c	74.3% 873 83.1% 71 - 0
lib/softoken/sdb.c	73.8% 906 85.7% 35 - 0
lib/softoken/sftkdb.c	78.1% 1137 89.1% 55 - 0
lib/softoken/sftkhmac.c	0.0% 97 0.0% 8 - 0
lib/softoken/sftkpars.c	91.6% 166 83.3% 6 - 0
lib/softoken/sftkpwd.c	81.7% 503 100% 23 - 0
lib/softoken/tlsprf.c	0.0% 88 0.0% 7 - 0
lib/ssl/authcert.c	0.0% 36 0.0% 1 - 0
lib/ssl/cmpcert.c	0.0% 34 0.0% 1 - 0
lib/ssl/derive.c	0.0% 400 0.0% 4 - 0
lib/ssl/dtlscon.c	0.0% 405 0.0% 26 - 0
lib/ssl/prelib.c	0.0% 6 0.0% 2 - 0
lib/ssl/ssl3con.c	0.0% 5666 0.0% 158 - 0
lib/ssl/ssl3ecc.c	0.0% 483 0.0% 24 - 0
lib/ssl/ssl3ext.c	0.0% 1167 0.0% 46 - 0
lib/ssl/ssl3gthr.c	0.0% 175 0.0% 4 - 0
lib/ssl/sslauth.c	0.0% 136 0.0% 8 - 0
lib/ssl/sslcon.c	0.0% 1670 0.0% 49 - 0
lib/ssl/ssldef.c	0.0% 82 0.0% 11 - 0
lib/ssl/sslenum.c	0.0% 4 0.0% 2 - 0
lib/ssl/sslerr.c	0.0% 6 0.0% 1 - 0
lib/ssl/sslerrstrs.c	0.0% 4 0.0% 2 - 0
lib/ssl/sslgthr.c	0.0% 168 0.0% 5 - 0
lib/ssl/sslinfo.c	0.0% 135 0.0% 8 - 0
lib/ssl/sslinit.c	0.0% 8 0.0% 1 - 0
lib/ssl/sslmutex.c	0.0% 109 0.0% 9 - 0
lib/ssl/sslnonce.c	0.0% 218 0.0% 20 - 0
lib/ssl/sslreveal.c	0.0% 37 0.0% 4 - 0
lib/ssl/sslsecur.c	0.0% 742 0.0% 47 - 0
lib/ssl/sslsnce.c	0.0% 863 0.0% 40 - 0

lib/ssl/sslsock.c	0.5% 1533 1.2% 85 - 0
lib/ssl/ssltrace.c	0.0% 123 0.0% 6 - 0
lib/ssl/unix_err.c	0.0% 267 0.0% 41 - 0
lib/util/derdec.c	0.0% 78 0.0% 3 - 0
lib/util/derenc.c	54.9% 195 100% 4 - 0
lib/util/dersubr.c	58.7% 126 83.3% 6 - 0
lib/util/dertime.c	71.8% 142 62.5% 8 - 0
lib/util/errstrs.c	100% 4 100% 2 - 0
lib/util/nssb64d.c	69.4% 255 94.4% 18 - 0
lib/util/nssb64e.c	52.7% 241 57.1% 14 - 0
lib/util/nssrwlk.c	76.8% 95 85.7% 7 - 0
lib/util/oidstring.c	81.2% 64 100% 1 - 0
lib/util/pkcs1sig.c	76.1% 71 100% 2 - 0
lib/util/portreg.c	14.3% 196 55.6% 9 - 0
lib/util/quickder.c	84.7% 360 100% 13 - 0
lib/util/secalgid.c	88.6% 44 100% 5 - 0
lib/util/secasn1d.c	75.8% 1099 78.2% 55 - 0
lib/util/secasn1e.c	91.0% 601 97.3% 37 - 0
lib/util/secasn1u.c	86.7% 30 100% 3 - 0
lib/util/secdig.c	48.6% 72 50.0% 6 - 0
lib/util/secitem.c	50.0% 238 64.7% 17 - 0
lib/util/secload.c	40.3% 62 66.7% 3 - 0
lib/util/secoid.c	72.3% 242 80.0% 20 - 0
lib/util/secport.c	71.3% 265 82.8% 29 - 0
lib/util/sectime.c	17.3% 52 20.0% 10 - 0
lib/util/templates.c	0.0% 17 0.0% 17 - 0
lib/util/utf8.c	28.6% 196 75.0% 4 - 0
lib/util/utilmod.c	85.2% 332 90.9% 11 - 0
lib/util/utilpars.c	80.7% 529 88.1% 42 - 0

=====

3.3 Fehler in Zertifikatsvalidierung

Der Test auf Fehler bei der Zertifikatsvalidierung wurde mit dem Tool x509test [x509test] durchgeführt. Die Ausgaben des Tools sind im folgenden dargestellt.

