

Kryptografie: Fragen von Klausuren 2016 und 2015

Natürlich alle angaben ohne gewähr :-)

1. Was ist multiple key public key cryptography und wie funktioniert das? Für welche Anwendungszenarien ist das von Vorteil (und welche Konkreten Nachteile klassischer Ansätze werden dabei vermieden) ? (2 Punkte)

Das verallgemeinerte Konzept der Public Key Kryptographie wird als Multiple Key Public Key bezeichnet.

Alice	K_A
Bob	K_B
Carol	K_C
Dave	$K_A \& K_B$
Ellen	$K_B \& K_C$
Frank	$K_C \& K_A$

Das könnte etwa so ausschauen:

Alice verschlüsselt mit K_A

⇒ Ellen oder Bob & Carol gemeinsam können entschlüsseln.

Dave K_A wie Alice

K_B wie Bob

oder so daß nur Carol entschlüsseln kann.

Encryption Decryption

K_A	$K_B \& K_C$
K_B	$K_A \& K_C$
K_C	$K_A \& K_B$
$K_A \& K_B$	K_C
$K_A \& K_C$	K_B
$K_B \& K_C$	K_A

→ Ausweitbar auf n (beliebig viele) Keys: ist eine Teilmenge von Schlüsseln verwendet worden um eine Nachricht zu verschlüsseln, wird der Rest der Schlüssel benötigt um zu entschlüsseln.

Anwendung: Eine Gruppe von Leuten und man will mit vorher nicht bekannten Teilgruppen geheim kommunizieren. Dies bräuchte:

- Einen Schlüssel für jede mögliche Kombination
→ viele Schlüssel.
- Jede Nachricht muß für jede Kombination extra verschlüsselt werden
→ viele Nachrichten.

Mit der Multiple Key Public Key Kryptographie ist die Handhabung einfacher.

2. Erklären sie das Konzept von randomisierter Verschlüsselung und wie dies bei Signaturen mit El Gamal angewendet wird. Was ist der Vorteil von RSA gegenüber dem Einsatz von El Gamal bei digitalen Signaturen ? (2Punkte)

Die Sicherheit dieses Verfahrens besteht in der Schwierigkeit diskrete Logarithmen in endlichen Körpern zu berechnen. Dieses Verfahren wurde ursprünglich für digitale Unterschriften verwendet (diese Variante wird zuerst gezeigt).

Primzahl p sowie Primitivwurzel $q \in \mathbf{N}$, $x \in \mathbf{N}$, q, x beliebig und $< p$.

$$y \equiv q^x \pmod{p}$$

y, q, p sind der Public Key, wobei q und p von mehreren AnwenderInnen verwendet werden können. x ist der Private Key.

Wähle K mit $(K, p - 1) = 1$ und berechne $a = q^K \pmod{p}$.

$b \equiv (M - xa)K^{-1} \pmod{p - 1}$ wird berechnet.

Die Unterschrift ist das Paar a und b . K muß geheim bleiben.

Um die Unterschrift zu verifizieren, muß folgendes berechnet werden:

$$\begin{aligned} y^a a^b \pmod{p} &= q^{q^K} a^b \\ \text{wobei } y^a a^b &= y^{q^K} q^{Kb} \\ &= q^{xq^K} q^{Kb} \\ &= q^{xa} q^{Kb} \end{aligned}$$

Jede Unterschrift benötigt ein neues K und muß zufällig gewählt werden. Die sichere Variante ersetzt M durch $H(M)$.

Für die Variante mit Verschlüsselung muß wieder ein K mit $(K, p - 1) = 1$ gewählt werden. $a = q^K \pmod{p}$

$$b = y^K M \pmod{p}$$

a und b sind der Ciphertext (mit doppelter Länge).

Für die Entschlüsselung muß

$$M = \frac{b}{a^x} \pmod{p}$$

berechnet werden.

$$\begin{aligned} a^x &\equiv q^{Kx} \pmod{p} \\ \frac{b}{a^x} &\equiv \frac{y^K M}{q^{Kx}} \\ &\equiv \frac{q^{xK} M}{q^{Kx}} \pmod{p} \\ &\equiv M \pmod{p} \end{aligned}$$

Bemerkung 1: durch die Wahl von K spricht man von randomisierter Verschlüsselung.

Bemerkung 2: y^K wird durch Diffie-Hellman key-exchange generiert und mit Message multipliziert (siehe dort!).

Vorteil von RSA:

Bei RSA wird keine Zufallszahl zum Signieren benötigt. Manche Systeme haben Probleme gute Zufallszahlen zu generieren. Besonders kritisch wird es, wenn eine Zufallszahl erraten wird oder zwei mal die gleiche Zufallszahl verwendet wird.

3. Beschreiben sie detailliert **eine** der drei besprochenen Szenarien die darlegen, warum RSA nie benutzt werden darf um ein „zufällig“ wirkendes Dokument zu signieren (chosen ciphertext Attacke). Erklären sie weiteres, was die Anwendung einer Hashfunktion bei der Signierung ändert (und ob das überhaupt was ändert).

