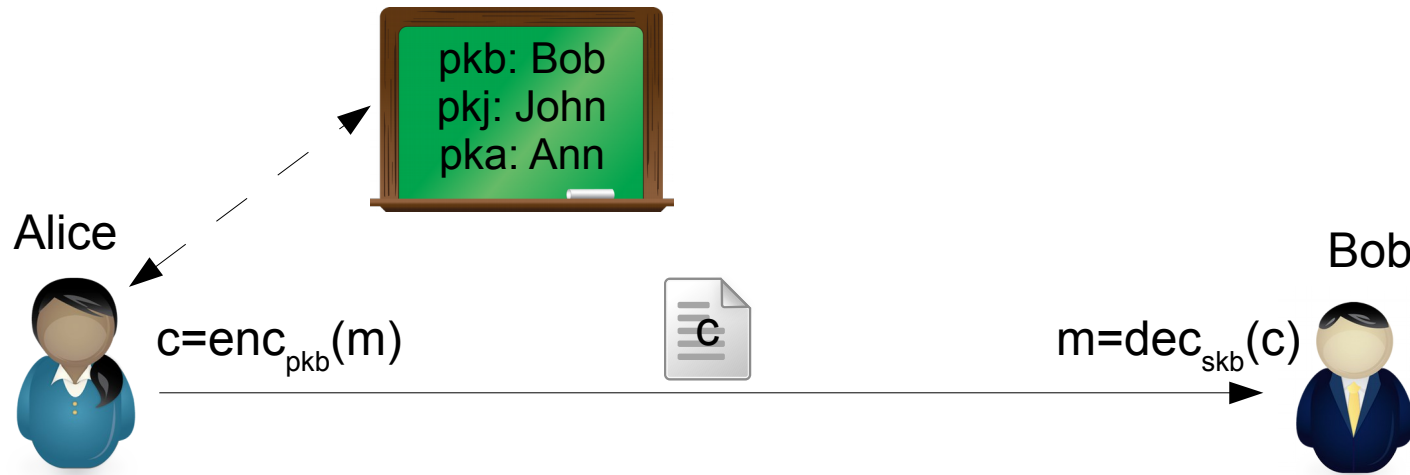


Themen zur Computersicherheit

Asymmetrische Chiffren

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

Asymmetrische Verschlüsselungsverfahren



- $\text{enc}: eK \times \Sigma_1^* \rightarrow \Sigma_2^*$, $\text{dec}: dK \times \Sigma_2^* \rightarrow \Sigma_1^*$
- Schlüssel $(pk, sk) \in eK \times dK$ mit $pk \neq sk$
 - pk heißt öffentlicher Schlüssel (public key)
 - sk heißt privater Schlüssel (private / secret key)
 - sk kann aus pk nicht effizient berechnet werden
- Wenn Alice eine vertrauliche Nachricht an Bob senden will,
 - verschlüsselt sie die Nachricht mit Bobs öffentlichen Schlüssel & sendet sie an Bob
 - Bob entschlüsselt sie mit seinem privaten Schlüssel
- Verfahren
 - Rivest-Shamir-Adleman (RSA)
 - Diffie-Hellman (DH)