3.3.1 Botan

Starting SSL/TLS X509 Certificate Test

Checking Root CA...

Checking Test Directory...

Building X509 Test Cases...

Starting Network Server...

Server Ready!

Passed positive test: ValidCert

Passed positive test: ValidChained

Passed positive test: ValidWildcard

Passed positive test: ValidAltName

FAILED positive test: ValidNameConstraint

- failed a dependency test, excluding descendant test cases...

- > ValidNameConstraint

- > InvalidNameConstraintExclude

- > InvalidNameConstraintPermit

- > InvalidNameConstraintPermitRight

- > InvalidNameConstraintPermitThenExclude

Passed negative test: InvalidSelfSign

Passed negative test: UnknownCriticalExtension

Passed negative test: UnknownCriticalExtensionChained

Passed negative test: InvalidNameNull

Passed negative test: InvalidNotBefore

Passed negative test: InvalidNotAfter

Passed negative test: InvalidIntegrity

Passed negative test: InvalidName

Passed negative test: InvalidNotBeforeChained

Passed negative test: InvalidNotAfterChained

Passed negative test: InvalidIntegrityChained

Passed negative test: InvalidNameAltName

FAILED negative test: InvalidExtendedKeyUsage

- severity: Low; ease: Low; see RFC5280 4.2.1.12

Passed positive test: UnknownNonCriticalExtension

FAILED negative test: InvalidKeyUsage

- severity: Low; ease: Low; see RFC5280 4.2.1.3

Passed positive test: UnknownNonCriticalExtensionChained

Passed negative test: InvalidIntCASelfSign

Passed negative test: MissingIntCABasicConstraintWithCertSign

Passed negative test: InvalidIntCAVersionOne

Passed negative test: InvalidIntCALoop

Passed negative test: InvalidIntCALen

Passed negative test: InvalidIntCAFlag

Passed negative test: InvalidIntCAKeyUsage

Passed negative test: InvalidIntCAVersionTwo

Passed negative test: MissingIntCAExtensions

Passed positive test: ValidIntCALen

Passed negative test: InvalidWildcardSingle

FAILED negative test: InvalidWildcardLeft

- severity: Medium; ease: High; see RFC6125 6.4.3

Passed negative test: InvalidWildcardMidMixed

Passed negative test: InvalidWildcardMid

Passed negative test: InvalidWildcardAll

Passed negative test: InvalidWildcardSingleAltName

FAILED negative test: InvalidWildcardLeftAltName

- severity: Medium; ease: High; see RFC6125 6.4.3

Passed negative test: InvalidWildcardMidMixedAltName

Passed negative test: InvalidWildcardMidAltName

Passed negative test: InvalidWildcardAllAltName

Passed negative test: InvalidNameNullAltName

Passed negative test: InvalidNameNullAltNameAndSubj

FAILED negative test: InvalidNameAltNameWithSubj

- severity: Low; ease: Low; see RFC6125 6.4.4

FAILED negative test: InvalidNameNullAltNameWithSubj

- severity: Low; ease: Medium; see RFC6125 6.4.4

Your SSL/TLS client did not pass 7 test(s) out of 45 total test(s).

Program Exits Correctly

3.3.2 LibreSSL

Starting SSL/TLS X509 Certificate Test

Checking Root CA...

Checking Test Directory...

Building X509 Test Cases...

Starting Network Server...

Server Ready!

Passed positive test: ValidCert

Passed positive test: ValidChained

Passed positive test: ValidWildcard

Passed positive test: ValidAltName

FAILED positive test: ValidNameConstraint

- failed a dependency test, excluding descendant test cases...

