Parameterwahl für sichere zeitgemäße Verschlüsselung

Prof. Dr. Mark Manulis
Kryptographische Protokolle
Fachbereich Informatik
TU Darmstadt / CASED
Mornewegstrasse 30
64293 Darmstadt

Room 4.1.15 (4th floor)

manulis (-at) informatik.tu-darmstadt.de
tel +49 (0)6151 16 50761
fax +49 (0)6151 16 72051

Datenschutz und Kryptographie

Datenschutz „bezeichnet den Schutz personenbezogener Daten vor Missbrauch … dass jeder Mensch grundsätzlich selbst entscheiden kann, wem wann welche seiner persönlichen Daten zugänglich sein sollen“.  (Quelle: Wikipedia)

Vorratsdatenspeicherung „bezeichnet Verpflichtung der Anbieter von Telekommunikations-diensten zur Registrierung von elektronischen Kommunikationsvorgängen ohne dass ein Anfangs-verdacht oder konkrete Hinweise auf Gefahren bestehen“.  (Quelle: Wikipedia)

Kryptographie ist die Wissenschaft der Verschlüsselung von Informationen

Klassische Ziele der Kryptographie

Vertraulichkeit	Daten vor unerlaubtem Zugriff schützen
Integrität	Daten vor unerlaubter Änderung schützen
Authentizität	Nachweis der Urheberschaft von Daten (eventuell mit Verbindlichkeit)

Kryptographie ist ein Werkzeug zum Datenschutz

Themen

Grundlagen und Taxonomie der Verschlüsselung

Perfekte Geheimhaltung vs. Praktische Sicherheit

Symmetrische Verschlüsselung (DES, 3-DES, AES)

Asymmetrische Verschlüsselung (RSA + Faktorisierung, ElGamal + DLog, ECC)

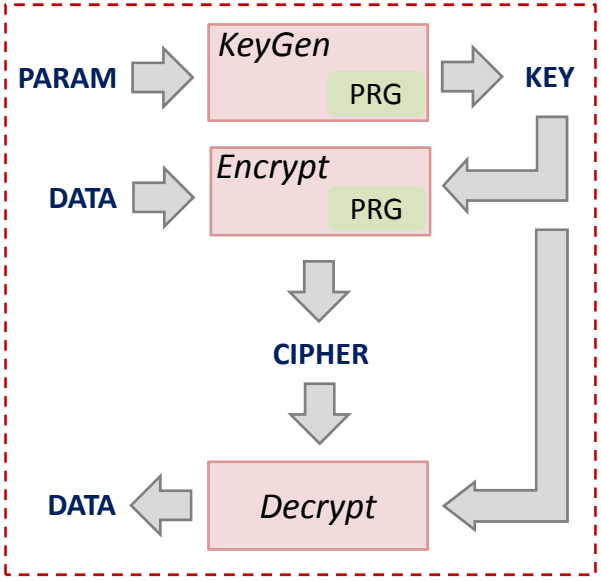
Zeitgemäße Verschlüsselung (Parameterwahl, praktisches How-To)

Ausblick und Trends

Grundlagen eines Verschlüsselungssystems

Algorithmen eines Verschlüsselungssystems

KeyGen	zur Schlüsselerzeugung (randomisiert)
Encrypt	zum Verschlüsseln (mglw. randomisiert)
Decrypt	zum Entschlüsseln



Prinzipien eines Verschlüsselungssystems

1. öffentliche Spezifikation von PARAM, *KeyGen, Encrypt, Decrypt*
2. nur KEY bleibt geheim
3. Beweis der Sicherheit des Systems (basierend auf annerkannten Annahmen; nicht immer möglich)

Sicherheit eines Verschlüsselungssystems

chosen-plaintext attacks (CPA)	Angreifer kriegt Zugang zu *Encrypt* (kann selbst verschlüsseln)
chosen-ciphertext attacks (CCA)	Angreifer kriegt Zugang zu *Decrypt* (kann selbst entschlüsseln)

