

Die wichtigsten Begriffe:

- Sender/Empfänger
- M,P: Plaintext/Text Nachricht oder Text oder Daten die verschlüsselt werden/wurden
- C: Chiffretext
- K: Key, Schlüssel
- $E(M)=C$ Verschlüsseln
- $D(C)=M$ Entschlüsseln
- $D(E(M))=C$ $D(E(M,K),K)=C$
- Symmetrisch: $D(E(M))=M$ Wenn die Verschlüsselung wiederholt angewendet wieder den Klartext ergibt: ROT13, XOR
- Kryptologie besteht aus Kryptographie und Kryptoanalyse
- Code(-ierung)-Steganographie(Verstecken)-Kryptographie
- Transposition: Änderung der Position der Zeichen, Beispiel: Spaltentransposition, Rückwärts
- Substitution: Änderung der Zeichen

Nennen und erläutern Sie vier verschiedene zentrale Aspekte für die Zielsetzungen kryptographischer Verfahren.

1. Vertraulichkeit: Die Nachricht soll für Dritte unlesbar bleiben.
2. Authentikation: Der Empfänger einer Nachricht soll über die Identität des Senders überzeugt werden.
3. Integrität: Der Empfänger soll über die Echtheit der Nachricht überzeugt werden.
4. Zugehörigkeit: Der Sender darf nicht später den Ursprung der Nachricht bestreiten können.

Grundmaxime von Kerckhoff (1880):

Die Sicherheit einer Chiffre darf nicht darauf beruhen, dass der Gegner das benutzte Verfahren nicht kennt.

- die verschlüsselte Nachricht sollte praktisch unknackbar sein
- die Korrespondenten (Sender und Empfänger) dürfen keinen Schaden erleiden, wenn das Chiffriersystem geknackt wurde (zeitweiliger Schutz)
- der Schlüssel muss leicht auswendig zu lernen und veränderbar sein
- die Kryptogramme müssen über Telegraphen übertragbar sein
- der Chiffrierapparat und die Dokumente müssen leicht transportierbar sein
- das System muss einfach zu benutzen sein, und sollte keine übermäßigen geistigen Anstrengungen verlangen

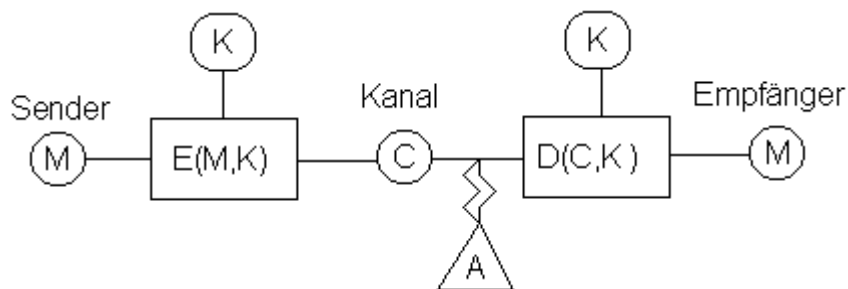
Nennen Sie mindestens drei „Grenzen“ bei dem Einsatz kryptographischer Verfahren.

- Aufwand ist höher als der Wert der Nachricht
- Material ist nicht ausreichend
- Zeit zur Entschlüsselung ist größer als...

Beschreiben Sie allgemein ein Kryptogramm!

Eine verschlüsselte Nachricht heißt Kryptogramm oder Chiffretext. Ausgangspunkt ist der

Klartext (engl. plaintext) mit der Bezeichnung M (engl. messages). Diesen soll der Sender mit dem Verschlüsselungsalgorithmus E (engl. encrypt) und dem Schlüssel K (engl. key) chiffrieren. Hieraus bekommt er das Chiffretext, den Geheimtext C (engl. ciphertext). Über einen sicheren Kanal gelangt der Geheimtext zum Empfänger. Seine Aufgabe besteht darin, mit der Entschlüsselungsfunktion D (von decrypt) und dem Schlüssel K den Klartext wieder zu enthüllen. Formelhaft lässt sich der Sachverhalt folgendermaßen darstellen, wobei E und D als Funktionen oder Algorithmen mit einer Reihe von Parametern betrachtet werden können: $C = E(M, K)$ Aus E mit den Parametern M und K ergibt sich der Geheimtext, auch Chiffretext oder Kryptogramm genannt. $M = D(C, K)$



Nennen und erläutern Sie die fünf verschiedenen Arten der Attacken auf ein Kryptosystem!

