Empfehlungen zur Kryptographie

Geeignete Algorithmen für qualifizierte elektronische Signaturen
Kryptographische Verfahren: Empfehlungen und Schlüssellängen

Julian Schiele
JulianSchiele@mytum.de

Sommerakademie der Studienstiftung des Deutschen Volkes
Arbeitsgruppe Applied Cryptography and Software Engineering

„ It is impossible to foresee the consequences of being clever.“

- CHRISTOPHER STRACHEY

Übersicht

1| Einleitung

Technische Richtlinie des BSI:
Kryptographische Verfahren: Empfehlungen und Schlüssellängen (20/06/08)

Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahn: (17/11/08)
Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen)

- Bewertung + langfristige Orientierung des Sicherheitsniveaus
- Keine mathematische & patentrechtliche Betrachtung
  Genaue Spezifikationen: ISO/IEC, IEEE, NIST
- Beste Algorithmen + Leistungsfähigkeit von Rechnern
  → Prognose für kommende 7 Jahre (bis 2015)
  bei Sicherheitsniveau von 100 Bit

Beispiel

symmetrisch		**asymmetrisch**		
Verschlüsselung	MAC	RSA	DSA	ECDSA
100	100	2048	224/2048	200

Schlüssellängen für ein Sicherheitsniveau von mind. 100 Bit

Anforderungen

Inhalt muss vertraulich bleiben → **Verschlüsselung**

Eindeutig zuzuordnender Sender
Keine Manipulation auf dem Weg → **Authentisierung**

Vorarbeit

Hashing reduziert auf feste Bitlänge

Schlüsseleinigungsverfahren, Secret Sharing, Zufallszahlengenerator

Weitere Sicherheitsfaktoren

Stärke der Algorithmen

Konkrete Implementierung in Hard- und Software

Zuverlässige Hintergrundsysteme (z.B. Public Key Infrastruktur)

2| Hashfunktionen

Bitstring auf feste Länge reduzieren: $m \in \{0,1\}^* \rightarrow h \in \{0,1\}^n$

Bedingungen

	Es ist praktisch unmöglich ...
Einweg-Eigenschaft	… für gegebenes h ein m mit H(m)=h zu finden
2nd-Preimage-Eigenschaft	… für gegebenes m≠k ein k mit H(m)=H(k) zu finden $k \in \{0,1\}^*$
Kollisionsresistenz	… m und n zu finden, so dass m≠k und H(m)=H(k)

Bitlänge n sollte mindestens 200 betragen. (Geburtstagsparadoxon)

Beispiel

geeignet bis Ende:	2009	2010	2015
	SHA-1*	SHA-1* + Seriennummer mit mind. 20 Bit Entropie	SHA-224 SHA-256 SHA-384 SHA-512
		RIPEMD-160	SHA-1, RIPEMD-160 ausschließlich zur Prüfung qualifizierter Zertifikate

Empfohlene Hashfunktionen

* zur Erzeugung qualifizierter Zertifikate, nicht jedoch anderer signierter Daten

3| Verschlüsselungsverfahren

→ Vertraulichkeit gewährleisten

Symmetrisches Verfahren

Verschlüsselungsschlüssel
= Entschlüsselungsschlüssel

Asymmetrisches Verfahren

Öffentlicher Schlüssel
≠ geheimer Schlüssel

3.1| Symmetrische Verschlüsselungsverfahren

a) Blockchiffren

Klartext fester Bitlänge
(= *Blockgröße*)

Schlüssel

Chiffretext gleicher Länge

→ Blockgröße sollte mindestens 128 Bit betragen

Beispiel

AES-128, AES-192, AES-256
SERPENT-128, SERPENT-192, SERPENT-256
Twofish-128, Twofish-192, Twofish-256

Empfohlene Blockchiffren

Bei längeren Klartexten → *Betriebsarten*

Partitionierung: Text in Blöcke aufteilen

Padding: ggf. letzten Block auffüllen

Beispiel

ISO-Padding
ESP-Padding

Empfohlene Paddingverfahren

ECB (Elektronic Code Book)
selber Schlüssel für jeden Block

→ Informationen über die Struktur

Von weiterem Wert abhängig:
(z.B. vorheriger Chiffreblock, Counter etc)

Bedingungen an Initialvektoren...