Angreifer's Ziel **Inhaltsinformationen über DATA aus CIPHER zu ermitteln**

Taxonomie Moderner Verschlüsselung

Symmetrische (Private-Key) Verschlüsselung		Asymmetrische (Public-Key) Verschlüsselung
KEY = K *Encrypt*(K, DATA) = CIPHER *Decrypt*(K, CIPHER) = DATA	***Bedeutung***	KEY = (SK, PK) *Encrypt*(PK, DATA) = CIPHER *Decrypt*(SK, CIPHER) = DATA
K ist geheim		SK ist geheim, PK ist öffentlich
Feistel-Netzwerk Substitution-Permutation-Netzwerk	***Design***	Einweg-Funktionen/Permutationen zahlentheoretische Annahmen
Bruteforce lineare/differentielle Kryptoanalyse algebraische Angriffe Seitenkanalangriffe	***Angriffe***	Bruteforce effizientere Lösungsalgorithmen für zahlentheoretische Probleme Seitenkanalangriffe

Perfekte Geheimhaltung

Vorkommende Mengen in einem Verschlüsselungsverfahren

KEYS	Menge aller möglichen Schlüssel
DATA	Menge aller möglichen Daten
CIPHERS	Menge aller möglichen Chriffrate

Annahmen über Wahrscheinlichkeitsverteilungen

- Wahrscheinlichkeitsverteilungen über ***KEYS*** und ***DATA*** sind unabhängig
- Wahrscheinlichkeitsverteilung über ***DATA*** ist möglicherweise bekannt
  (einige Inhalte sind wahrscheinlicher als andere)

Definition der *perfekten Geheimhaltung*

Wahrscheinlichkeitsverteilungen über DATA und CIPHERS sind unabhängig

$\Leftrightarrow$

Chiffrate sollen *keine* Informationen über die verschlüsselten Daten liefern

$\Leftrightarrow$

die Verschlüsselung von DATA soll sich von der Verschlüsselung von DATA' nicht unterscheiden

One-Time Pad (Vernam, Mauborgne 1918)

Parameter

KEYS, DATA, CIPHERS sind $\{0,1\}^n$ Mengen (alle Bitstrings der Länge n)

Algorithmen

KeyGen	wähle K zufällig aus $\{0,1\}^n$	
Encrypt	C = K XOR DATA	(bitweise Addition)
Decrypt	DATA = C XOR K	

DATA = 0101	XOR
K = 1100	
C = 1001	XOR
K = 1100	
DATA = 0101	

Perfekte Geheimhaltung (Shannon 1949)

C verschlüsselt jeden Wert aus ***DATA*** mit gleicher Wahrscheinlichkeit:

alle Schlüssel sind gleichwahrscheinlich und

für jeden Wert DATA existiert der passende Schlüssel K = C XOR DATA

Praktische Nachteile

- die Länge von K muss *mindestens* die Länge von DATA haben
- jeder K darf nur *einmal* benutzt werden (sonst C XOR C' = DATA XOR DATA')

→ key management ist schwierig

Praktische Sicherheit

Ausgangslage

Schlüssellänge kleiner als Länge von DATA ⇒ keine perfekte Geheimhaltung

⇒ theoretisch unsicher

Praktische Sicherheit

(t, ε)-Sicherheit von Verschlüsselungsverfahren

- t maximal Zeit für den Angriff (gegeben durch moderne Rechenleistung, z.B. in CPU cycles)
- ε Wahrscheinlichkeit eines erfolgreichen Angriffs

