Ethical Hacking

Die Netz Security

Crypto Cheat Sheet

Steckbrief: Kryptographie

Manu Carus

http://www.ethical-hacking.de/
mailto:manu.carus@ethical-hacking.de

Ethical Hacking

Die Netz Security

INHALTSVERZEICHNIS

1 Kryptographie

Definitionen

Die **Kryptographie** ist die Wissenschaft der Verschlüsselung von Information.
Dem steht die **Kryptoanalyse** entgegen, mit dem Ziel, bisher als sicher eingeschätzte Verfahren zu analysieren und zu brechen.
Beide Wissenschaften sind Teilgebiete der **Kryptologie**.

Kryptographische Ziele

Vertraulichkeit (Secrecy)
"keeping information secret"
Zu schützende Information muss geheim bleiben. Nur ausdrücklich berechtigte Personen dürfen in der Lage sein, vertrauliche Nachrichten zu lesen oder Informationen über deren Inhalt zu erlangen.

Integrität (Integrity)
„knowing that information hasn't been tampered with"
Der Empfänger einer Nachricht muss in der Lage sein festzustellen, ob die Nachricht nach ihrer Erzeugung und vor ihrem Empfang modifiziert wurde.

Authentizität (Authentication)
„knowing the origin and destination of information"
Urheber und Empfänger einer Nachricht müssen beide eindeutig identifizierbar sein.

Nichtabstreitbarkeit (Non-Repudiation):
„knowing that information, once sent, cannot be retracted or denied".
Der Urheber einer Nachricht darf nicht in der Lage sein, seine Urheberschaft zu bestreiten. Die Urheberschaft muss sich gegenüber Dritten nachweisen lassen.

Anforderung

Eine „sichere" Applikation muss diese vier kryptographischen Ziele erreichen!

2 Sichere Algorithmen

Algorithmus		Länge[1]	
RSA-2048	Asymmetrische Verschlüsselung	2048 bit	256 Bytes
3DES	Cipher	112 bit[2]	14 Bytes
AES-128	Cipher	128 bit	16 Bytes
AES-192	Cipher	192 bit	24 Bytes
AES-256	Cipher	256 bit	32 Bytes
Blowfish	Cipher	128 bit	16 Bytes
Twofish	Cipher	128, 192, 256 bit	16, 24, 32 Bytes
DSA	Digitale Signatur	1024 bit	128 Bytes
ECDSA	Digitale Signatur	256, 384, 512 bit	32, 48, 64 Bytes
RSA-2048	Digitale Signatur	2048 bit	256 Bytes
RIPEMD-160	Hashalgorithmus	160 bit	20 Bytes
SHA-256	Hashalgorithmus	256 bit	32 Bytes
SHA-384	Hashalgorithmus	384 bit	48 Bytes
SHA-512	Hashalgorithmus	512 bit	64 Bytes
Diffie-Hellman	Key Agreement	≥ 1024 bit	≥ 128 Bytes
AES-CMAC	Message Authentication Code	128 bit 192 bit 256 bit	16 Bytes 24 Bytes 32 Bytes
HMACSHA256	Message Authentication Code	256 bit	64 Bytes
MAC-3DES-CBC	Message Authentication Code	112 bit	14 Bytes
PBKDF2	Password-based Key Derivation	beliebig	beliebig

Stand: 12/2012

(fett markiert ≈ **Empfehlung**)

[1] Länge des Schlüssels bzw. des Hashwertes
[2] effektive Schlüssellänge

3 Libraries und Frameworks

3.1 OpenSSL

Anwendung	Kommando	Parameter
Zufallszahlen	`openssl rand`	
Hashing	`openssl dgst -alg`	`alg ∈ {sha256, sha384, sha512, ripemd160}`
Message Authentication Codes	`openssl dgst alg -hmac`	`alg ∈ {sha256, sha384, sha512, ripemd160}`
Symmetrische Verschlüsselung	`openssl enc -cipher`	`cipher ∈ {aes128, aes192, aes256, aes-128-cbc, aes-192-cbc, aes-256-cbc, bf, bf-cbc, blowfish, des3}`
Asymmetrische Verschlüsselung	`openssl rsautl`	
Hybride Verschlüsselung	`openssl smime -cipher`	`cipher ∈ {aes128, aes192, aes256, des3}`

Digitale Signatur	`openssl dsaparam`
	`openssl gendsa -cipher`
	`openssl genrsa -cipher`
	`openssl dsa`
	`openssl rsa`
	`openssl dgst -alg`
		`cipher ∈ {aes128, aes192, aes256, des3}`
		für DSA:
		`alg ∈ {dss1}`
		für RSA:
		`alg ∈ {sha256, sha384, sha512, ripemd160}`