Primzahlen, teilerfremde Zahlen

- Sei p eine Primzahl
- $\mathbf{Z}/p\mathbf{Z} = \text{GF}(p)$ ist ein endlicher (Galois-)Körper
 - $\mathbf{Z}/p\mathbf{Z}$ ist eine additive Gruppe
 - $\{1, \dots, p-1\}$ ist eine multiplikative Gruppe in $\mathbf{Z}/p\mathbf{Z}$
 - $x^{p-1} = 1 \pmod p$ für alle $0 < x < p$
- Primzahltests
 - kleine Primzahlen: Sieb des Eratosthenes
 - große Primzahlen: z.B. Rabin-Miller (probabilistisch)
- ggT, Teilerfremdheit
 - a, b sind teilerfremd, wenn $\text{ggT}(a,b) = 1$
 - erweiterter Euklidischer Algorithmus: $(g,u,v) = \text{EE}(a,b)$ mit $g = \text{ggT}(a,b) = u \cdot a + v \cdot b$
 - Inverse mod p : $\text{EE}(a,p) = (1, a^{-1} \pmod p, v)$
 - ebenso gilt für beliebige a, n mit $\text{ggT}(a,n) = 1$: $\text{EE}(a,n) = (1, a^{-1} \pmod n, v)$
- Satz von Euler und Folgerungen
 - $\phi(n)$ ist die Zahl der zu n teilerfremden Zahlen kleiner n (Eulersche Phi-Funktion)
 - $\text{ggT}(a,n) = 1 \Rightarrow a^{\phi(n)} = 1 \pmod n$
 - \Rightarrow kleiner Fermat: p prim: $a^{p-1} = 1 \pmod p$ für alle $a < p$
 - (*): p, q prim: $a^{k\phi(p \cdot q)+1} = a \pmod{(p \cdot q)}$ für alle a
- Wieviele Primzahlen gibt es?
 - unendlich viele
 - Primzahlen sind häufig: Es gibt ungefähr $n/(\ln n)$ Primzahlen kleiner n
 - Beispiel: es gibt ca 2^{1013} Primzahlen der Länge 1024 Bit

RSA

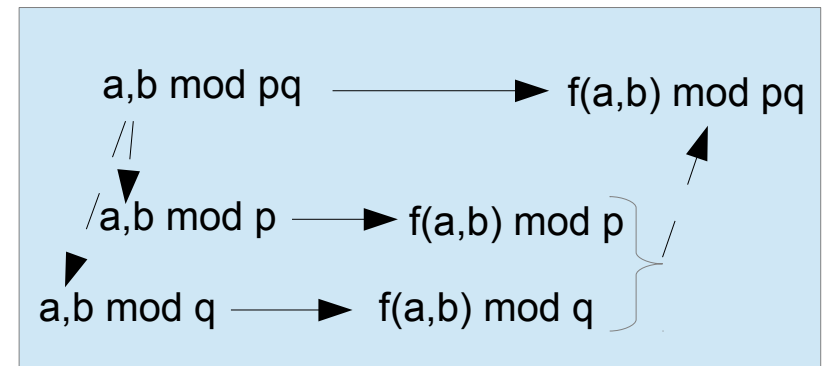
- seinen p, q Primzahlen
- $n = p \cdot q$
- $\phi(n) = (p-1) \cdot (q-1)$ die Zahlen in $\mathbf{Z}/n\mathbf{Z}$ die zu p und q teilerfremd sind
- (n,e) ist ein öffentlicher Schlüssel, wenn e teilerfremd zu $\phi(n)$
- (p,q,d) mit $d = e^{-1} \bmod \phi(n)$ ist der dazugehörige private Schlüssel
 - $\Rightarrow e \cdot d = 1 + k \cdot \phi(n)$
- für Nachricht $m < n$ gilt:
 - $\text{rsa-enc}_{(n,e)}(m) = m^e \bmod n$
 - $\text{rsa-dec}_{(p,q,d)}(c) = c^d \bmod n$
 - $\text{rsa-dec}_{(p,q,d)}(\text{rsa-enc}_{(n,e)}(m)) = \text{rsa-dec}_{(p,q,d)}(m^e \bmod n) = (m^e)^d \bmod n = m^{ed} \bmod n = m \bmod n$
 - wegen Folgerung (*) aus Satz von Euler:
 - (*): p,q prim: $a^{k\phi(p \cdot q)+1} = a \bmod (p \cdot q)$ für alle a

Beispiel

- $p = 11$
- $q = 5$
- $n = p \cdot q = 55$
- $\phi(n) = (p-1) \cdot (q-1) = 10 \cdot 4 = 40$
- $e = 3$ -- $\text{ggT}(3,40) = 1$
- $3 \cdot d = 1 + k \cdot 40$
 - $k = 1$: 41 kein Vielfaches von 3
 - $k = 2$: $81 / 3 = 27 \Rightarrow d = 27$
- öffentlicher Schlüssel (55,3)
- privater Schlüssel (11,5,27)
- Nachricht $m = 50$
- $\text{enc}_{(55,3)}(m) = 50^3 \bmod 55 = 40 \bmod 55$
- $\text{dec}_{(11,5,27)}(40) = 40^{27} \bmod 55 = 50 \bmod 55$

Der Chinesische Restesatz (Chinese Remainder Theorem / CRT)

- erste Erwähnung durch chin. Mathematiker Sun Tzu aus dem 1. Jahrhundert.
- CRT (Version mit 2 Primfaktoren)
 - Sei $n = p \cdot q$ mit p, q prim, dann gibt es zu zwei Zahlen a und b genau ein $x \in \mathbf{Z}/n\mathbf{Z}$ mit $x = a \pmod p$ und $x = b \pmod q$
- Garners Formel
 - $x = (((a - b) \cdot (q^{-1} \pmod p)) \pmod p) \cdot q + b$
- Relevanz
 - eine Rechnung $\pmod n$ kann ersetzt werden durch
 - zwei Rechnungen $\pmod p$ bzw $\pmod q$ und
 - einer Instanz von Garners Formel
 - lohnt sich wenn Rechnungsaufwand superlinear in der Zahlenlänge ($\text{ld } n$) ist
 - z.B. Langzahlmultiplikation
 - naiv: $O((\text{ld } n)^2)$
 - Karazuba $O((\text{ld } n)^{\text{ld } 3})$
 - Schönhage-Strassen $O((\text{ld } n) \cdot (\log(\text{ld } n)) \cdot (\log(\log(\text{ld } n))))$
 - Fürer (2007); Harvey et al (2014): $O((\text{ld } n) \cdot (\log(\text{ld } n)) \cdot 2^{3 \log^*(\text{ld } n)})$
- RSA: der private Schlüssel wird oft im CRT Format angegeben:
 - (p, q, d_p, d_q, q^{-1}) mit $d_p = d \pmod p$ und $d_q = d \pmod q$



Einschränkungen und Fallstricke

- Die Sicherheit von RSA beruht auf der Schwierigkeit große Zahlen zu faktorisieren
- Der numerische Wert der Nachricht $\text{INT}(m)$ muss kleiner als der Modulus n sein.
- Für kleines $\text{INT}(m)$ und kleines e kann $\text{INT}(m)^e$ kleiner als n sein. Dann kann $\text{INT}(m)$ durch Wurzelziehen in den natürlichen Zahlen berechnet werden.
- Verschiedene RSA Moduli dürfen keine gemeinsamen Teiler haben.
 - ggTs zu berechnen ist viel effizienter möglich als Zahlen zu faktorisieren
 - Nadia Heninger, Zakir Durumeric, Eric Wustrow, J. Alex Halderman: „Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices“ Proc. 21st USENIX Security Symposium, 2012
- Zufallsprimzahlgeneratoren müssen viele verschiedene Primzahlen generieren

Homomorphie Eigenschaft von RSA

Es gilt

- $\text{rsa-enc}_{(n,e)}(m_1 \cdot m_2) = \text{rsa-enc}_{(n,e)}(m_1) \cdot \text{rsa-enc}_{(n,e)}(m_2) \text{ mod } n,$

- denn

- $\text{rsa-enc}_{(n,e)}(m_1 \cdot m_2)$

- $= (m_1 \cdot m_2)^e \text{ mod } n$

- $= m_1^e \cdot m_2^e \text{ mod } n$

- $= (m_1^e \text{ mod } n) \cdot (m_2^e \text{ mod } n) \text{ mod } n$

- $= \text{rsa-enc}_{(n,e)}(m_1) \cdot \text{rsa-enc}_{(n,e)}(m_2) \text{ mod } n$

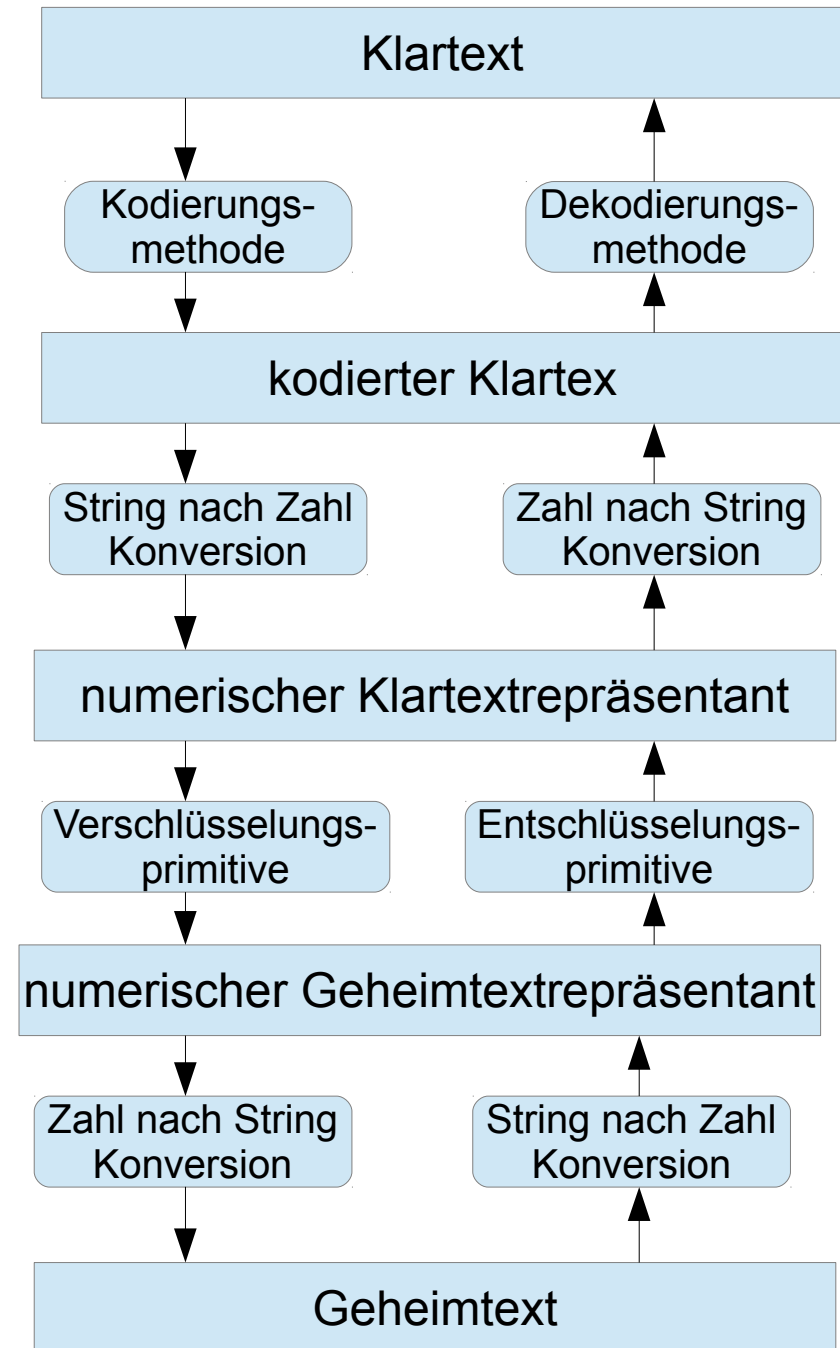
- die RSA Verschlüsselung ist homomorph bzgl der Multiplikation modulo n
- die RSA Verschlüsselung ist nicht homomorph bezüglich der Addition oder xor

Verschlüsselung nach PKCS #1 v2.2

- beschreibt Verschlüsselungsschemata
 - RSAES-OAEP (Optimal Asymmetric Encryption Padding)
 - RSAES-PKCS-v1_5

Sicherheit

- RSAES-OAEP ist sicher gegen adaptive CCA Attacken, falls die Verschlüsselungsprimitive schwer (partiell) zu invertieren ist
- RSAES-PKCS-v1_5 nicht für neue Anwendungen empfohlen, da CCA-Angriffe bekannt (Bleichenbacher-Attacke)



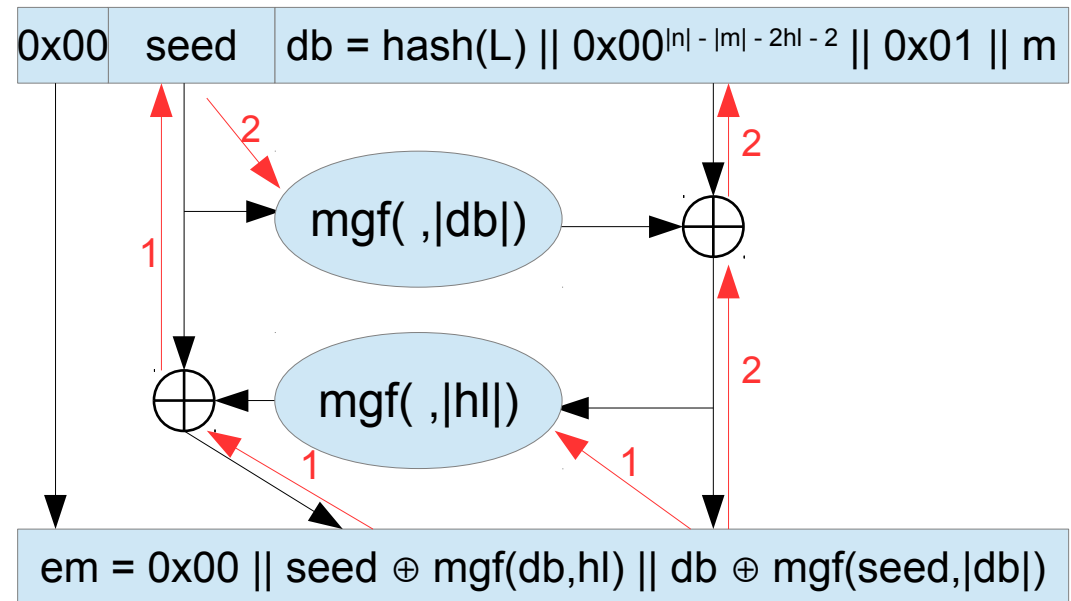
PKCS #1 Kodierungsoperationen

PKCS #1- v1_5 Kodierung

- Eingabe
 - Länge des Modulus: $|n|$ in Bytes
 - Klartext: m mit $|m| \leq |n| - 11$
- Ausgabe:
 - kodierter Klartext:
 - $0x00 \parallel 0x02 \parallel PS \parallel 0x00 \parallel m$
 - mit PS besteht aus $|n| - |m| - 3$ pseudozufällig generierten Bytes ungleich $0x00$
- Umkehrung
 - m beginnt nach erstem $0x00$ Byte nach dem zweiten Byte

PKCS #1- v2.2 OAEP Kodierung

- Eingabe
 - Länge des Modulus: $|n|$ in Bytes
 - Hashfunktion: $\text{hash}: \Sigma \rightarrow \Sigma^{hl}$
 - maskenerzeugende Funktion: $\text{mgf}: \Sigma^* \times \mathbf{N} \rightarrow \Sigma^*$
 - optionales Label: $L \in \Sigma^*$
- seed: Zufallsbytestring der Länge hl



- Ausgabe
 - kodierter Klartext: em
- Umkehrung
 - $\text{seed} = \text{seed} \oplus \text{mgf}(db, hl) \oplus \text{mgf}(db \oplus \text{mgf}(\text{seed}, |db|), hl)$
 - $db = db \oplus \text{mgf}(\text{seed}, |db|) \oplus \text{mgf}(\text{seed}, |db|)$
 - m beginnt nach dem ersten $0x01$ Byte hinter Byte $2 \cdot hl$

Diffie-Hellman Schlüsselaustausch

- PKCS #3
- Sei p prim, dann heißt k der *diskrete Logarithmus* von n zur Basis a modulo p wenn $a^k = n \text{ modulo } p$
- Es gibt kein effizientes Verfahren um für große Primzahlen p diskrete Logarithmen zu berechnen
- Protokoll: seinen p prim und $g \in \mathbf{Z}/p\mathbf{Z}$ öffentlich

Alice



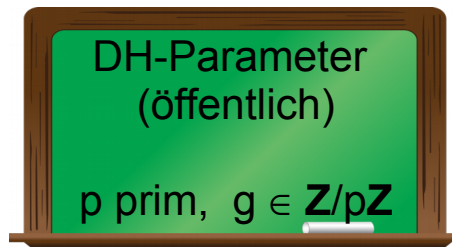
privater Schlüssel von Alice

wählt Schlüssel $a \in \mathbf{Z}/p\mathbf{Z}$
sendet $g^a \text{ mod } p$ an Bob

öffentlicher Schlüssel von Alice

berechnet
 $k = (g^b)^a \text{ mod } p = g^{ab} \text{ mod } p$

$c = \text{enc}_k(m)$



Bob



privater Schlüssel von Bob

wählt Schlüssel $b \in \mathbf{Z}/p\mathbf{Z}$
sendet $g^b \text{ mod } p$ an Alice
berechnet
 $k = (g^a)^b \text{ mod } p = g^{ba} \text{ mod } p$

öffentlicher Schlüssel von Bob

$m = \text{dec}_k(c)$

Ein paar Begriffe aus der Gruppentheorie

- p prim, dann ist $Z_p^* = \{1, \dots, p-1\}$ die multiplikative Gruppe modulo p
- für $a \in Z_p^*$ ist $\{a^1, \dots, a^{p-1}\}$ die von a erzeugte Untergruppe von Z_p^* ,
- $|\{a^1, \dots, a^{p-1}\}|$ heißt die *Ordnung* von a ,
- die Ordnung ist immer ein Teiler von $p-1$
- wenn a die Ordnung $p-1$ hat heißt a *primitives Element* von Z_p^*

Schlüsselerzeugung

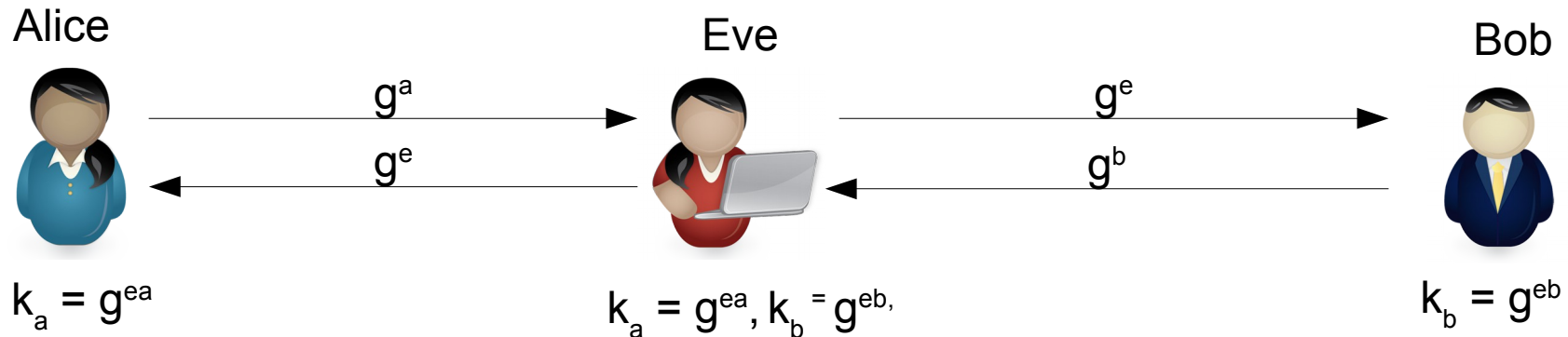
- Das gemeinsame Geheimnis $k = g^{ab} \bmod p$ muss nicht unbedingt Element des Schlüsselraums sein.
- Deshalb wird der eigentliche Schlüssel oft mit Hilfe einer (Schlüsselableitungsfunktion basierend auf einer) kryptographischen Hashfunktion von k abgeleitet.

Die Wahl des DH-Parameters g

- $\{ g^1 \bmod p, g^2 \bmod p, g^3 \bmod p, \dots \}$ muss sehr große Menge sein
- wenn g primitives Element dann ist $|\{g^i \bmod p \mid i < p\}| = p-1$
- Wie kann man verhindern, dass g mit kleiner Ordnung gewählt wird?
 - wähle p so, dass $p-1$ außer der 2 keine kleinen Teiler hat
 - sichere Primzahlen: $p = 2 \cdot q + 1$ für q prim
 - dann hat Z_p^* nur Untergruppen der Ordnung 1, 2, q und $2 \cdot q$
 - \Rightarrow für $\alpha \in \{2, \dots, p-2\}$ mit $\alpha^2 \not\equiv 1 \pmod p$ und $\alpha^2 \not\equiv p-1 \pmod p$, setze $g = \alpha^2$
 - (es gibt einen einfachen Test um festzustellen ob $h \bmod p$ eine Quadratzahl ist)
- Effizienter Schlüsselaustausch
 - nutze g , das kleinere Untergruppe der Ordnung q erzeugt
 - dann gilt: $g^x \bmod p = g^{x \bmod q} \bmod p$
 - RFC2631, ANSI X9.42
 - wähle $p = N \cdot q + 1$, mit q prim
 - wobei die Größe von q die Größe des erzeugbaren Schlüsselraum bestimmt
 - wähle g so, dass $g = \alpha^N \bmod p$, $g \not\equiv 1 \pmod p$, $g \not\equiv p-1 \pmod p$
 - die öffentlichen DH Parameter sind dann (p, q, g)

Angriffe auf DH

- Man in the middle Angriff:
 - Eve fängt Protokoll zwischen Alice und Bob ab.



- Gegenmaßnahme: Signieren der Nachrichten (→ später in der Vorlesung)
- Manipulation der DH Parameter
 - z.B. g mit geringer Ordnung
 - Gegenmaßnahme:
 - verifizieren von p und g (bzw q)
 - Längentest von p und q
 - q teilt p ?
 - $g \neq 1 \pmod p$, $g^q = 1 \pmod p$
 - verifizieren von empfangenem g^a , bzw g^b
 - z.B. für alle g^b muss gelten $1 < g^b \pmod p < p$ und $(g^b)^q = 1 \pmod p$

Statische und Ephemere DH Verfahren

- Wie lange sollen erzeugte Schlüssel benutzt werden?
 - für immer
 - nur für eine Nachricht
- Was passiert wenn ein privater Schlüssel entdeckt wird?
 - Können dann alle alten Nachrichten entschlüsselt werden?
 - Können dann alle zukünftigen Nachrichten entschlüsselt werden?
- statisches DH Verfahren: Schlüssel bleibt konstant
 - static-static: Alice und Bob halten Schlüssel konstant
- ephemeres (flüchtig) DH Verfahren: Schlüssel ändern für jede Nachricht
 - static-ephemeral: Bob ändert den Schlüssel für jede Nachricht
 - ephemeral-static: Alice ändert den Schlüssel für jede Nachricht
 - ephemeral-ephemeral: Alice & Bob ändern die Schlüssel für jede Nachricht
 - → perfect forward privacy (PFP)
- Verschlüsselung:
 - wenn beim static-ephemeral DH Verfahren Alice ihren öffentlichen Schlüssel in einem öffentlichen Verzeichnis veröffentlicht, dann kann DH ähnlich wie RSA zum Verschlüsseln genutzt werden
 - sonst muss Alice ihren öffentlichen Schlüssel zusammen mit der verschlüsselten Nachricht an Bob schicken.

Varianten von DH: ECDH

- Elliptic Curve Diffie-Hellman (ECDH)
- Elliptische Kurve über endlichen Körper $GF(p^n)$: $E = \{ (x,y) \in GF(p^n)^2 \mid y^2 = x^3 + ax + b \} \cup O$
- auf Elliptischen Kurven lässt sich eine (multiplikative) Gruppe beschreiben wobei
 - $(x_1, y_1) \cdot (x_2, y_2) = (x_3, y_3)$ mit
 - $x_3 = ((y_2 - y_1) / (x_2 - x_1))^2 - x_1 - x_2$; $y_3 = y_1 + ((y_2 - y_1) / (x_2 - x_1)) \cdot (x_1 - x_3)$ für $x_1 \neq x_2$ oder $y_1 \neq y_2$
 - $x_3 = ((3x_1^2 + a) / (2y_1))^2 - 2x_1$; $y_3 = -y_1 + ((3x_1^2 + a) / 2y_1) \cdot (x_1 - x_3)$ für $x_1 = x_2$ und $y_1 = y_2$
 - Potenzierung (bzgl. A in Gruppe): $A^n = A \cdot A \cdot \dots \cdot A$ (n-faches Produkt)
 - (diskreter) Logarithmus zur Basis A von X: finde n so das $X = A^n$
- das Problem des diskreten Logarithmus bei elliptischen Kurven ist schwerer als bei Standard-DH → kleiner Modulus
- Domain Parameter (p, a, b, G, n, h)
 - $G \in E$ ist Generator, n Ordnung von G in E, h Kofaktor von G
- Schlüssel
 - privater Schlüssel: $d \in \{1, \dots, n-1\}$
 - öffentlicher Schlüssel: $P = G^d \in E$
- Bemerkung: In der Literatur wird „•“ oft als Addition „+“ bezeichnet und dann die n-fache Addition durch ein skalares Produkt bezeichnet

Sind DH bzw EC Kryptographie sicher?

Gerücht: Die NSA kann DH knacken.

- Beobachtung. sehr viele Server nutzen die gleichen DH Parameter
 - Die Berechnung von $\log(g^a \bmod p)$ hängt zu einem großen Teil nur von p und g ab.
 - Möglicher Angriff: Vorberechnen von Teilrechnungen aller diskreten Logarithmen bezüglich häufig genutzter Parameter
 - Vermutung NSA hat Ressourcen um so bis zu 1024 bit Schlüssel zu knacken
 - das wäre zusammen mit einem Angriff, der beim Schlüsselaustauschprotokoll eine kleinen DH Modulus (z.B. Logjam) erzwingt ein effektiver Angriff auf DH
 - Literatur:
 - Adirian et al: „Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice“, <https://weakdh.org/imperfect-forward-secrecy-ccs15.pdf>
- „gezinkte“ Primzahlen & SNFS erlauben effiziente Logarithmenberechnung
 - Fried et al: „A kilobit hidden SNFS discrete logarithm computation“
<http://eprint.iacr.org/2016/961>
 - diskreter Log von 1024-Bit Primzahl in 2 Monaten (auf 3000 Kernen) geknackt
 - „gezinkte“ Primzahlen sind sehr selten, lassen sich aber konstruieren
 - Vorschlag (FIPS 186-4): veröffentliche mit p den Seed des PRNG der p erzeugt hat

Es gibt wohl einen (möglichen) Angriff auf ECC per Quantum Computing

- 2015: NSA empfiehlt
 - keine neuen Investitionen in ECC Verfahren zu tätigen
 - und statt dessen in Verfahren zu investieren, die gegen Quantum-Computing sicher sind
- diese NSA Empfehlung ist bei Wissenschaftlern umstritten
- Literatur:
 - Kobitz und Menezes: „A riddle wrapped in an enigma“, <https://eprint.iacr.org/2015/1018.pdf>