Wahl der Sicherheitsparameter

t ist vorgegeben aber die Schlüssellänge n nicht

wähle n so dass ε möglichst klein ist

$\varepsilon < t/2^n$ ist eine gute Approximation

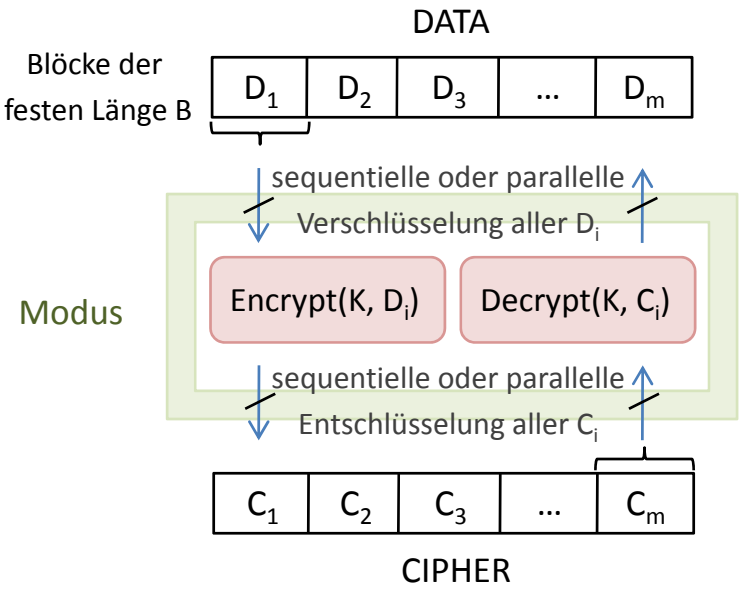
Beispiel

für moderne Rechner ist die Abschätzung $t < 2^{80}$ ausreichend

bei n = 128 Bits wäre $\varepsilon < 1/2^{48} \approx 3{,}55 \times 10^{-15}$ (die Zeit eines Bruteforce-Angriffs wäre 2^{128})

Symmetrische Verschlüsselung

Blockchiffren

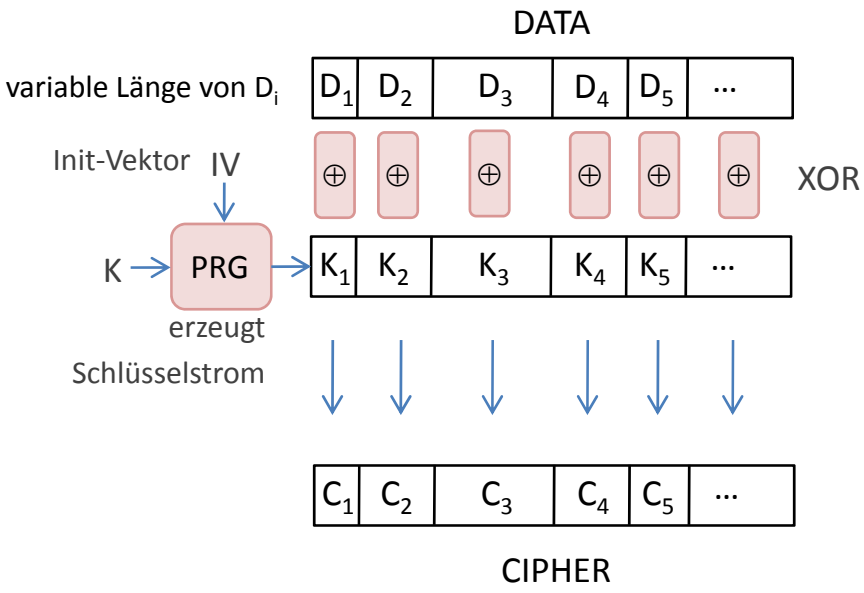


Blöcke werden je nach Modus vorbereitet

CBC, OFB, CTR sind CPA-sichere Modi

Beispiele: DES, 3-DES, AES, …

Stromchiffren



IV wird zufällig gewählt

synchronised mode	IV neu per DATA
unsynchronisiert mode	IV neu per D_i

Beispiele: RC4, A5/1, …, siehe auch eSTREAM project (http://www.ecrypt.eu.org/stream/)

Data Encryption Standard (DES)

Allgemeines

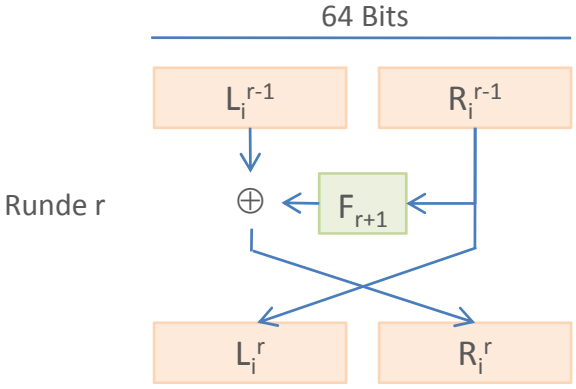
entwickelt von IBM, FIPS-Standard 1977 - 1999

Parameter

Schlüssellänge 64 Bits (nur 56 Bits sind zufällig)

Blocklänge 64 Bits

Feistel-Netzwerk mit 16 Runden



Sicherheit

theoretische Angriffe durch lineare- und differentielle Kryptoanalyse

relativ kleine Schlüssellänge → Bruteforce 2^{56} (DES Challenge III benötigte etwa 22 Stunden)

relativ kleine Blocklänge → je nach Modus kann ebenfalls zum Problem werden

DES sollte *nicht* mehr verwendet werden

Triple-DES (3-DES)

Allgemeines

FIPS-Standard seit 1999

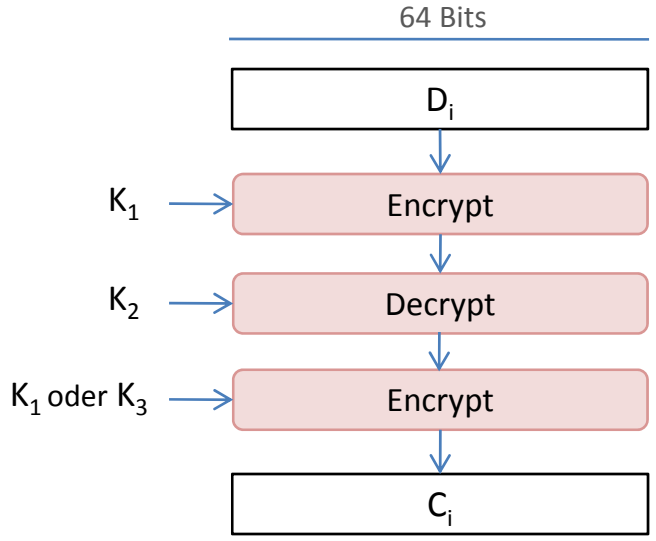
als Ersatz für DES

Parameter

2 oder 3 unabhängige DES-Schlüssel

3 sequentielle Ausführungen von DES

zweite Ausführung ist Entschlüsselung



Sicherheit

effektive Schlüssellängen: 112 Bits bei 2 Schlüsseln 168 Bits bei 3 Schlüsseln

CPA-Angriff auf Variante mit 2 Schlüsseln mit Laufzeit 2^{56} bei 2^{56} bekannten (D_i, C_i)-Paaren

meet-in-the-middle Angriff auf Variante mit 3 Schlüsseln mit Laufzeit 2^{112}

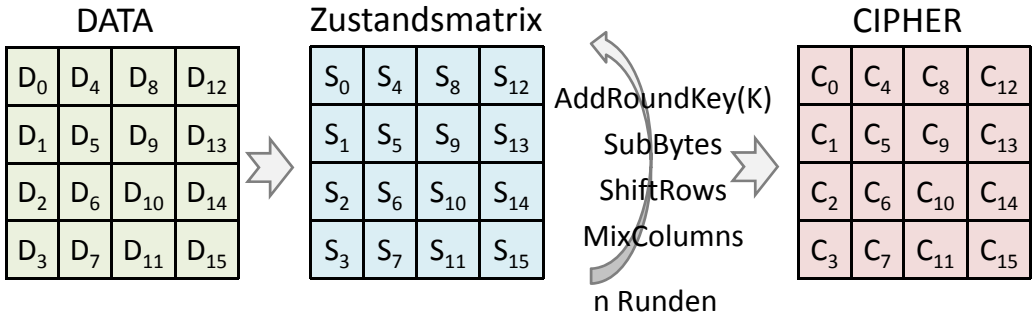
3-DES mit 3 Schlüsseln wird immernoch als praktisch sicher angesehen (aber nicht sehr effizient)

Advanced Encryption Standard (AES)

Allgemeines

NIST-Standard seit 2000

gedacht als Ersatz für DES/3-DES



Parameter

Schlüssellängen 128, 192 oder 256 Bits

Blocklänge 128 Bits (als 4x4-Bytes Matrix)

Substitution-Permutation-Netzwerk mit 10, 12 oder 14 Runden je nach Schlüssellänge

Sicherheit

bekannte Angriffe nur auf *reduzierte* Rundenanzahl

Laufzeit 2^{72} auf 6 Runden AES-128, 2^{188} auf AES-192, 2^{204} auf AES-256

kein nennenswertes Risiko für Praxis

AES ist somit sicher

Asymmetrische Verschlüsselung

Zahlentheoretische Probleme

komplexitätstheoretische Sicherheit, Beschränkung der Rechenleistung des Angreifers

Angreifer = Algorithmus mit polynomieller Laufzeit (in der Länge des Inputs)

Verfahren mit praktischer Relevanz

$N = pq$

Faktorisierung von Primzahlen

RSA

Paillier

$y = g^x \bmod p$

Diskrete Logarithmen in Gruppen primer Ordnung

ElGamal

ECC

RSA Verschlüsselung

Algorithmen ("Textbook RSA")