Beispiel

CBC (Cipher-Block Chaining)	nur unvorhersagbare Initialvektoren zu verwenden
GCM (Galois-Counter-Mode)	Zählerstände dürfen sich bei geheimem Schlüssel nicht wiederholen
CTR (Counter Mode)	Zählerstände dürfen sich bei gleichem Schlüssel nicht wiederholen

Empfohlene Betriebsarten für Blockchiffren

b) Stromchiffren

Schlüssel + Initialisierungsvektor

Schlüsselstrom

→ diesen auf Klartext Modulo 2 addieren

→ keine Stromchiffren empfehlenswert

3.2| Asymmetrische Verschlüsselungsverfahren

→ in der Praxis: zur Übertragung des symmetrischen Schlüssels

Anforderungen

Klartext darf nicht aus Chiffretext rekonstruierbar sein

Geheimer darf nicht aus öffentlichem Schlüssel konstruierbar sein

Zur Zuordnung: Public Key Infrastruktur

Algorithmen

Generierung von Schlüsselpaaren

Verschlüsselung & Entschlüsselung

Beispiel

RSA
DLIES
ECIES

Empfohlene asymmetrische Verschlüsselungsverfahren

a) RSA

→ 1977 von Ronald **R**ivest, Adi **S**hamir und Leonard **A**dleman entwickelt

Grundlage

Faktorisierungsproblem ganzer Zahlen

Schlüsselgenerierung

1. Primzahlen p und q wählen mit: $0{,}1 < |ld\ p - ld\ q| < 30$
2. Öffentlichen Exponenten e wählen mit: $ggT(e, (p-1)\cdot(q-1)) = 1$
   und: $2^{16} + 1 \leq e \leq 2^{1824} - 1$
3. Geheimen Exponenten d berechnen: $e \cdot d = 1 \bmod kgV(p-1, q-1)$

Modulus n = p ·q

→ **öffentlicher Schlüssel** (n, e)

geheimer Schlüssel d

Schlüssellänge

Länge des Modulus n: mind. 2048 Bit

→ zuerst e wählen, dann d → keine kleinen geheimen Exponenten

→ geeignete Primzahltests verwenden

→ p und q geheim halten

→ vor Verschlüsselung: Klartext auf Bitlänge des Modulus formatieren

-> RSA-SigAlgo

b) DLIES

→ Discrete Logarithm Integrated Encryption Scheme

Grundlage

Diskretes Logarithmenproblem in Körpern F_p mit Primordnung p

Schlüsselgenerierung

1. Primzahlen p wählen
2. Erzeuger g aus F_p^* wählen
3. $A := g^a$ bestimmen mit zufälliger Zahl a ϵ {1, .. , p-2)

→ **öffentlicher Schlüssel** (p, g, A)

geheimer Schlüssel a

Schlüssellänge

Länge der Primzahl p: mind. 2048 Bit

c) ECIES

→ Elliptic Curve Integrated Encryption Scheme

Grundlage

Diskretes Logarithmenproblem auf elliptischen Kurven

Schlüsselgenerierung

1. EC-Systemparameter (p, a, b, P, q, i) erzeugen
2. d zufällig und gleichverteilt in {1, .. , q-1} wählen
3. G := d · *P*

öffentlicher Schlüssel (p, a, b, P, q, i) mit G

geheimer Schlüssel d

Schlüssellänge

Ordnung q des Basispunktes P: $q \geq 2^{224}$

4| Datenauthentisierung

→ Garantie, dass Daten nicht verändert wurden

Funktionsweise

Beweisender berechnet mit Schlüssel die Prüfsumme der Daten

Prüfer vergleicht empfangene mit zu erwartender Prüfsumme

Varianten

Symmetrisches Verfahren:
Selber Schlüssel

↓

MAC (Message Authentication Codes)

⟷

Asymmetrisches Verfahren:
Beweisender: geheimer Schlüssel
Prüfer: öffentlicher Schlüssel