Schlüsselaustausch	`openssl dhparam`	

3.2 Java Cryptography Architecture

Klasse	algorithm ∈ { … }
javax.crypto.Cipher	AES AESWrap Blowfish DESede DESedeWrap DSA ECDSA HmacSHA256 HmacSHA384 HmacSHA512 RSA SHA1PRNG SHA-256 SHA256withECDSA SHA256withRSA SHA-384 SHA384withECDSA SHA384withRSA SHA-512 SHA512withECDSA SHA512withRSA
javax.crypto.KeyGenerator	
javax.crypto.Mac	
javax.crypto.SecretKeyFactory	
java.security.AlgorithmParameters	
java.security.AlgorithmParameterGenerator	
java.security.KeyFactory	
java.security.KeyPairGenerator	
java.security.MessageDigest	
java.security.SecureRandom	
java.security.Signature	

3.3 .NET

Klasse	Version
System.Security.Cryptography.AesManaged	.NET 3.0
System.Security.Cryptography.AesCryptoServiceProvider	.NET 3.0
System.Security.Cryptography.DSACryptoServiceProvider	.NET 2.0
System.Security.Cryptography.ECDiffieHellmanCng	.NET 3.0
System.Security.Cryptography.ECDsaCng	.NET 3.0
System.Security.Cryptography.KeyedHashAlgorithm.HMACRIPEMD160	.NET 2.0
System.Security.Cryptography.KeyedHashAlgorithm.HMACSHA256	.NET 2.0
System.Security.Cryptography.KeyedHashAlgorithm.HMACSHA512	.NET 2.0
System.Security.Cryptography.KeyedHashAlgorithm.MACTripleDES	.NET 2.0
System.Security.Cryptography.ProtectedData	.NET 2.0
System.Security.Cryptography.ProtectedMemory	.NET 2.0
System.Security.Cryptography.RijndaelManaged	.NET 2.0
System.Security.Cryptography.RIPEMD160Managed	.NET 2.0
System.Security.Cryptography.RNGCryptoServiceProvider	.NET 2.0
System.Security.Cryptography.RSACryptoServiceProvider	.NET 2.0
System.Security.Cryptography.SHA256Managed	.NET 2.0
System.Security.Cryptography.SHA384Managed	.NET 2.0
System.Security.Cryptography.SHA512Managed	.NET 2.0
System.Security.Cryptography.TripleDESCryptoServiceProvider	.NET 2.0
System.Security.SecureString	.NET 2.0

3.4 CryptoAPI und CAPICOM

Funktion	ALG_ID ∈ { … }
CryptCreateHash()	CALG_3DES CALG_3DES_112 CALG_AES CALG_AES_128 CALG_AES_192 CALG_AES_256 CALG_DSS_SIGN CALG_ECDSA CALG_HMAC CALG_MAC CALG_RSA_SIGN CALG_SHA_256 CALG_SHA_384 CALG_SHA-512 Provider for Diffie-Hellman: CAPICOM_PROV_MS_DEF_ DSS_DH_PROV
CryptDecrypt()	
CryptEncrypt()	
CryptDecryptAndVerifyMessageSignature()	
CryptDecryptMessage()	
CryptEncryptMessage()	
CryptGenKey()	
CryptGenRandom()	
CryptHashData()	
CryptHashMessage()	
CryptHashPublicKeyInfo()	
CryptHashSessionKey()	
CryptProtectData()	
CryptProtectMemory()	
CryptSignAndEncryptMessage()	
CryptSignHash()	
CryptSignMessage()	
CryptSignMessageWithKey()	
CryptUnprotectData()	
CryptUnprotectMemory()	
CryptVerifyDetachedMessageHash()	
CryptVerifyDetachedMessageSignature()	
CryptVerifyMessageHash()	
CryptVerifyMessageSignature()	
CryptVerifyMessageSignatureWithKey()	
CryptVerifySignature()	

4 Best Practices

4.1 Kryptographie

⇨ Kryptographische Agilität: Konfiguration und Absicherung der eingesetzten Algorithmen.

4.2 Zufallszahlen

⇨ GUIDs mit kryptographisch sicheren Zufallszahlen erzeugen.

4.3 Hashing

⇨ SHA-256.
⇨ Hashwerte der Länge 256 bit und größer (≥ 32 Bytes).
⇨ Hashing statt Verschlüsselung.
⇨ Salted Hashing statt Hashing bei kleinen Eingaben (z.B. bei Passwörtern).
⇨ Vergleich langer Eingaben via Vergleich ihrer Hashwerte.
⇨ Integritätsprüfung via Hashing.
⇨ Hashwerte verschlüsselt übertragen.
⇨ Digitale Signatur statt Hashing.

4.4 Symmetrische Verschlüsselung

⇨ AES-256.
⇨ Schlüssel der Länge 128 bit und größer (≥ 16 Bytes).
⇨ Schlüssel der Länge 256 bit empfohlen.
⇨ SSL statt symmetrischer Verschlüsselung.
⇨ Hybride Verschlüsselung statt symmetrischer Verschlüsselung.
⇨ Schlüssel aus kryptographisch sicheren Zufallszahlen generieren.
⇨ Initialisierungsvektor (IV) benutzen und aus kryptographisch sicheren Zufallszahlen generieren.
⇨ Initialisierungsvektor (IV) nicht mehrfach verwenden.
⇨ Cipher Block Chaining (CBC) verwenden.
⇨ PKCS #7 Padding verwenden.
⇨ Passwort nur in Verbindung mit PBKDF2 als Schlüssel verwenden.
⇨ Passwörter als Schlüssel: min. 12 Zeichen Länge, starke Passwörter verwenden (Passphrases).

4.5 Asymmetrische Verschlüsselung

⇨ RSA-2048 für Datenübertragung.
⇨ RSA mit Schlüssel > 2048 bit Länge für Langzeitarchivierung.
⇨ Hybride Verschlüsselung statt asymmetrischer Verschlüsselung.
⇨ Versand an n Empfänger: symmetrischen Schlüssel n-mal asymmetrisch verschlüsseln.

4.6 Digitale Signatur

- ⇨ DSA statt RSA.
- ⇨ Signaturverfahren mit Hashwerten der Länge 160 bit und größer (≥ 20 Bytes).
- ⇨ Protokoll über alle signierten Dokumente führen.

4.7 Message Authentication Codes

- ⇨ HMACSHA256 und AES-CMAC-256.
- ⇨ lange Schlüssel verwenden.
- ⇨ Schlüssel aus kryptographisch sicheren Zufallszahlen generieren.
- ⇨ Passwörter als Schlüssel: mind. 16 Zeichen lang, starke Passwörter verwenden (Passphrases).
- ⇨ Digitale Signatur statt Message Authentication Code.

4.8 Schlüsselaustauschverfahren

- ⇨ Diffie-Hellman nur in Verbindung mit gegenseitiger Authentisierung anwenden (MITM).

5 Worst Practices

5.1 Kryptographie

- ⇨ Security by Obscurity.
- ⇨ Ad-hoc-Algorithmen.

5.2 Hashing

- ⇨ MD5.
- ⇨ SHA-1.

5.3 Symmetrische Verschlüsselung

- ⇨ DES.
- ⇨ Electronic Code Book (ECB).
- ⇨ Daten auf einer durchgehend abgesicherten Leitung zweifach verschlüsseln.
- ⇨ Verschlüsselte Daten komprimieren.

5.4 Asymmetrische Verschlüsselung

- ⇨ Inhalte asymmetrisch verschlüsseln.
- ⇨ RSA-1024.

6 Rules of Thumb

6.1 Zufallszahlen

⇨ Die Berechnung kryptographisch sicherer Zufallszahlen ist ca. 10 Mal aufwändiger als die Berechnung einfacher Zufallszahlen.

6.2 Hashing

⇨ Hashwerte der Länge 256 bit können ca. 10^{76} verschiedene Dokumente kollisionsfrei darstellen.

6.3 Symmetrische Verschlüsselung

⇨ AES verschlüsselt schneller als Triple DES.

⇨ Twofish statt Blowfish.

⇨ Triple DES Schlüsselstärke beträgt effektiv nur 112 bit.

⇨ Encoding zwischen Absender und Empfänger synchronisieren.

6.4 Asymmetrische Verschlüsselung

⇨ Asymmetrische Verschlüsselung ist ca. 1.000 mal langsamer als symmetrische Verschlüsselung.

6.5 Schlüsselaustauschverfahren

⇨ Die Berechnung geeigneter Diffie-Hellman-Parameter ist sehr aufwändig und muss durchgeführt werden, lange bevor die beiden Kommunikationspartner eine Verbindung aufbauen.

7 Salvatorische Klauseln

⇨ Unterstützt die Umgebung keinen kryptographisch sicheren Algorithmus, und ist unter den gegebenen Umständen keine andere Lösung möglich, so sollte ein kryptographisch schwacher Algorithmus eingesetzt werden anstatt gänzlich auf die Absicherung zu verzichten. Dieser Umstand ist nachvollziehbar zu dokumentieren und als solches vom Auftragnehmer abzunehmen.

⇨ Sollten die in diesem Dokument als sicher eingestuften und empfohlenen Algorithmen zu einem späteren Zeitpunkt gebrochen werden, so gelten bis zu der nächsten Aktualisierung dieses Dokuments die Empfehlungen des *Bundesamts für Sicherheit in der Informatik* (BSI) sowie das *Gebot der Vorsicht*.

8 Abkürzungsverzeichnis

3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
BF	Blowfish
CAPICOM	Cryptographic Application Programming Interface through Component Object Model
CBC	Cipher Block Chaining
CFB	Cipher Feedback Mode
CMAC	Cipher-based Message Authentication Code
CNG	Cryptography Next Generation
DESede	Synonym für 3DES / Triple DES
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
ECB	Electronic Code Book
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
GUID	Globally Unique Identifier
HMAC	key-Hashed Message Authentication Code
IV	Initialization Vector
JCA	Java Cryptography Architecture
MAC	Message Authentication Code
MD	Message Digest
MITM	Man-in-the-Middle
OFB	Output Feedback Mode
PBKDF	Password-Based Key Derivation Function
PGP	Pretty Good Privacy
PKCS	Public Key Cryptography Standards
PKI	Public Key Infrastructure
PRNG	Pseudorandom Number Generator
RC	Rivest Cipher oder Ron's Code
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
RSA	Rivest Shamir Adleman
SHA	Secure Hash Algorithm
S/MIME	Secure Multipurpose Internet Mail Extension
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TRNG	True Random Number Generator