> ValidNameConstraint

> InvalidNameConstraintExclude

> InvalidNameConstraintPermitThenExclude

> InvalidNameConstraintPermit

> InvalidNameConstraintPermitRight

Passed negative test: UnknownCriticalExtension

Passed negative test: InvalidSelfSign

Passed negative test: UnknownCriticalExtensionChained

FAILED negative test: InvalidNameNull

- severity: Medium; ease: Medium; see RFC6125 6.4

FAILED negative test: InvalidName

- severity: Medium; ease: High; see RFC6125 6.4

Passed negative test: InvalidNotAfter

Passed negative test: InvalidNotBefore

Passed negative test: InvalidIntegrity

FAILED negative test: InvalidNameAltName

- severity: Medium; ease: High; see RFC6125 6.4

Passed negative test: InvalidNotAfterChained

Passed negative test: InvalidNotBeforeChained

Passed negative test: InvalidIntegrityChained

Passed negative test: InvalidKeyUsage

Passed positive test: UnknownNonCriticalExtension

Passed negative test: InvalidExtendedKeyUsage

Passed positive test: UnknownNonCriticalExtensionChained

Passed negative test: InvalidIntCASelfSign

Passed negative test: InvalidIntCAVersionOne

Passed negative test: InvalidIntCAVersionTwo

FAILED negative test: MissingIntCABasicConstraintWithCertSign

- severity: High; ease: Low; see RFC5280 4.2.1.3

Passed negative test: InvalidIntCAKeyUsage

Passed negative test: InvalidIntCALen

Passed negative test: MissingIntCAExtensions

Passed negative test: InvalidIntCAFlag

Passed negative test: InvalidIntCALoop

Passed positive test: ValidIntCALen

FAILED negative test: InvalidWildcardLeft

- severity: Medium; ease: High; see RFC6125 6.4.3

FAILED negative test: InvalidWildcardMid

- severity: Medium; ease: Medium; see RFC6125 6.4.3

FAILED negative test: InvalidWildcardSingle

- severity: Medium; ease: Medium; see RFC6125 6.4.3

FAILED negative test: InvalidWildcardMidMixed

- severity: Medium; ease: Medium; see RFC6125 6.4.3

FAILED negative test: InvalidWildcardAll

- severity: Medium; ease: Medium; see RFC6125 6.4.3

FAILED negative test: InvalidWildcardLeftAltName

- severity: Medium; ease: High; see RFC6125 6.4.3

FAILED negative test: InvalidWildcardMidAltName

- severity: Medium; ease: Medium; see RFC6125 6.4.3

FAILED negative test: InvalidWildcardSingleAltName

- severity: Medium; ease: Medium; see RFC6125 6.4.3

FAILED negative test: InvalidWildcardMidMixedAltName

- severity: Medium; ease: Medium; see RFC6125 6.4.3

FAILED negative test: InvalidWildcardAllAltName

- severity: Medium; ease: Medium; see RFC6125 6.4.3

FAILED negative test: InvalidNameNullAltName

- severity: Medium; ease: Medium; see RFC6125 6.4.4

FAILED negative test: InvalidNameNullAltNameAndSubj

- severity: Medium; ease: Medium; see RFC6125 6.4.4

FAILED negative test: InvalidNameAltNameWithSubj

- severity: Low; ease: Low; see RFC6125 6.4.4

FAILED negative test: InvalidNameNullAltNameWithSubj

- severity: Low; ease: Medium; see RFC6125 6.4.4

Your SSL/TLS client did not pass 19 test(s) out of 45 total test(s).

Program Exits Correctly

3.3.3 NSS

Starting SSL/TLS X509 Certificate Test

Checking Root CA...

Checking Test Directory...

Building X509 Test Cases...

Starting Network Server...

Server Ready!

Passed positive test: ValidCert

Passed positive test: ValidChained

Passed positive test: ValidWildcard

Passed positive test: ValidAltName

Passed positive test: ValidNameConstraint

Passed negative test: UnknownCriticalExtension

Passed negative test: InvalidSelfSign

Passed negative test: UnknownCriticalExtensionChained

Passed negative test: InvalidNotAfter

Passed negative test: InvalidNameNull

Passed negative test: InvalidIntegrity

Passed negative test: InvalidName

Passed negative test: InvalidNotBefore

Passed negative test: InvalidNotAfterChained

Passed negative test: InvalidIntegrityChained

Passed negative test: InvalidNameAltName

Passed negative test: InvalidNotBeforeChained

Passed positive test: UnknownNonCriticalExtension

Passed negative test: InvalidKeyUsage

Passed negative test: InvalidExtendedKeyUsage

Passed positive test: UnknownNonCriticalExtensionChained
Passed negative test: InvalidIntCAFlag
Passed negative test: MissingIntCAExtensions
Passed negative test: InvalidIntCAVersionTwo
Passed negative test: InvalidIntCAVersionOne
Passed negative test: InvalidIntCALoop
Passed negative test: MissingIntCABasicConstraintWithCertSign
Passed negative test: InvalidIntCAKeyUsage
Passed negative test: InvalidIntCALen
Passed negative test: InvalidIntCASelfSign
Passed positive test: ValidIntCALen
Passed negative test: InvalidWildcardLeft
Passed negative test: InvalidWildcardMid
Passed negative test: InvalidWildcardSingle
Passed negative test: InvalidWildcardAll
Passed negative test: InvalidWildcardMidMixed
Passed negative test: InvalidWildcardLeftAltName
Passed negative test: InvalidWildcardMidAltName
Passed negative test: InvalidWildcardSingleAltName
Passed negative test: InvalidWildcardAllAltName
Passed negative test: InvalidWildcardMidMixedAltName
Passed negative test: InvalidNameNullAltName
Passed negative test: InvalidNameNullAltNameAndSubj
Passed negative test: InvalidNameAltNameWithSubj
Passed negative test: InvalidNameNullAltNameWithSubj
Passed negative test: InvalidNameConstraintExclude
Passed negative test: InvalidNameConstraintPermitThenExclude
Passed negative test: InvalidNameConstraintPermit
Passed negative test: InvalidNameConstraintPermitRight
Congratulations! Your SSL/TLS client passed all 49 test(s).