Brute Force Attack

Man kann Verfahren mit "nackter Gewalt" angreifen, das heißt, man probiert alle möglichen Schlüssel aus. Eine unelegante, aber gefährliche Methode. Da man den brute force attack in den verschiedensten Ansätzen betreiben kann, wird er meist **nicht als eigenständige Gruppe** beschrieben, sondern als eine Möglichkeit innerhalb der folgenden fünf Angriffstypen.

1. Known Ciphertext Attack oder Ciphertext Only Attack (nur der abgefangene Chiffretext steht zur Verfügung)

In diesem einfachen Fall steht dem Angreifer ein entsprechend großes Stück Geheimtext zur Verfügung. Was durchaus realistisch ist, denn wenn ich jemanden abhören kann, steht mir zumindest ein Stück verschlüsselter Text zur Verfügung. Diesen kann man dann beispielsweise nach sichtbarer Entropie untersuchen. Ein Verfahren, das nicht einmal diesem einfachsten Angriff widersteht, ist wertlos.

Gegeben: $C_i = E_k(M_i)$, $i = 1; 2$

Aufgabe: Bestimme P_i , k oder einen Dechiffrier-Algorithmus

2. Known Plaintext Attack (bekannte KlartextChiffretextpaare stehen zur Verfügung)

Der Gegner kennt ein zusammengehöriges Paar Geheimtext/Klartext. Dabei kann es sich bei dem Klartext auch um begründete Vermutungen über den Inhalt handeln. Vor allem dann, wenn sich im Text Standardformulierungen finden lassen, ist diese Methode oft wirkungsvoll. Damit wurde diese eine Nachricht geknackt, nicht notwendigerweise die ganze Übermittlung. Allerdings gibt es Verfahren, die bei erfolgreichen Angriffen dieser Art komplett bloßgestellt werden.

Gegeben: Paare $(M_i; C_i)$, $C_i = E_k(M_i)$

Aufgabe: Bestimme k oder Dechiffrier-Algorithmus

3. Chosen Plaintext Attack (gewählter Klartext)

Steht dem Gegner das Verfahren mit integriertem aktuellem (aber nicht bekannten)

Schlüssel zur Verfügung, kann er selbstgewählte Klartexte damit verschlüsseln und daraus Rückschlüsse auf den verwendeten Schlüssel ziehen. Beispielsweise kann man einen Text voller Nullen oder "A"s verschlüsseln und nach sich wiederholenden Mustern schauen. Auch wenn solch ein Angriff nicht gelingt, kann der Eindringling selbst Meldungen einschleusen. Verfahren, die diesem und den folgenden widerstehen, sind als äußerst sicher einzustufen.

Gegeben: Paare $(M_i; C_i)$ mit gewählten P_i

Aufgabe: Bestimme k oder Dechiffrier-Algorithmus

4. Chosen Cyphertext Attack (gewählter Chiffretext)

Ähnlich wie der *chosen plaintext attack*, nur dass der Angreifer nun eine Auswahl von Geheimtext aussuchen kann und damit dann den entschlüsselten Text finden kann. Dieser Angriff kann bei public key- Systemen eingesetzt werden.

Gegeben: Verschlüsselungsfunktion E_k

Aufgabe: Bestimme Entschlüsselungsfunktion D_k

Die **Methoden** der Kryptoanalyse lassen sich wie folgt gliedern:

1. **Vollständiges Durchsuchen** des Schlüsseltextes und Ausprobieren aller Schlüssel.
2. **Trial and Error.** " Intelligentes" Probieren indem man gewisse Eigenschaften des Kryptosystems ausnutzt ohne jedoch ein systematisches Verfahren anzuwenden.
3. **Statistische Methoden.** Hierbei werden statische Eigenschaften der Klartextsprache ausgenutzt. Z.B. häufig auftretende Zeichenfolgen.
4. **Strukturanalyse.** Konkrete mathematische Analyse des Verfahrens mit dem Ziel, ein maßgeschneidertes „Knack“ Verfahren zu entwickeln.

Gegeben sei eine Cäsar-Chiffre bei beliebigem Modulus n . Ein Angreifer kenne nur einige Kryptogramme c_1, c_2, \dots, c_l . Diskutieren Sie in nachfolgenden Situationen, ob der Angreifer die Kryptogramme entschlüsseln kann:

1. Der Angreifer kennt den Schlüssel k und den Modulus n .

Kennt der Gegner k und n , so kann er leicht mit $D(c, k) \equiv c - k \pmod{n}$ entschlüsseln.

2. Der Angreifer kennt den Schlüssel k , aber nicht den Modulus n .

Kennt der Angreifer nur den Schlüssel $k \in \mathbb{Z}_n$, so kann er zunächst nur hoffen, dass die Kryptogramme alle Ziffern des möglichen Alphabets \mathbb{Z}_n enthalten. Abzählen der verschiedenen Ziffern liefert dann den möglichen Modulus n . Ist dieses n der wirkliche Modulus, so ist die Entschlüsselung wie in 1. möglich. Enthalten c_1, c_2, \dots, c_l jedoch nicht alle Ziffern, etwa weil l zu klein ist, so kann nur geraten werden bzw. es können eventuell Methoden der Sprachstatistik eingesetzt werden.

3. Der Angreifer kennt den Modulus n , aber nicht den Schlüssel k .

Kennt der Angreifer n aber nicht $k \in \mathbb{Z}_n$, so kann er die Texte $c_1 - i, c_2 - i, \dots, c_l - i$ für $i = 0, 1, \dots, n - 1$ durchprobieren, bis er meint, eine oder

mehrere sinnvolle Nachrichten gefunden zu haben, die dann den (möglichen) Schlüssel $k \in \mathbb{Z}_n$ liefern.

4. Der Angreifer kennt weder Schlüssel k noch Modulus n .

Der Gegner kann wie in 2. bzw. 3. vorgehen.

Kennt der Gegner für das Kryptogramm $c_1 \in \mathbb{Z}_n$ den entsprechenden Klartext $m_1 \in \mathbb{Z}_n$, so ändert dieser Umstand hinsichtlich 1. nichts, bzgl. 3. findet er den Schlüssel $k \equiv c_1 - m_1 \pmod{n}$. Für 4. kennt er dann den Schlüssel $k \equiv c_1 - m_1 \pmod{n}$, aber nicht den Modulus, und er kann wie unter 2. fortfahren.

Beschreiben Sie den Eukl. Algorithmus zur Bestimmung des größten gemeinsamen Teilers zweier natürlicher Zahlen. Begründen Sie die Korrektheit dieses Verfahrens und nennen Sie seine Laufzeit in Abhängigkeit von der Eingangsgröße.

Mit dem so genannten euklidischen Algorithmus geht es einfacher. Wenn nämlich zwei Zahlen a und b einen gemeinsamen Teiler besitzen, dann besitzen sowohl die Summe als auch die Differenz dieser Zahlen den gemeinsamen Teiler ebenfalls. Der Euklid'sche Algorithmus berechnet den größten gemeinsamen Teiler zweier positiver ganzer Zahlen.

Beispiel 1:

ggT ($u = 120$, $v = 35$)
120 durch 35 gleich 3 Rest 15
35 durch 15 gleich 2 Rest 5
15 durch 5 gleich 3 Rest 0
====> ggT(120,35) = 5

Beispiel 2:

ggT ($u = 63$, $v = 8$)
63 durch 8 gleich 7 Rest 7
8 durch 7 gleich 1 Rest 1
7 durch 7 gleich 1 Rest 0
====> ggT(63,8) = 1 **====> 63 und 8 sind relativ prim**

Zur Berechnung des $\text{ggT}(a,b)$ ermitteln wir zunächst die größere Zahl der beiden: $\max(a,b)$. Analog hierzu ist $\min(a,b)$ das Minimum. Für den Fall $a=b$ sind wir übrigens bereits fertig! - Ansonsten setzen wir $x := \max(a,b)$ und ziehen nun solange $\min(a,b)$ von x ab, bis $x < \min(a,b)$. Falls nun der Fall eintreten sollte, daß $x=0$, so ist $\min(a,b)$ der gesuchte ggT . Andernfalls setzen wir $a := \min(a,b)$ und $b := x$; und wiederholen den Vorgang. Da die Zahlen immer kleiner, aber nie negativ werden, terminiert das Verfahren. - Zur Beschleunigung kann man statt wiederholter Subtraktion von $\min(a,b)$ auch den gleich den Rest der Division durch $\min(a,b)$ verwenden. Am Ergebnis ändert sich dadurch nichts! Mit dem beschriebenen Verfahren lässt sich der größte gemeinsame Teiler zweier Zahlen also auch ohne vorherige Primzahlzerlegung schnell ermitteln.

Beschreiben Sie den Primzahlentest von Solovay und Strassen und erläutern Sie eine Fehlerwahrscheinlichkeit.

Von R. Solovay und V. Strassen stammt ein stochastischer Algorithmus zur Entscheidung, ob eine vorgegebene Zahl m eine Primzahl ist oder nicht. Falls der Algorithmus das

Resultat "nein" liefert, so ist gesichert, dass m keine Primzahl ist. Liefert der Algorithmus hingegen das Resultat "ja", so weiß man nur, dass diese Antwort mit einer Wahrscheinlichkeit $p \geq 0.5$ korrekt ist. Die Sicherheit der Entscheidung kann man verbessern, indem man den Algorithmus mit demselben Eingangswert öfter aufruft. Bei k -maligem Aufruf verringert sich dann die Wahrscheinlichkeit p_k einer Fehlentscheidung auf 2^{-k} . Jede noch so kleine Irrtumswahrscheinlichkeit ist unterschreitbar:

p_k	1%	0.1%	0.01%	0.001%
$k \geq$	7	10	14	17

Die beim Solovay-Strassen-Algorithmus nicht vermeidbare Unsicherheit bei der Entscheidung "m ist eine Primzahl" wird, speziell bei sehr großen Zahlen (die z.B. bei Verschlüsselungsverfahren auftreten), durch einen deutlich verringerten Rechenaufwand aufgewogen. Während ein nahe liegender deterministischer Algorithmus

$$O(\sqrt{m})$$

Rechenoperationen benötigt (um für

$$1 = 2, 3, \dots, \lfloor \sqrt{m} \rfloor$$

zu probieren, ob m geteilt wird), verringert der stochastische Algorithmus den Aufwand auf $O(k \cdot \log m)$ Rechenschritte.

Th.J.:

Es handelt sich um einen probabilistischen Primzahltest. Untersucht wird eine Zahl p

1. Wähle eine Zufallszahl a kleiner als p .
2. Falls $\text{ggT}(a,p) \neq 1$, besteht p den Test nicht und ist zusammengesetzt.
3. Berechne $j = a^{(p-1)/2} \bmod p$.
4. Berechne das Jacobi-Symbol $J(a,p)$. --> das J-Symbol ist eine Funktion über die Menge der reduzierten Reste der Teiler von p
5. Falls $j \neq J(a,p)$, so ist p definitiv nicht prim.
6. Falls $j = J(a,p)$, so liegt die Wahrscheinlichkeit, dass p nicht prim ist, bei höchstens 50%.

Nehmen Sie an, dass im RSA- System der Modulus n nicht so gewählt wird, dass n die Gestalt $n = p \cdot q$ für verschiedene Primzahlen p, q hat, sondern dass gilt

- (i) $n = p^2$ für eine ungerade Primzahl p oder
- (ii) n ist eine Primzahlpotenz

Wir nehmen an, dass ein Angreifer den öffentlichen Schlüssel (n, e) kennt sowie die Information in (i) bzw. (ii) hat. Diskutieren Sie in beiden Fällen die Sicherheit dieses abgewandelten RSA- Systems bei verschiedenen Attacken.

noch nicht gelöst

Erläutern Sie den Aufbau und die Arbeitsweise des RSA-Kryptosystems und nennen Sie möglichst die Rechenzeiten bei den einzelnen Schritten

RSA Rivest, Shamir, Adleman

1. Wähle Primzahlen! Die Länge soll zwischen 384 und 512 Bit gewählt werden.
2. Berechne $n=pq$ (n hat dann die Länge von 512 bis 1024 Bit)

3. Wähle eine kleine ungerade Zahl E, die zu $\varphi(n)=(p-1)(q-1)$ relativ prim ist, d.h. es gilt $\text{ggT}(E, \varphi(n))=1$ - $\varphi(n)$ Eulersche φ -Funktion zählt die Anzahl aller Zahlen $x < n$ mit $\text{ggT}(x, n)=1$
4. Berechne D mit dem erweiterten euklidischen Algorithmus.
5. E, n ist der öffentliche Schlüssel; D, n ist der geheime Schlüssel

- **Primzahlen:** p, q ; **p = 11 q = 17** **Rechenzeit: $O((\log_q)^2)$, $p \leq q$**
- **öffentlichen Schlüssel: (n, E) und privaten Schlüssel: (n, D) erstellen**
 1. **n** = pq **n = 11 * 17 = 187**
 2. **$\varphi(n)$** = (p-1)(q-1) **$\varphi(n) = 10 * 16 = 160$**
 3. **E** - zufällig gewählt (ungerade und kleiner **$\varphi(n)$**) **E = 23**
 4. nun noch **D** berechnen:
 $ED \bmod \varphi(n) = 1$ oder $ED = \varphi(n) + 1$
- **Bestimmung von D wird mit dem erweiterten Euklidischen Algorithmus**

ges.: D

geg.: E = 23; (p-1)(q-1) = 160

Lösung: **$1 = E^D \bmod (p-1)(q-1)$ $1 = 23^D \bmod 160$**

Auf geht's: t_{i-2} **springe zur vorletzten berechneten Zeile,**
 t_{i-1}, q_{i-1} **springe zur letzten berechneten Zeile**

		$t_i = t_{i-2} - t_{i-1} * q_{i-1}$	
		$t_0 = 0$	
160 = 6 * 23 + 22	$q_0 = 6$	$t_1 = 1$	
23 = 1 * 22 + 1	$q_1 = 1$	$t_2 = 0 - 1 * 6$	$t_2 = -$
6			
22 = 22 * 1 + 0 Ende, da 0 erreicht!		$t_3 = 1 - (-6 * 1)$	<u>$t_3 = 7$</u>

Hurra, D ist gefunden! D = 7

Was ist , wenn t_i ein negatives Ergebnis enthält? Dann **$D = t_i \bmod \varphi(n)$**

Verschlüsselung:

m = Nachricht

Blöcke

$c = m^E \bmod n$

$c = \{c + c + \dots + c\}$

Entschlüsselung:

$m = c^D \bmod n$

m = 6882

m = 68; m = 82;

$c = 68^{23} \bmod 187 = 85$; $c = 82^{23} \bmod 187 = 125$;

c = 85 125

$m = 85^7 \bmod 187 = 68$, $m = 125^7 \bmod 187 = 82$

1. Aufwand hängt vom gewählten Test ab, z.B. Solvay-Strassen
2. Aufwand minimal
3. 4. Aufwand minimal

Beschreiben Sie den Aufbau und die Arbeitsweise von Playfair-Chiffren

5 x 5 – Matrix

Alphabet = {A,...,Z} – {J}, I = J

Schlüsselwort und Restalphabetbuchstaben oder Schlüssel aus 25 Buchstaben

Doppelte Buchstaben im Klartext werden durch einen Füllbuchstaben (der nicht doppelt vorkommt) getrennt

- E: Klartext (P) => buchstabenweise gruppieren (Zweiergruppen)
Buchstaben in der Schlüsselmatrix nach Spalten-, Zeilen- und Rechteckregel zuordnen.
- D: Chiffretext (C) => buchstabenweise gruppieren (Zweiergruppen)
Buchstaben analog entgegengesetzt (zu E) in der Schlüsselmatrix nach Spalten-, Zeilen- und Rechteckregel zuordnen.

G	R	U	E	N
S	P	A	B	C
D	F	H	I/J	K
L	M	O	Q	T
V	W	X	Y	Z

Wie funktioniert der Diffie Hellman Schlüsselaustausch?

Hier können sich zwei Partner, die zuvor noch keine Daten ausgetauscht haben, in aller Öffentlichkeit auf einen geheimen Schlüssel für einen symmetrischen Algorithmus einigen, ohne das Dritte ihn erfahren. Genauso wie beim RSA basiert alles auf der Rechnung "c = m^k MOD n", und es müssen auch hier sehr große Zahlen (1024Bit oder mehr) gewählt werden.

Partner A	Öffentlichkeit	Partner B
	p (Primezahl) s (\mathbf{N} und $s < p$)	
a (\mathbf{N} und $a < p$) $\mathfrak{A} = s^a \text{ mod } p$	$\mathfrak{A}, \mathfrak{Q}$	b (\mathbf{N} und $b < p$) $\mathfrak{Q} = s^b \text{ mod } p$
k = $\mathfrak{Q}^a \text{ mod } p$		k = $\mathfrak{A}^b \text{ mod } p$

Abbildung 6: Ablauf des Diffie Hellman Schlüsselaustauschs

Für den Schlüsselaustausch (Abbildung 6) einigen sich beide Partner in aller Öffentlichkeit auf eine Primzahl (p) und eine natürliche Zahl (s) die kleiner als p ist. Jetzt sucht jeder Partner für sich je eine natürliche Zahl (a, b) aus, die auch kleiner als p sein muss, und jeder errechnet sich damit eine Zahl ($\mathfrak{A}, \mathfrak{Q}$), die sich die Partner in aller Öffentlichkeit austauschen. Die Zahlen a und b bleiben geheim. Nun kann sich jeder Partner die Zahl k errechnen. Ein Dritter, der die Zahlen p, s, \mathfrak{A} und \mathfrak{Q} mitgehört hat, kann nicht die Zahl k berechnen, weil er weder die Zahl a noch die Zahl b kennt. Die beiden Partner können aber die Zahl k nun als Basis für einen symmetrischen Algorithmus nutzen. Die große Gefahr bei diesem Schlüsselaustausch ist, dass sich ein Dritter die Leitung auftrennt und dann nach beiden Partnern hin getrennt je einen Schlüsselaustausch vornimmt. Beide Partner denken, sie hätten ein gemeinsamen geheimen Schlüssel k, aber dabei haben sie jeweils mit dem Angreifer den Schlüsselaustausch vorgenommen und sind auch im Besitz verschiedener Schlüssel k_1 und k_2 . Dieses Verfahren erfordert es unbedingt, mit seinem Partner die Checksumme des Schlüssels k zu vergleichen, und zwar so, dass es der Angreifer nicht verfälschen kann.

Betrachten Sie ein RSA-Kryptosystem mit den Parametern

n = p · q und e; d. Eine Nachricht $m \in \{0; 1; \dots; n-1\}$ heißt Fixpunkt, falls $m^e \equiv m \text{ mod } n$ gilt.

a. Zeigen Sie, dass mit jedem $m \neq 0$ auch $n-m$ ein Fixpunkt ist.

b. Für welche Verschlüsselungsexponenten e gilt, dass jede Nachricht $m \in \{0; 1; \dots; n-1\}$ ein Fixpunkt ist?

a) Ausgangslage: $m^e \equiv m \pmod n$

Lösung(?) $(n-m)^e \equiv (n-m) \pmod n$

$$\begin{aligned} &\equiv (-m)^e \pmod n \\ &\equiv (-1)^e \cdot (-m)^e \pmod n \\ &\equiv (-m)^e \pmod n \\ &\equiv (-1)^e \cdot (-m)^e \pmod n \\ &\equiv -m \pmod n \\ &\equiv (n-m) \pmod n \end{aligned}$$

b) $e=1$

Was sind die Vor- Nachteile von asymmetrischer Verschlüsselung im Vergleich mit symmetrischer Verschlüsselung?

Der primäre Vorteil von asymmetrischer Kryptographie ist die erhöhte Sicherheit und Bequemlichkeit: Private Schlüssel müssen nie übertragen oder jemanden gezeigt werden. Im Gegensatz dazu müssen bei symmetrischer Kryptographie die geheimen Schlüssel (entweder persönlich oder durch einen Kommunikationskanal) übertragen werden, so dass ein Feind die Chance hat, die geheimen Schlüssel bei der Übertragung abzufangen. Ein weiterer wichtiger Vorteil von asymmetrischer Kryptographie ist die Möglichkeit der elektronischen Unterschrift. Eine Authentifikation mit symmetrischer Kryptographie erfordert das Teilen eines Geheimnisses und manchmal auch das Vertrauen in eine dritte Instanz. In Konsequenz kann ein Sender ein zuvor authentifizierte Nachricht abstreiten, in dem er behauptet, das das geteilte Geheimnis durch einen der Beteiligten kompromittiert wurde. Asymmetrische Authentifizierung andererseits verhindert diese Leugnungsvariante, jeder Nutzer hat die alleinige Verantwortlichkeit für den Schutz seines privaten Schlüssels. Deshalb wird diese Eigenschaft der asymmetrischen Authentifizierung auch *Unleugbarkeit* genannt.

Ein Nachteil der asymmetrischen Verschlüsselung ist die Geschwindigkeit: Es gibt beliebte symmetrische Verschlüsselungsverfahren, die deutlich schneller als jedes momentan verfügbare asymmetrische Verschlüsselungsverfahren sind. Trotzdem kann asymmetrische Verschlüsselung mit symmetrischer Verschlüsselung kombiniert werden, um das Beste aus beiden Welten zu nutzen: Die Sicherheitsvorteile der asymmetrischen und die Geschwindigkeitsvorteile der symmetrischen Verschlüsselung. Die asymmetrische Verschlüsselung wird dazu eingesetzt, den Schlüssel der symmetrischen Verschlüsselung zu chiffrieren, mit dem eine große Menge von Dateien oder Nachrichten chiffriert wird. Diese Hybridtechnik wird *digitaler Umschlag* genannt (Für RSA Verfahren).

Wie schnell ist RSA?

Eine "RSA Operation" zum verschlüsseln, entschlüsseln, unterschreiben oder überprüfen ist im Kern nur eine modulare Potenzierung, die durch eine Folge von modularen Multiplikationen dargestellt werden kann.

Für praktische Anwendungen wird ein kleiner öffentlicher Exponent gewählt. Tatsächlich können alle Nutzer den gleichen öffentlichen Exponenten haben, jeder mit einem anderen Modul. (Es gibt nur einige Einschränkungen an die Auswahl der Primfaktoren des Moduls, wenn ein fester öffentlicher Exponent benutzt wird.) Ein kleiner öffentlicher Exponent macht die Verschlüsselung schneller als die Entschlüsselung und die Überprüfung einer Unterschrift schneller als das Unterschreiben. Mit den üblichen Algorithmen zur modularen

Potenzierung benötigen Operationen mit dem öffentlichen Schlüssel $O(k^2)$ Schritte, Operationen mit dem privaten Schlüssel $O(k^3)$ Schritte und die Schlüsselerzeugung $O(k^4)$ Schritte, wobei k die Größe des Moduls in Bit ist. (Die $O()$ Notation gibt eine obere Schranke für das asymptotische Verhalten eines Algorithmus an. "Schnelle Multiplikationstechniken", wie FFT basierende Algorithmen benötigen asymptotisch weniger Schritte, jedoch werden sie seltener implementiert, da sie deutlich komplexer und bei kleinen Schlüsselgrößen langsamer als die üblichen Verfahren sind.

Wie groß sollten die RSA-Module (Schlüssel) sein?

Die beste Größe eines RSA Moduls hängt von den eigenen Sicherheitsanforderungen ab. Je größer der Modul, desto sicherer und langsamer ist das Verfahren. Die Länge des Moduls sollte zuerst von der Überlegung abhängen, wie viel die eigenen zu schützenden Daten wirklich Wert sind und dann von der Überlegung, wie mächtig die potentiellen Feinde sind.

Eine spezielle Untersuchung der Sicherheit eines 512 Bit Schlüssels zeigt, dass dieser mit weniger als 1Mio. US\$ Aufwand in 8 Monaten 1997 zu faktorisieren ist. Es wird angenommen, dass 512 Bit Schlüssel nicht länger hinreichende Sicherheit im Zeitalter der neuen Faktorisierungsmethoden und der verteilten Rechentechnik leisten können. Solche Schlüssel sollte nicht nach 1997 oder 1998 eingesetzt werden. Die RSA Laboratorien empfehlen 768 Bit für den privaten Gebrauch, 1024 Bit für den geschäftlichen Gebrauch und 2048 Bit für wirklich wichtige Schlüssel wie die einer Zertifikationsinstanz. Ein 768-bit Schlüssel sollte bis ins Jahr 2004 benutzbar sein.

Was ist das ElGamal Kryptosystem?

Das ElGamal System ist ein asymmetrisches Kryptosystem, das auf dem diskreten Logarithmus Problem ($a^x \bmod p$) basiert. Es enthält sowohl Verschlüsselungs- als auch Unterschriftsalgorithmen. Der Verschlüsselungsalgorithmus prinzipiell mit dem Diffie-Hellman Schlüsseltausch identisch. Das System wird durch eine Primzahl p und einer Ganzzahl g , deren Potenzen modulo p eine große Anzahl von Elementen umfasst (wie bei Diffie-Hellman), parametrisiert.

Schlüsselgenerierung

- wähle global:
 - p , so das DLL schwierig öffentlich
 - a primitive Wurzel von p öffentlich
- $p = 3137$
- $a = 577$

- jeder Tln. wählt:
 - geheimen Schlüssel k_i ($k_i < p-1$) geheim
 - k_i relativ prim zu $p-1$, d.h. $\text{ggT}(k_i, p-1)=1$ (nur Signatursystem nach El Gamal)
 - berechnet $-k_i \bmod (p-1)$ geheim
 - $\gamma_i = a^{-k_i} \bmod p$ (*) öffentlich

Teilnehmer B:

- wählt $k_B = 1762$ geheim (privater Schlüssel)
- $-k_B \bmod (p-1) = -1762 \bmod 3136 = -1762 + 3136 = 1374$ geheim
- $\gamma^B = a^{-k_B} \bmod p = 577^{1374} \bmod 3137 = 858$
- öffentlicher Schlüssel ($p=3137, a = 577, \gamma^B = 858$)

Verschlüsselung

- A will Nachricht m ($m < p$) an B schicken

- A besorgt sich p, a, γ_B
- A wählt Zufallszahl z ($z < p$)
- A berechnet $c = \gamma_B^z \cdot m \bmod p$
- **A sendet an B: $a^z \bmod p, c$**

- **A will B vertraulich die Nachricht $m = 2115$ schicken.**
- **A besorgt sich öffentlichen Schlüssel ($p=3137, a = 577, \gamma^B = 858$) von B**
- **A wählt $z = 932$ geheim**
- **berechnet $a^z \bmod p = 577^{932} \bmod 3137 = 1852$**
- **berechnet $\gamma_B^z \bmod p = 858^{932} \bmod 3137 = 749$**
- **berechnet $c = \gamma_B^z \cdot m \bmod p = 749 \cdot 2115 \bmod 3137 = 3087$**
- **schickt $a^z = 1852$ und $c = 3087$ an B**

Entschlüsselung

- B berechnet $m^* = (a^z)^{k_B} \cdot c \bmod p$

B berechnet $(a^z)^{k_B} \cdot c \bmod p = 1852^{1762} \cdot 3087 \bmod 3137 = 2115$

Eine einfachere und kürzere Variante

Schlüsselgenerierung:

- p (**23**), so das DLL schwierig
- g (**7**) primitive Wurzel von p
- Wahl des geheimen Schlüssels (Exponenten) a (**6**) ($a < p-1$) und a ist relativ prim zu $p-1$, d.h. $\text{ggT}(a, p-1)=1$ (s. Eukl. Algo.)
- Berechnung $A = g^a \bmod p$ ($A=4$)
- öffentlicher Schlüssel $(p, g, A) = (23, 7, 4)$

Verschlüsselung (E):

- B besorgt sich den öffentlichen Schlüssel von A
- B wählt eine Zufallszahl b (**3**) $b \in \{1, \dots, p-2\}$
- und berechnet $B = g^b \bmod p$ ($B=21$) und verschlüsselt Nachricht ($m = 15$) mit
- $c = A^b m \bmod p$ ($c=17$)
- Schlüsseltext (B, c) (**21, 17**)

Entschlüsseln (D):

- A hat (B, c) von B erhalten
- A bestimmt den Exponenten $x = p-1-a$ und berechnet
- $m = B^x c \bmod p$ (**$21^{(23-1-6)} \cdot 17 \bmod 23 = 15$**)