↓

Signaturverfahren

↓

4.1| Signaturverfahren

Vorgehen

→ **Hashing** der Daten

→ Mit geheimem Schlüssel **Signatur** des Hashwertes berechnen

→ Prüfer verifiziert die Prüfsumme mit öffentlichem Schlüssel

Algorithmen

Generierung von Schlüsselpaaren

Hashfunktion

Signier- & Verifizieralgorithmus

Beispiel

RSA
DSA
DSA-Varianten
Merkle-Signaturen

Empfohlene Signaturverfahren

a) RSA-Signieralgorithmus

→ Grundlage und Vorgehensweise wie bei RSA-Verschlüsselungsalgorithmus

Allgemeiner Hinweis

Ein kryptographisches Verfahren kann häufig für verschiedene Anwendungen eingesetzt werden:
→ ABER dann müssen für unterschiedliche Anwendungen jeweils verschiedene Schlüssel benutzt werden !

Beispiel

Bis Ende	2008	2009	2010	2015
mindestens	1280	1536	1728	1976
empfohlen	2048	2048	2048	2048

Empfohlene Schlüssellängen für Modul n

b) DSA

→ Digital Signature Algorithm

Grundlage

Diskretes Logarithmenproblem in Körpern F_p mit Primordnung p

Schlüsselgenerierung

1. Primzahlen p und q wählen, so dass: *q teilt (p-1)*
2. Berechne g mit beliebigem x ϵ F_p: $g := x^{(p-1):q} \bmod p$
3. Wenn g=1 → wiederhole 2.
4. Berechne A mit a ϵ {1, .. , q-1}: $A := g^a$

→ **Öffentlicher Schlüssel** (p, q, g, A)

Geheimer Schlüssel a

Beispiel

Bis Ende	**2008**	**2009**	**2015**
p mindestens	1280	1536	
empfohlen	2048	2048	2048
q	160	160	224

Empfohlene Schlüssellängen für p und q

c) DSA-Varianten

Grundlage

Diskretes Logarithmusproblem in elliptischen Kurven

Gruppe $E(F_p)$

Elliptische Kurve E und Punkt P auf $E(F_p)$ erzeugen, so dass:

$ord(P) = q$ mit Primzahl $q \neq p$

$r_0 := min(\ r:\ q$ teilt $p^r-1)$ mit $r_0 > 10^4$

Gruppe $E(F_{2^n})$

Elliptische Kurve E und Punkt P auf $E(F_{2^n})$ erzeugen, so dass:

Primzahl m

$E(F_{2^n})$ ist nicht über F_2 definierbar

$ord(P) = q$ mit Primzahl q

$r_0 := min(\ r:\ q$ teilt $2^{nr}-1)$ mit $r_0 > 10^4$

d) Merklesignaturen

→ beruht nicht auf der Schwierigkeit eines mathematischen Problems, sondern auf der Stärke einer Hashfunktion

5| Instanzauthentisierung

→ Beweisender weist den **Besitz eines Geheimnisses** nach

Symmetrisches Verfahren:
Geheimnis = Schlüssel

→ vorheriger Austausch

Asymmetrisches Verfahren:
Geheimnis = geheimer Schlüssel

→ PKI nötig

Passwortbasiertes Verfahren:
Geheimnis = PIN

→ z.B. für Freischaltung von Chipkarten

5.1| Symmetrische Instanzauthentisierung

Vorgehen

Challenge-Response-Verfahren

Beweisender		Prüfer
		Wähle Zufallswert r
	← (Challenge)	
Berechne Prüfsumme von r		
	→ (Response)	
		Verifiziere Prüfsumme

→ Verschlüsselungsverfahren + MAC-Verfahren

5.2| Asymmetrische Instanzauthentisierung

→ Challenge-Response-Verfahren

Beweisender		Prüfer
		Wähle Zufallswert r
	← (Challenge)	
Berechne Prüfsumme von r mit geheimem Schlüssel		
	→ (Response)	
		Verifiziere Prüfsumme mit öffentlichem Schlüssel