KeyGen lange Primzahlen (P, Q); Modulus N = PQ; Exponenten e und d mit ed = 1 mod (P-1)(Q-1)
PK = (N, e), SK = (N, d)

Encrypt DATA $\in$ [1, N-1], CIPHER = $(DATA)^e$ mod N

Decrypt CIPHER $\in$ [1, N-1], DATA = $(CIPHER)^d$ mod N

RSA in der Praxis

Textbook RSA ist nicht sicher

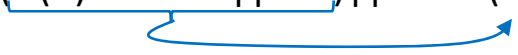
Probleme bei kurzen Exponenten e und bei gemeinsamen Moduli N

Textbook RSA ist deterministisch (gleiches DATA führt zum gleichen CIPHER)

In Praxis wird "RSA mit Padding" verwendet, Teil von PKCS#1-Standard (aktuell in Version 2.1)

RSAES-PKCS1-v1_5 bietet nur CPA-Sicherheit $(R||DATA)^e$

RSAES-OAEP bietet auch CCA-Sicherheit $((G(R) \oplus DATA||0..0)||R \oplus H(\ ))^e$



Algorithmen zur Faktorisierung

n (= log N) Länge von N = PQ
finde die Primzahlen P und Q

Bruteforce Angriff

probiere alle Zahlen im Interval [2, …, $\sqrt{N}$]

Laufzeit $O(N^{1/2} \cdot (\log N)^c) \approx O(2^{n/2} n^c)$, somit exponentiell im Sicherheitsparameter n

→ P und Q sollen ungefähr gleiche Länge haben, also |P| = |Q| = n/2

Pollard's P-1 Methode

effizient falls P-1 nur kleine Primfaktoren hat

Laufzeit O(B n/(log B)) mit B größer als der größter Faktor von P-1

→ P-1 und Q-1 sollen keine kleinen Primfaktoren enthalten, also sog. *strong primes* sein

→ P = 2P'+1 und Q = 2Q'+1 mit großen Primzahlen P' und Q', also $|P'| = |Q'| \approx n/2 - 1$
Laufzeit wäre dann immernoch exponentiell, ungefähr $O(2^{n/2})$

Pollard's rho Methode

Laufzeit $O(N^{1/4} \cdot (\log N)^c) = O(2^{n/4} n^c)$, etwas besser als Bruteforce

Quadratic Sieve

Laufzeit ungefähr $2^{O((n \cdot (\log n))1/2)}$, somit sub-exponentiell im Sicherheitsparameter n

General Number Field Sieve

etwas besser als Quadratic Sieve, immernoch mit sub-exponentieller Laufzeit

Faktorisierung von N
in polynomieller Zeit $O(n^c)$
ist nicht bekannt

ElGamal Verschlüsselung

Algorithmen

KeyGen Primzahlen P und Q mit Q teilt P-1

zyklische Gruppe $\mathbb{G}$ = <g> der Ordnung Q als Untergruppe von $\mathbb{Z}^*_P$ = [1,…,P-1]

PK = (y), SK = (y, x) mit $y = g^x \bmod P$

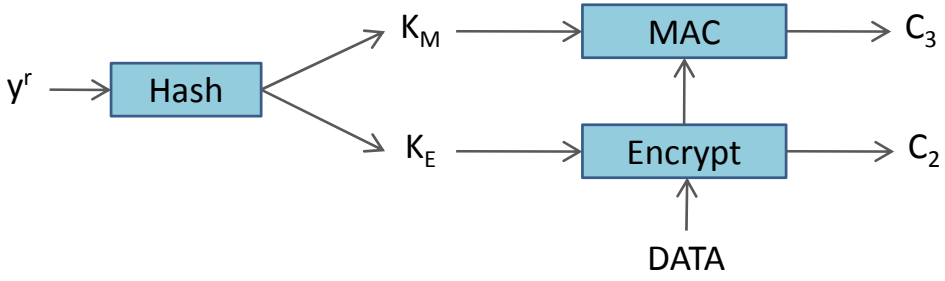
Encrypt DATA $\in \mathbb{G}$, wähle r zufällig aus [1, Q-1], CIPHER = $(C_1, C_2) = (g^r \bmod P, y^r \cdot \text{DATA} \bmod P)$

Decrypt CIPHER = (C_1, C_2), DATA = $C_2 / C_1^x \bmod P$

ElGamal in der Praxis

ElGamal bietet nur CPA-Sicherheit

In Praxis wird oft DHIES verwendet, Teil von IEEE P1363-2000 Standard



CIPHER = (C_1, C_2, C_3)

Algorithmen zur Berechnung von DLog

Bruteforce Angriff

probiere alle Zahlen im Interval [1, …, Q-1]

Laufzeit $O(Q) \approx O(2^n)$, somit exponentiell im Sicherheitsparameter n

→ Q soll hinreichend lang sein

Baby-Step/Giant-Step

Laufzeit $O(Q^{1/2} \cdot (\log Q)^c) \approx O(2^{n/2} n^c)$, etwas besser als Bruteforce

Pohlig-Hellman

anwendbar wenn die Ordnung der Gruppe, also Q, bekannte Faktoren hat

Laufzeit $O(Q_{max}^{1/2} \cdot (\log Q)^c)$ wo Q_{max} ist der größte Faktor von Q

→ es empfiehlt sich daher Q als Primzahl zu wählen

Index Calculus

anwendbar nur in der zyklischen Gruppe $\mathbb{Z}^*_P = [1,\ldots,P-1]$

Laufzeit ungefähr $2^{O((n \cdot (\log n))1/2)}$, somit sub-exponentiell im Sicherheitsparameter n

→ es empfiehlt sich daher die Untergruppe G von $\mathbb{Z}^*_P$ zu verwenden

General Number Field Sieve

etwas besser als Index Calculus, immernoch mit sub-exponentieller Laufzeit

n (= log Q) Länge von Q
gegeben $y = g^x \bmod P$
finde x

Berechnung von DLog
in polynomieller Zeit $O(n^c)$
ist nicht bekannt

Elliptic Curve Cryptography (ECC)

Grundlagen von ECC

gerechnet wird auf elliptischen Kurven über endliche Körper $\mathbb{F}$ der Ordnung P

(1) P ist prim E(x,y) : $y^2 = x^3 + ax + b$ mit $4a^3 + 27b \neq 0$

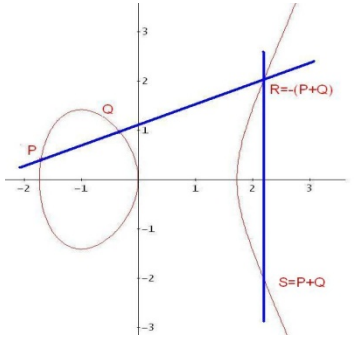
(2) P ist zweier Potenz (2^k) E(x,y) : $y^2 + xy = x^3 + ax^2 + b$ mit $b \neq 0$

die Menge aller Punkte auf E(x,y) zusammen mit abstraktem Punkt O bildet eine *kommutative* Gruppe E($\mathbb{F}$)

DLog in ECC

$E(\mathbb{F}_P)$ hat zyklische Untergruppen $\mathbb{G}$ = <G> := {O, G, …, (Q-1)·G} primer Ordnung Q

erzeugender Element G ist ein Punkt auf E(x,y)



DLog-Problem gegeben Y = X·G mit X aus [1, …, Q-1] finde X

ECC-Verschlüsselung

Varianten von ElGamal und DHIES basierend auf elliptischen Kurven

die schnellsten Algorithmen für DLog in ECC benötigen Laufzeit von ca. $O(Q^{1/2}) = O(2^{n/2})$

→ Q in ECC kann kleiner im Vergleich zu Q in $\mathbb{Z}_Q$ gewählt werden (|Q| = 160 Bits, Laufzeit des Angriffs $O(2^{80})$)

die Wahl von Kurven ist ebenfalls wichtig (IEEE P1363, SEC von Certicom, FIPS 186-2 von NIST)

→ wegen Sicherheit und Effizienz

Sicherheitsparameter- Zusammenfassung

Sicherheitsparameter für symmetrische Verfahren

Schlüssellänge

soll hinreichend lang sein um Bruteforce-Angriffen praktisch zu begegnen

Sicherheitsparameter für asymmetrische Verfahren

RSA-basierte Verfahren

Länge des Modulus N = PQ

bestimmt durch die Längen der Primzahlen P und Q

DLog-basierte Verfahren

Ordnung P der Gruppe $\mathbb{Z}_P$

bestimmt durch die Länge der Primzahl P

Ordnung Q der Untergruppe $\mathbb{G} \subset \mathbb{Z}_P$

bestimmt durch die Länge der Primzahl Q, impliziert die Schlüssellänge

ECC-basierte Verfahren

Ordnung P des Körpers $\mathbb{F}_P$

bestimmt durch die Länge von P (P ist entweder eine Primzahl oder eine zweier Potenz)

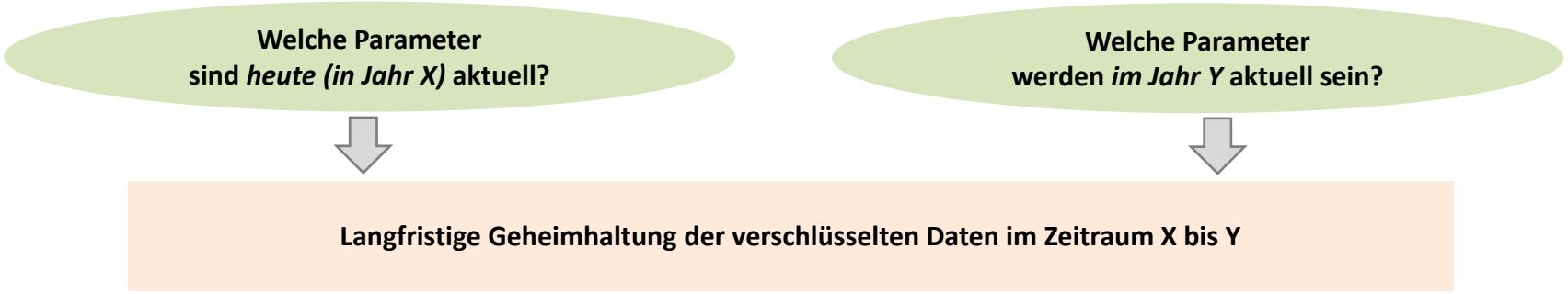
Ordnung Q der Untergruppe $E(\mathbb{F}_P)$

bestimmt durch die Länge der Primzahl Q

Parameter für die elliptische Kurve E(x,y) (sind meistens vorgegeben)

Zeitgemäße Verschlüsselung

Ziel : **Kontinuierliche Anpassung der Sicherheitsparameter in Abhängigkeit der modernen Technologie (Zeit)**



Wie berechnet man aktuelle Parameter?

A. K. Lenstra, E. R. Verheul: *„Selecting Cryptographic Key Sizes“* (Public Key Cryptography 2000, http://www.win.tue.nl/~klenstra/key.pdf)

aktualisiert in

A. K. Lenstra: *„Key Lengths“* (Kapitel im Handbook of Information Security, http://www.keylength.com/biblio/Handbook_of_Information_Security_-_Keylength.pdf)

Gleichungen zur Berechnung der Längen in Abhängigkeit der geschätzten MIPS (Rechenleistung) im Jahr X basierend auf kryptoanalytischen Erfahrungen mit bekannten Verschlüsselungsverfahren

vorausgesetzt, dass ein *signifikanter* kryptoanalytischer Durchbruch ausbleibt (auch in der Technologieentwicklung)

Wahl der Parameter – Praktisches *How-To*

Zahlreiche Empfehlungen

vom NIST

in 2007: *Recommendation for Key Management*, Special Publication 800-57 Part 1
http://csrc.nist.gov/groups/ST/toolkit/key_management.html

in 2009: *Cryptographic Key Management Project* (noch als draft version)
http://csrc.nist.gov/groups/ST/toolkit/key_management.html

vom ECRYPT

in 2009: *ECRYPT Yearly Report on Algorithms and Keysizes (2008-2009)*, Rev. 1.0 (momentan aktuellste)
http://www.ecrypt.eu.org/documents.html

von BNetzA

in 2009: *Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung*
http://www.bundesnetzagentur.de/media/archive/15549.pdf (bezieht sich auf Signaturen)

Übersichtsprojekte im Internet

www.keylength.com (in Englisch)

keylength.com

BlueKrypt | Cryptographic Key Length Recommendation

In most cryptographic functions, the key length is an important security parameter. Both academic and private organizations provide recommendations and mathematical formulas to approximate the minimum key size requirement for security. Despite the availability of these publications, choosing an appropriate key size to protect your system from attacks remains a headache as you need to read and understand all these papers. This web site implements mathematical formulas and summarizes reports from well-known organizations allowing you to quickly evaluate the minimum security requirements for your system. You can also easily compare all these techniques and find the appropriate key length for your desired level of protection.

The lengths provided here are designed to resist mathematic attacks; they do not take algorithmic attacks, hardware flaws, etc. into account.

Choose a method

- Lenstra and Verheul Equations (2000)
- Lenstra Updated Equations (2004)
- ECRYPT II Recommendations (2009)
- NIST Recommendations (2007)
- DCSSI Recommendations (2007)
- Fact Sheet NSA Suite B Cryptography (2009)
- Network Working Group RFC3766 (2004)
- BSI Recommendations (2009)

Authors: Damien Giry, Philippe Bulens
Approved by Prof. Jean-Jacques Quisquater
Contact: keylength@bluekrypt.com

I would like to thank Prof. Arjen K. Lenstra for his kind authorization and comments.
Surveys of laws and regulations on cryptology: Crypto Law Survey / Digital Signature Law Survey.

Lenstra Updated Equations (2004)

Computation for year 2030

Options
Date until when user trusts DES: 1982 (default)
Double Moore factorizing law (default)

Year	Symmetric	Asymmetric		Discrete Logarithm		Elliptic Curve	Hash
		Optimistic	Conservative	Key	Group		
2028	87	1633	1958	174	1633	174	174
2029	88	1665	2010	175	1665	175	175
2030	**88**	**1698**	**2063**	**176**	**1698**	**176**	**176**
2031	89	1732	2118	178	1732	178	178
2032	90	1765	2173	179	1765	179	179

ECRYPT II Recommendations (2009)

All key sizes are provided in bits. These are the minimal sizes for security.

Level	Protection	Symmetric	Asymmetric	Discrete Logarithm		Elliptic Curve	Hash
				Key	Group		
1	Attacks in "real-time" by individuals *Only acceptable for authentication tag size*	32	-	-	-	-	-
2	Very short-term protection against small organizations *Should not be used for confidentiality in new systems*	64	816	128	816	128	128
3	Short-term protection against medium organizations, medium-term protection against small organizations	72	1008	144	1008	144	144
4	Very short-term protection against agencies, long-term protection against small organizations *Smallest general-purpose level,* *Use of 2-key 3DES restricted to 2^{40} plaintext/ciphertexts,* *protection from 2009 to 2012*	80	1248	160	1248	160	160
5	Legacy standard level *Use of 2-key 3DES restricted to 10^8 plaintext/ciphertexts,* *protection from 2009 to 2020*	96	1776	192	1776	192	192
6	Medium-term protection *Use of 3-key 3DES,* *protection from 2009 to 2030*	112	2432	224	2432	224	224
7	Long-term protection *Generic application-independent recommendation,* *protection from 2009 to 2040*	128	3248	256	3248	256	256
8	"Foreseeable future" *Good protection against quantum computers*	256	15424	512	15424	512	512

Ausblick

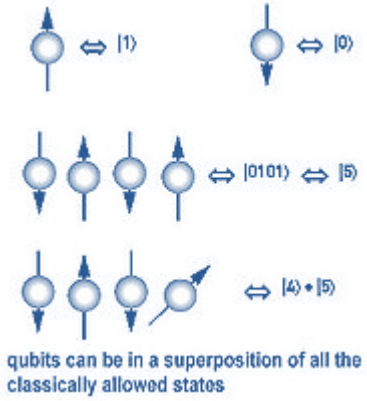
Trend I: Quanten-Rechner

großes Risiko für moderne Kryptoverfahren; bedingt durch die Algorithmen von

P. W. Shor: *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. In SIAM J. on Computing, 26/1997 (http://de.wikipedia.org/wiki/Shor-Algorithmus)

zur Faktorisierung und Berechnung von Diskreten Logarithmen mit polynomieller Laufzeit

noch keine praktische Relevanz



(Quelle: Wikipedia)

Trend II: Post-Quantum Kryptographie

gegen Shor's Algorithmus resistente Verschlüsselungsverfahren

z.B. NTRU, McEliece, …

benötigen keine Quanten-Rechner!

aktuelles Forschungsthema (auch in Darmstadt)

Workshop PQCrypto 2010 (http://pqc2010.cased.de/)