Program Exits Correctly

3.4 Fehler in der TLS-Protokollvalidierung

Die folgenden State-Diagramme zeigen die TLS State Machines der Bibliotheken.

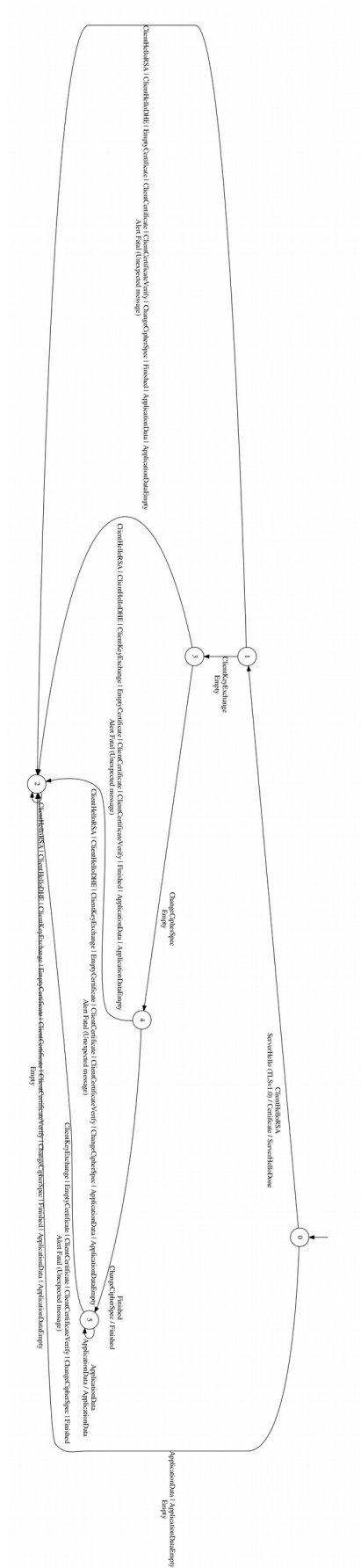


Abbildung 1: TLS State Machine der Botan-Bibliothek

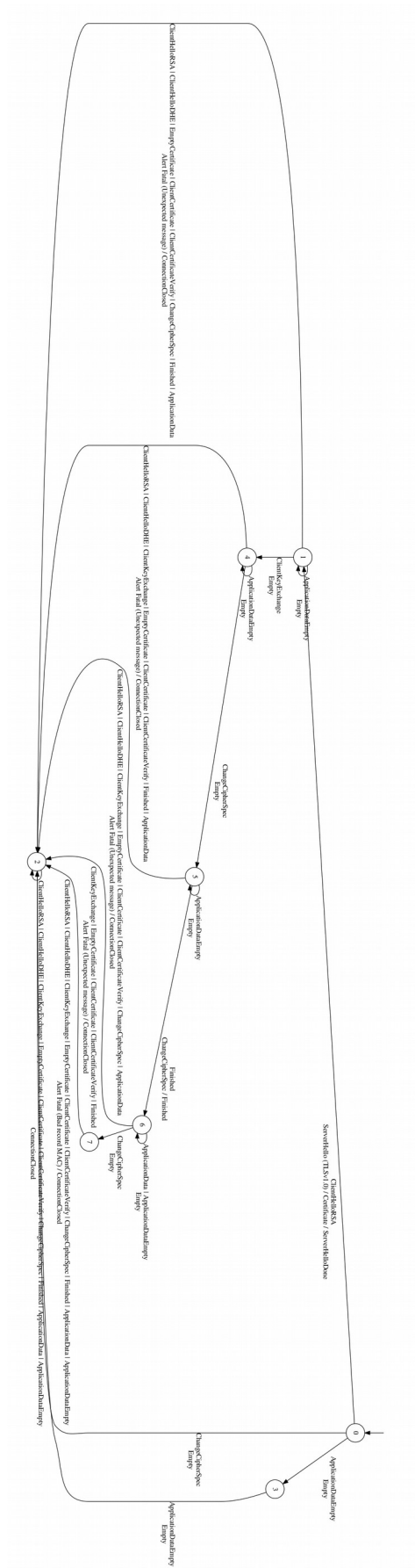
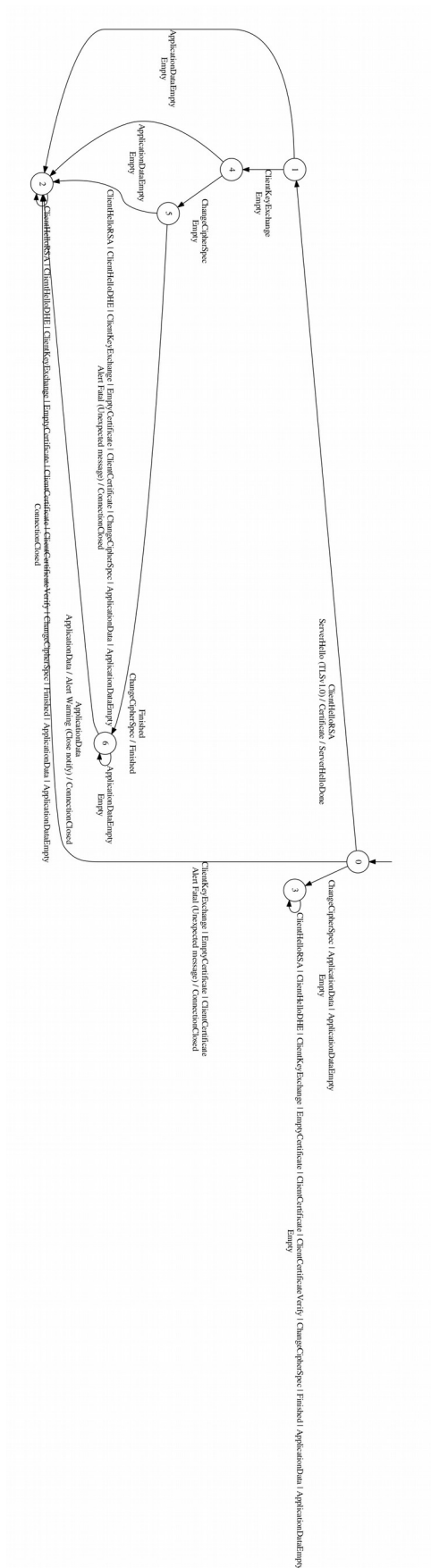