5.3| Passwortbasierte Instanzauthentisierung

Passwörter sind meist kurz
→ Anzahl der Zugriffsversuche begrenzen

Nebenbedingungen

Passwort mit mindestens 6 Ziffern *[Entropie hat ld(10^6) Bit]*

Maximal 3 Zugriffsversuche

Kontaktbehaftete Chipkarten

Ungesicherter Übertrag der PIN von Kartenleser zur Chipkarte
→ zertifizierten Kartenleser benutzen

Kontaktlose Chipkarten

Kein ungesicherter Übertrag möglich
→ zusätzlicher Schutz nötig

z.B. PACE (Password Authenticated Connection Establishment):
Zusätzliches Schlüsseleinigungsverfahren

6| Schlüsseleinigungsverfahren

→ Austausch eines Schlüssels über unsicheren Kanal
→ mit Instanzauthentisierung kombinieren

Alle empfohlenen symmetrischen Verschlüsselungsverfahren

Alle empfohlenen asymmetrischen Verschlüsselungsverfahren sowie:

	Diffie-Hellman	**EC Diffie-Hellman**
Grundlage	Diskrete Logarithmenproblem in F_p	Diskrete Logarithmenproblem in elliptischen Kurven
Systemparameter	Primzahl p, Erzeuger g aus F_p^*	EC-Systemparameter (p,a,b,P,q,i)
Ablauf	A sendet g^x an B mit $x \in \{1, .., p-1)$ B sendet g^y an A mit $y \in \{1, .., p-1)$ A berechnet $(g^y)^x = g^{xy}$ B berechnet $(g^x)^y = g^{xy}$	A sendet xP an B mit $x \in \{1, .., p-1)$ B sendet yP an A mit $y \in \{1, .., p-1)$ A berechnet x(yP) = xyP B berechnet y(xP) = xyP
Schlüssellänge	p mind. 2048 Bit	q mind. 224 Bit

→ Die Systemparameter vorher **authentisch** austauschen

7| Secret Sharing

→ Shamir Secret-Sharing-Verfahren

Speichern von Schlüsseln → Kopien anlegen → gefährdet Geheimnis 

→ Schlüssel K in n Teilgeheimnisse aufteilen, so dass:

t ≤ n Teilgeheimnisse: Rekonstruktion von K

t - 1 Teilgeheimnisse: keine Information über K

Weitere Anwendung: t-aus-n-Augenprinzip

8| Zufallszahlengenerator

→ Entropie (= Unvorhersagbarkeit) in Bit

Physikalische Zufallszahlengeneratoren

Physikalische Rauschquellen:
elektromagnetische, elektromechanische, quantenmechanische Effekte

Deterministische Nachbearbeitung

→ P2-Generator mit Stärke der Mechanismen HOCH

Bedingungen

- Stochastisches Modell
- Entropiezuwachs oberhalb Schranke
- Regelmäßige statistische Tests
- Fehler → Stilllegen der Rauschquelle

8| Zufallszahlengenerator

Deterministische Zufallszahlengeneratoren

→ Pseudozufallszahlengeneratoren

Zufallswert (Seed) → pseudozufällige Bitfolge

Innerer Zustand: mit Seed initialisiert, erneuert → Zufallszahl

→ K4-Generator mit Stärke der Mechanismen HOCH

Bedingungen

Es ist praktisch unmöglich ...

Zufallszahlenfolge → Vorgänger, Nachfolger der Teilfolge & innere Zustand

Innerer Zustand → Vorgänger der Teilfolge, Vorgängerzustand

Seedgenerierung

Physikalischer Zufallszahlengenerator

GNU / Linux: /dev/random (in Kernelversion 2.6.21.5)

Windows: ReadTimeStampCounter() & KeQuerySystemTime()

Fragen ?

Falls nun nach der Sommerakademie
noch Fragen zu diesem Thema auftauchen sollten,
könnt Ihr euch gerne an mich wenden:

JulianSchiele@mytum.de

Oder eben selbst in den entsprechenden papers nachlesen:

Technische Richtlinie des BSI:
Kryptographische Verfahren: Empfehlungen und Schlüssellängen
(20/06/08)

Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahn: (17/11/08)
Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen)