Themen zur Computersicherheit

Schlüssel

PD Dr. Reinhard Bündgen
buendgen@de.ibm.com

Themen zu Schlüsseln

- Wie erzeugt man Schlüssel?
- Gute Schlüssel / schlechte Schlüssel
- Was sind die richtigen Schlüssellängen
- Darstellung von Schlüsseln

Erzeugen von symmetrischen Schlüsseln

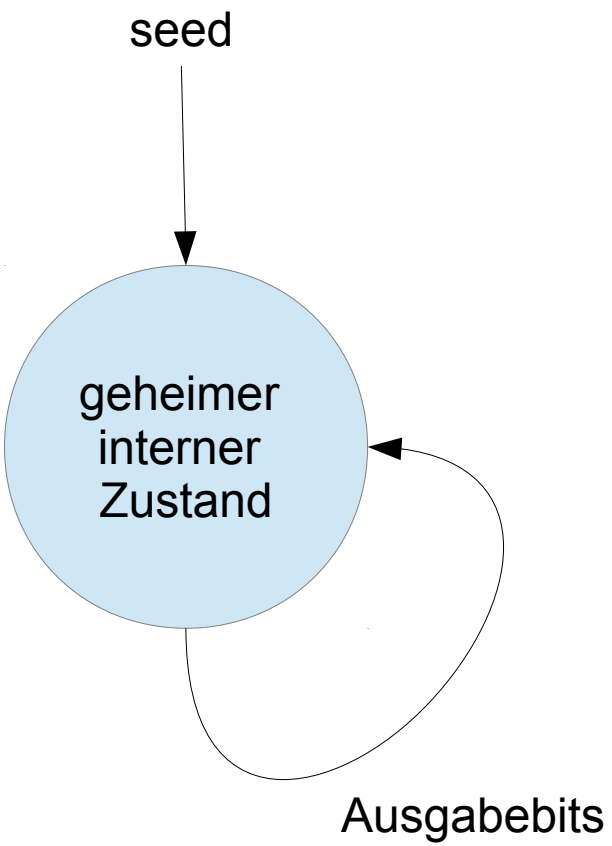
- Zufallsworte (byte strings) der Schlüssellänge
  - i.A. pseudozufällige Daten
- Test auf Schlüsselgüte
  - falls Schlüssel schwach, neuer Versuch

- schwache DES Schlüssel
  - 4 schwache Schlüssel: involutorisch
  - 6 halbschwache Schlüsselpaare

Pseudozufallsgeneratoren

kryptografisch sichere Pseudozufallszahl Generatoren

- generieren, abhängig von einem initialen Wert (seed), einen deterministischen Bitstrom
- Bitverteilung statistisch zufällig
- ohne Wissen des internen Zustands (einschl. seeds) kann aus bisher erzeugtem Bitstrom künftiger Bitstrom nicht vorhergesagt werden
- Basieren oft auf Hashfunktionen oder Verschlüsselungsverfahren.



Ein paar Größenordnungen

- Wie wahrscheinlich ist es einen Schlüssel zu raten?
- EXTREM gering!

Lotto (6 aus 49 mit Superzahl)	1 / 139.838.160 um ca €5,000,000 zu gewinnen
DES: 2^{56} Schlüssel	72.057.594.037.927.936
2^{64} ns (Zyklen auf 1 Ghz Rechner)	ca 2^{34}s, ca 500 Jahre
Alter des Universums	$5*10^{28}$ ns
AES128: 2^{128} Schlüssel	38 Ziffern
Zahl der Atome der Erde	ca $1.33*10^{50}$
Zahl der Atome im Sonnensystem	ca $1.2*10^{57}$
AES-192: 2^{192} Schlüssel	58 Ziffern
AES-256/SHA-256: 2^{256} Schlüssel/Hashes	77 Ziffern
Zahl der Atome im Universum	ca 10^{82} (10^{78} – 10^{85})
SHA-512: 2^{512} Hashes	154 Ziffern

- Schlüssellängen für symmetrische Schlüssel sind nicht vergleichbar mit Schlüssellängen für asymmetrische Schlüssel

Erzeugung von RSA Schlüsseln

- erzeuge p und q mit $\lceil \mathrm{ld}\ p \rceil = \lceil \mathrm{ld}\ q \rceil = \lceil (\mathrm{ld}\ n) / 2 \rceil$
  - wiederhole erzeuge Zufallszahl x der Länge $\lceil (\mathrm{ld}\ n) / 2 \rceil$ bis x prim
- wähle kurzen (aber nicht zu kurzen) öffentlichen Schlüssel e
  - einige Standards verlangen $e \geq 17$
  - beliebtes $e = 2^{16}+1$
- invertiere e modulo $(p-1)\bullet(q-1)$ mit erweitertem Euklidischen Algorithmus
- Eventuell berechne die CRT Parameter zu privatem Schlüssel d

Erzeugung von DH Parametern und Schlüsseln

- Erzeuge p und q
  - für p = N • q + 1 und pl= ⌈ld p⌉ , ql = ⌈ld q⌉
  - erzeuge zufällige Primzahl q der gewünschten Länge ql
  - wiederhole: erzeuge zufällige gerade Zahl N der Länge pl - ql bis N • q + 1 ist prim

- Erzeuge g
  - wiederhole: erzeuge 2 < a < p-2 bis $a^N \neq 1 \bmod p \wedge a^{N \cdot q} = 1 \bmod p$
  - setze g = a^N

- Erzeugung der Schlüssel von Alice und Bob:
  - jeweils: erzeuge Zufallszahl kleiner g

Sicherheitsmaße verschiedener Verfahren

nach NIST SP 800-57, Teil 1, Rev.3, Juli 2012

Sicherheit in Bits	Symmetrische Verfahren	DSA, DH (Diskrete Log)	RSA (Faktorisierung)	ECC
80	2DES	n=1024, q=126	n=1024	n=160 – 223
112	3DES	n=2048, q=224	n=2048	n=224 – 255
128	AES-128	n=3072, q=256	n=3072	n=256 – 383
192	AES-192	n=7680, q=384	n=7680	n=384 – 511
256	AES-256	n=15360, q=512	n=15360	n=512+

Empfohlene Schlüssel- und Hashlängen

Organisation	Kriterien	Hash	Symmetric	RSA	DH key	DH group	ECC
ECRYPT II	Level 7: -2030	224	112	2432	224	2432	224
(2012)	Level 8: -2040	256	128	3248	256	3248	256
	Level 9: “forever”	512	256	15424	512	15424	512
BSI	2013 - 2015	224		1976	224	2048	224
(2014)	2016-2020	256		1976	256	2048	250
	> 2020	256		1976	256	2048	250
NSA Suite B	secret	256	128				256
(2013)	top secret	384	256				384
ANSSI	2010-2020	200	100	2048	200	2048	200
(2010)	>2020	256	128	4096	200	2048	256
NIST	2011-2030	224	112	2048	224	2048	224
(2012)	>2030	256	128	3072	256	3072	256
	>>2030	384	192	7680	385	7680	384
	>>>2030	512	256	15360	512	15360	512

Quelle: www.keylength.com (06/2014)

Darstellung von Schlüsseln

- einfache Symmetrische Schlüssel
  - Bytearray & Länge
- asymmetrische Schlüssel und Parameter: Strukturen aus mehreren Komponenten
  - Strukturbeschreibungsspache zur Darstellung Schlüssel
    - für Übertragung im Netz
    - für Softwarebibliotheken

Abstract Syntax Notation One (ASN.1)

- reine Struktur Beschreibungssprache
  - BNF ähnlich
  - elementare ASN.1 Daten Typen (z.B.)
    - `BIT STRING`
    - `BOOLEAN`
    - `INTEGER`
  - komplexe ASN.1 Daten Typen (z.B.)
    - `CHOICE`
    - `SEQUENCE (OF)`
    - `SET (OF)`
  - `OBJECT IDENTIFIER`
  - freies Buch: (*)
- Kodierungen von ASN.1
  - Basic Encoding Rules (BER)
  - Canonical Encoding Rules (CER)
  - Distinguished Encoding Rules (DER)
  - Packed Encoding Rules (PER)

(*) http://www.oss.com/asn1/resources/books-whitepapers-pubs/asn1-books.html#dubuisson

Beispiele von ASN.1 Beschreibungen aus PKCS #1

```
pkcs-1    OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1
}

RSAPrivateKey ::= SEQUENCE {
    version           Version,
    modulus           INTEGER,  -- n
    publicExponent    INTEGER,  -- e
    privateExponent   INTEGER,  -- d
    prime1            INTEGER,  -- p
    prime2            INTEGER,  -- q
    exponent1         INTEGER,  -- d mod (p-1)
    exponent2         INTEGER,  -- d mod (q-1)
    coefficient       INTEGER,  -- (inverse of q) mod p
    otherPrimeInfos   OtherPrimeInfos OPTIONAL
 }

 OAEP-PSSDigestAlgorithms    ALGORITHM-IDENTIFIER ::= {
     { OID id-sha1       PARAMETERS NULL }|
     { OID id-sha224     PARAMETERS NULL }|
     { OID id-sha256     PARAMETERS NULL }|
     { OID id-sha384     PARAMETERS NULL }|
     ...
 }
 sha1    HashAlgorithm ::= {
     algorithm   id-sha1,
     parameters  SHA1Parameters : NULL
 }

 HashAlgorithm ::= AlgorithmIdentifier { {OAEP-PSSDigestAlgorithms} }

 RSAES-OAEP-params ::= SEQUENCE {
     hashAlgorithm      [0] HashAlgorithm     DEFAULT sha1,
     maskGenAlgorithm   [1] MaskGenAlgorithm  DEFAULT mgf1SHA1,
     pSourceAlgorithm   [2] PSourceAlgorithm  DEFAULT pSpecifiedEmpty
}
```