Abbildung 2: TLS State Machine der LibreSSL-Bibliothek



Bibliographie

- [OpenSSLA] W. Meyer zu Bergsten, R. Korthaus, J. Somorowsky, C. Mainka, Prof. Dr. J. Schwenk: „Quellcode-basierte Untersuchung von kryptographisch relevanten Aspekten der OpenSSL-Bibliothek“, Juni 2015
- [apachehttpd] Apache httpd. <http://httpd.apache.org/>
- [ASK05] Onur Acıçmez, Werner Schindler, Çetin K. Koç: „Improving Brumley and Boneh Timing Attack on Unprotected SSL Implementations“. ACM CCS 2005.
<http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=DCD55885759798FF35E87947FEEC7E58?doi=10.1.1.63.3510&rep=rep1&type=pdf>
- [BB2003] D. Boneh, D. Brumley: „Remote timing attacks are practical“, Proceedings of the 12th Usenix Security Symposium, 2003
- [BBPV2012] B. B. Brumley, M. Barbosa, D. Page, and F. Vercauteren: Practical realisation and elimination of an ECC-related software bug attack. CTRSA, 2012
- [BDFPS14] Bhargavan, Delignat-Lavaud, Fournet, Pironi, Strub: „Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS“, Security and Privacy 2014, <http://prosecco.gforge.inria.fr/personal/karthik/pubs/triple-handshakes-and-cookie-cutters-sp14.pdf>
- [BEAST] Thai Duong, Juliano Rizzo: Here Come The ☹ Ninjas.
<http://packetstormsecurity.com/files/download/105499/Beast-SSL.rar>
- [BDPLR] K. Bhargavan, A. Delignat-Lavaud, A. Pironi, A. Langley, M. Ray: „Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension“, <http://tools.ietf.org/html/draft-bhargavan-tls-session-hash-00>
- [Blei] Daniel Bleichenbacher: „Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1“, Crypto, 1998
- [BnetzA] Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen: „Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen)“, vom 13.1.2014
- [BPSSL2014] Ivan Ristić: „Bulletproof SSL and TLS“, Feisty Duck, 16. August 2014
- [BR93] Mihir Bellare and Phillip Rogaway: „Entity authentication and key distribution“, in Douglas R. Stinson, editor, Advances in Cryptology – CRYPTO’93
- [BREACH] Yoel Gluck, Neal Harris, Angelo Prado, BREACH. SSL, „Gone in 30 Seconds“, <http://breachattack.com/>, <http://breachattack.com/resources/BREACH%20-%20SSL,%20gone%20in%2030%20seconds.pdf>
- [BT2011] B. B. Brumley, N. Tuveri: „Remote Timing Attacks are Still Practical“, Cryptology ePrint Archive: Report 2011/232, Mai 2011
- [CAB] CA/Browser Forum: „Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates“, v.1.2.3
- [CODEMET] Steve Cornett, Bullseye Testing Technology: „Minimum Acceptable Code Coverage“, <http://www.bullseye.com/minimum.html>

- [COMBA] P. G. Comba: „Exponentiation cryptosystems on the IBM PC“, IBM Systems Journal, Vol. 29, No. 4, December 1990
- [CK01] R. Canetti and H. Krawczyk: „Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels“, Advances in Cryptology - EUROCRYPT '01
- [CP2005] C. Percival: „CACHE MISSING FOR FUN AND PROFIT“, Proceedings of BSDCan 2005
- [CPPBESTPR] Jason Turner, et al.: „C++ Best Practices: A Forkable Coding Standards Document“, https://github.com/lefticus/cppbestpractices/blob/master/02-Use_the_Tools_Available.md, Commit 3da497
- [CRIME-BS13] Tal Be'ery, Amichai Shulman. „A perfect CRIME? Only TIME will tell“, Blackhat 2013, <https://media.blackhat.com/eu-13/briefings/Beery/bh-eu-13-a-perfect-crime-beery-wp.pdf>
- [CRIMEimp] Adam Langley: „CRIME“, <https://www.imperialviolet.org/2012/09/21/crime.html>
- [CRIME-koto] Krzysztof Kotowicz, „If it's CRIME, then I'm guilty“, <http://blog.kotowicz.net/2012/09/if-its-crime-than-im-guilty.html>
- [EarlyCCSadv] OpenSSL Security Advisory, „Early CCS“, https://www.openssl.org/news/secadv_20140605.txt, 5. Juni 2014
- [EarlyCCSimp] Adam Langley, „Early ChangeCipherSpec Attack“, <https://www.imperialviolet.org/2014/06/05/earlyccs.html>
- [EarlyCCSLep] Masashi Kikuchi: „How I discovered CCS Injection Vulnerability (CVE-2014-0224)“, <http://ccsinjection.lepidum.co.jp/blog/2014-06-05/CCS-Injection-en/index.html>
- [FAUT] S. Schinzel, I. Schmitt: „FAU-Timer“, <https://code.google.com/p/fau-timer/>, 2013
- [FIPS186-1] U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology: „DIGITAL SIGNATURE STANDARD (DSS)“, 15. Dezember 1998
- [FLAGS] https://www.openssl.org/docs/crypto/X509_VERIFY_PARAM_set_flags.html
- [FLUSH] Y. Yarom, K. Falkner: „FLUSH+RELOAD: a high resolution, low noise, L3 cache side-channel attack“, SEC'14 Proceedings of the 23rd USENIX conference on Security Symposium
- FORK1 Andrew Ayer: „LibreSSL's PRNG is Unsafe on Linux“, https://www.agwa.name/blog/post/libressls_prng_is_unsafe_on_linux
- FORK2 Diverse Autoren, <https://news.ycombinator.com/item?id=8034045>
- [GCV] „Gcov – Using the GNU Compiler Collection (GCC)“, <https://gcc.gnu.org/onlinedocs/gcc/Gcov.html>
- [GCM] U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology: „Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC“, November 2007
- [GKS13] Florian Giesen, Florian Kohlar, and Douglas Stebila: „On the security of TLS renegotiation“, CCS '13, 2013
- [HAC] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone: „Handbook of Applied Cryptography“, Fifth Printing, 2001, ISBN 0-8493-8523-7
- [Heartbleed] „The Heartbleed Bug“, <http://heartbleed.com/>

- [HeartbleedAdv] OpenSSL Security Advisory: “Heartbleed,”
v] https://www.openssl.org/news/secadv_20140407.txt, 7. April 2014
- [HeartbleedClosud] Nick Sullivan: „Heartache and Heartbleed: The insider’s perspective on the aftermath of Heartbleed“, 2014,
<http://events.ccc.de/congress/2014/Fahrplan/events/6212.html>
- [HeartbeatRFC] R. Seggelmann, M. Tuexen, M. Williams: „Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension“, 2012
]
- [HOST] https://www.openssl.org/docs/crypto/X509_check_host.html
- [IANAEECC] Internet Assigned Numbers Authority: „Transport Layer Security (TLS) Parameters: EC Named Curve Registry“, <http://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-8>, vom 8.12.2014
- [JKSS12] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk: „On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors“, Advances in Cryptology – CRYPTO 2012
- [KKO63] A. Karatsuba, YU. Ofman, „Multiplication of multidigit numbers on automata“, Soviet Physics – Doklady, 7 (1963), 595–596.
- [KM07] B. Kopf and H. Mantel. Transformational Typing and Unification for Automatically Correcting Insecure Programs. 2007
- [KR02] Vlastimil Klíma, Tomáš Rosa: „Attack on Private Signature Keys of the OpenPGP format, PGP programs and other applications compatible with OpenPGP“, 2002,
<https://eprint.iacr.org/2002/076.pdf>
- [LMSS2014] M. Lochter, J. Merkle, J.-M. Schmidt, T. Schütze: „Requirements for Standard Elliptic Curve“, Oktober 2014, <http://eprint.iacr.org/2014/832>
- [Lucky13] N.J. AlFardan and K.G. Paterson: „Lucky Thirteen: Breaking the TLS and DTLS Record Protocols“, IEEE Security and Privacy, 2013,
<http://www.isg.rhul.ac.uk/tls/Lucky13.html>
- [Lucky13imp] Adam Langley: „Lucky Thirteen attack on TLS CBC“, 2013,
<https://www.imperialviolet.org/2013/02/04/luckythirteen.html>
- [MALSHA1] A. Albertini, J.-P. Aumasson, M. Eichlseder, F. Mendel, M. Schläffer: „Malicious Hashing: Eve’s Variant of SHA-1“, Cryptology ePrint Archive: Report 2014/694, 03. September 2014, <http://eprint.iacr.org/2014/694>
FAQ: <http://malicioussha1.github.io/#faq>
- [mitmAdv] OpenSSL Security Advisory [07-Jan-2009]: „Incorrect checks for malformed signatures“, https://www.openssl.org/news/secadv_20090107.txt
- [MS14] Daniel A. Mayer, Joel Sandin: „Time Trial - Racing Towards Practical Remote Timing Attacks“, Blackhat 2014
- [MS15] Heiko Mantel and Artem Starostin. Transforming Out Timing Leaks, More or Less. ESORICS 2015
- [MSWS] Christopher Meyer, Juraj Somorovsky, Eugen Weiss, and Jörg Schwenk, Sebastian Schinzel, Erik Tews: „Revisiting SSL/TLS Implementations: New Bleichenbacher Side Channels and Attacks“, USENIX Security, 2014

- [NCsupport] Tom Leek: „OpenSSL ECC named curve support“, 2013,
<http://security.stackexchange.com/questions/43992/openssl-ecc-named-curve-support>
- [NETSEC] John Viega, Matt Messier, Pravir Chandra: „Network Security with OpenSSL“,
Frist Edition, 2002, ISBN 0-596-00270-X
- [nginx] Nginx, <http://nginx.org/>
- [openssh] OpenSSH. <http://www.openssh.com/>
- [OpenSSLDH] OpenSSL Dokumentation zu DH.
https://www.openssl.org/docs/ssl/SSL_CTX_set_tmp_dh_callback.html
- [P1OVECT1] PKCS#1 Test Vectors, Original: <ftp://ftp.rsa.com/pkcs-1v2/p1ovect1.txt> (offline),
available at <ftp://ftp.sunet.se/pub/security/pca/docs/PKCS/ftp.rsa.com/pkcs-1v2/p1ovect1.txt>
- [PCK1996] Paul C. Kocher: „Timing attacks on implementations of Diffie-Hellman, RSA,
DSS, and other systems“, Advances in Cryptology - CRYPTO '96
- [PKCS1] B. Kaldinski, J. Staddon: „PKCS #1: RSA Cryptography Specifications“, Version
2.0, October 1998, available at <https://www.ietf.org/rfc/rfc2437.txt>
- [PKI] Carlisle Adams, Steve Lloyd: „Understanding PKI“, Second Edition, 2003, ISBN
0-672-32391-5
- [POODLE] Bodo Möller, Thai Duong, Krzysztof Kotowicz: „This POODLE bites: Exploiting
the SSLv3.0 Fallback“, 2014, <https://www.openssl.org/~bodo/ssl-poodle.pdf>
- [POODLEadv] OpenSSL Security Advisory, „POODLE und SSLv3.0 Fallback“,
https://www.openssl.org/news/secadv_20141015.txt, 15. Oktober 2014
- [RenegAdv] OpenSSL Security Advisory: „TLS Renegotiation Attack“,
https://www.openssl.org/news/secadv_20091111.txt, 11. November 2009
- [RFC2785] R. Zuccherato: Methods for Avoiding the "Small-Subgroup" Attacks on the
Diffie-Hellman Key Agreement Method for S/MIME,
<https://www.ietf.org/rfc/rfc2785.txt>
- [RFC2785] Internet Engineering Task Force: RFC2785: „Methods for Avoiding the "Small-
Subgroup" Attacks on the Diffie-Hellman Key Agreement Method for S/MIME“,
<http://tools.ietf.org/html/rfc2785>, März 2000
- [RFC3280] Internet Engineering Task Force: RFC3280: „Internet X.509 Public Key
Infrastructure Certificate and Certificate Revocation List (CRL) Profile“,
<http://tools.ietf.org/html/rfc3280>, April 2001
- [RFC3447] Internet Engineering Task Force: RFC3447: „Public-Key Cryptography Standards
(PKCS) #1: RSA Cryptography Specifications Version 2.1“,
<http://tools.ietf.org/html/rfc3447>, Februar 2003
- [RFC3779] Internet Engineering Task Force: RFC3779: „X.509 Extensions for IP Addresses
and AS Identifiers“, <http://tools.ietf.org/html/rfc3779>, Juni 2004
- [RFC4419] Internet Engineering Task Force: RFC4419: „Diffie-Hellman Group Exchange for
the Secure Shell (SSH) Transport Layer Protocol“,
<http://tools.ietf.org/html/rfc4419>, März 2006
- [RFC5246] Internet Engineering Task Force: RFC5246: „The Transport Layer Security (TLS)
Protocol Version 1.2“, <https://tools.ietf.org/html/rfc5246>, August 2008

- [RFC5639] Internet Engineering Task Force: RFC5639: „Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation“, <https://tools.ietf.org/html/rfc5639>, März 2010
- [RFC5580] Internet Engineering Task Force: RFC5580: „Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile“, <http://tools.ietf.org/html/rfc5280>, März 2008
- [RFC5746] Internet Engineering Task Force: RFC5746: „Transport Layer Security (TLS) Renegotiation Indication Extension“, <http://tools.ietf.org/html/rfc5746>, Februar 2010
- [RFC6125] Internet Engineering Task Force: RFC6125: „Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)“, <http://tools.ietf.org/html/rfc6125>, März 2011
- [RSAblAdv] OpenSSL Security Advisory [17 March 2003], RSA Blinding, https://www.openssl.org/news/secadv_20030317.txt
- [RSAblinding] RSA Blinding patch and a recent snapshot, OpenSSL mailing list, <http://comments.gmane.org/gmane.comp.encryption.openssl.devel/4919>
- [RS2000] Jean-Francois Raymond, Anton Stiglic: „Security Issues in the Diffie-Hellman Key Agreement Protocol“, Dezember 2000
- [RTOpenSSL] Issue Tracking System for OpenSSL, 2014, <http://rt.openssl.org/Ticket/Display.html?id=3180&user=guest&pass=guest>
- [RTOpenSSL] Issue Tracking System for OpenSSL, 2014, <http://rt.openssl.org/Ticket/Display.html?id=3180&user=guest&pass=guest>
- [RTSS09] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage: Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds, CCS '09
- [sloccount] David A. Wheeler: „SLOCCount“, <http://www.dwheeler.com/sloccount/>
- [SAFEEDCC] D.J. Bernstein: „SafeCurves: choosing safe curves for elliptic-curve cryptography“, <http://safecurves.cr.yp.to/>, vom 19. Januar 2014
- [SSL3] A. Frier, P. Karlton, P. Kocher: "The SSL 3.0 Protocol", Netscape Communications Corp., Nov 18, 1996.
- [sslyze] SSLyze, <https://github.com/iSECPartners/sslyze>
- [testsslsh] testssl.sh: Testing SSL/TLS, <https://testssl.sh/>
- [TIOBE] Paul Jansen, TIOBE Software: „The TIOBE Quality Indicator – a pragmatic way of measuring code quality“, Version 3.5, 16. Oktober 2015
- [TR021021] BSI Technische Richtlinie BSI TR-02102-1: „Kryptografische Verfahren: Empfehlungen und Schlüssellängen“, Version 2014-01 vom 10.2.2014
- [TR021022] BSI Technische Richtlinie BSI TR-02102-2: „Verwendung von Transport Layer Security (TLS)“, Version 2014-01
- [TRUST] https://www.openssl.org/docs/apps/x509.html#TRUST_SETTINGS
- [VFY] https://www.openssl.org/docs/ssl/SSL_CTX_set_verify.html
- [VFY-CERT] https://www.openssl.org/docs/crypto/X509_verify_cert.html

- [x509test] <https://github.com/yymax/x509test>
- [ZDNET] <http://www.zdnet.com/68-percent-of-top-free-android-apps-vulnerable-to-cyberattack-researchers-claim-7000032875/>
- [ZJRR] Y. Zhang, A. Juels, M.K. Reiter, T. Ristenpart: „Cross-Tenant Side-Channel Attacks in PaaS Clouds“, Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security
- [ZJRR12] Yinqian Zhang, Ari Juels, Mike Reiter, and Thomas Ristenpart: Cross-VM Side Channels and Their Use to Extract Private Keys, CCS '12