Szene 1: Eve bekommt c von Alice, das mit RSA und ihrem Public Key verschlüsselt wurde. Eve will $m = c^d$ berechnen. Sie wählt $r < n$ zufällig und holt e . Dann

$$\begin{aligned} x &= r^e \mod n \\ \text{berechnet sie } y &= xc \mod n \quad x = r^e \mod n \Rightarrow r = x^d \mod n \\ t &= r^{-1} \mod n \end{aligned}$$

Eve bringt Alice dazu, y mit ihrem Private Key zu unterschreiben und dabei y zu entschlüsseln. (Muß die Nachricht, nicht den Hash unterschreiben.) Alice schickt Eve dann $u = y^d \mod n$

Eve rechnet

$$tu \mod n = r^{-1}y^d \mod n = r^{-1}x^d c^d \mod n = c^d \mod n = m, \text{ weil } x^d = r \text{ ist.}$$

Szene 2: Trent hat die Aufgabe, Nachrichten mit RSA zu signieren. Mallory hat ein m' , die er von Trent unterschrieben haben möchte, was Trent normalerweise nicht täte.

Mallory wählt ein x beliebig und berechnet $y = x^e \mod n$, $m = ym' \mod n$ und schickt m zu Trent. Trent retourniert $m^d \mod n$.

Mallory berechnet

$$\begin{aligned} &(m^d \mod n)x^{-1} \mod n \\ &= ((x^e m')^d \mod n)x^{-1} \mod n \\ &= x^{ed}x^{-1} \mod n \cdot m'^d \mod n = m'^d \mod n \end{aligned}$$

Szene 3: Eve möchte, daß Alice m_3 unterschreibt. Sie generiert m_1 und m_2 , sodaß $m_3 \equiv m_1 m_2 \pmod{n}$.

Wenn Alice m_1 und m_2 unterschreibt, kann Eve m_3 berechnen:
 $m_3^d = (m_1^d \mod n)(m_2^d \mod n)$.

Diese Szenen zeigen, daß RSA nie benutzt werden darf, um von jemandem Fremden ein zufälliges Dokument zu unterzeichnen. Es muß zuvor immer eine One-Way Hash Funktion benutzt werden!

Falls man eine Hash-Funktion benutzt:

Wenn man vorher eine One-Way Hash Funktion anwendet, dann ???

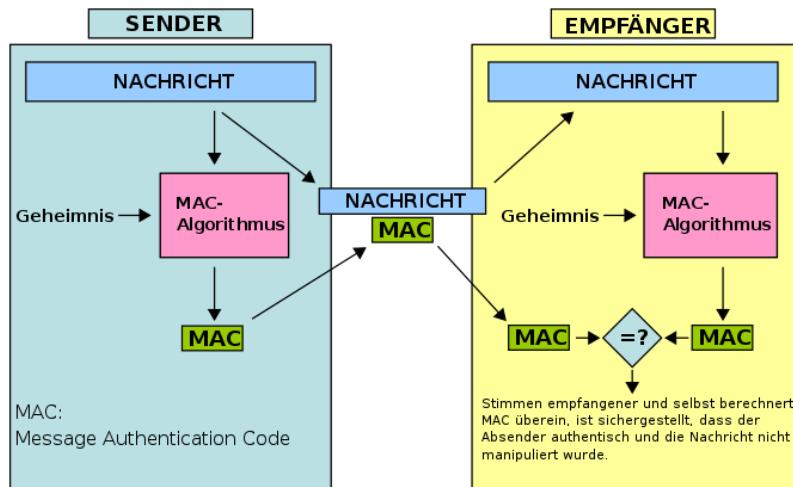
4. Bleibt bei Substitution Ciphers oder Transposition Ciphers das Histogramm des Ciphertextes unverändert verglichen mit dem Histogramm des Plaintext? Begründen sie ihre Antwort! Welche Histogramm-basierte Attacke gibt es gegen Verschlüsselung (von Texten) mit Substitution Ciphers ? (2 Punkte)

Simple Substitution Ciphers: Ein Char im Plaintext wird durch ein entsprechendes Char im Ciphertext ersetzt (Caesar Cipher). Die klassische Methode zur Kryptoanalyse mit einer Ciphertext-only Attacke verwendet eine Häufigkeitsanalyse der verwendeten Symbole, z.B. sind die Auftrittswahrscheinlichkeiten von einzelnen Buchstaben oder Buchstaben Tupeln und Tripeln für alle Sprachen bekannt. Durch einen Abgleich der Häufigkeitshistogramme zwischen Plain- und Ciphertext wird die Substitution erkannt.

Transposition Ciphers: Die Zeichen des Plaintextes bleiben gleich. Es wird nur die Anordnung geändert, z.B. die Zeilen als Spalten angeordnet. Da die Verteilung der Buchstaben gleich bleibt, wie in der normalen Sprache, bietet diese einen verwertbaren Hinweis auf die Verschiebung. Transposition (oder Permutation) kann einer known-Plaintext Attacke nicht widerstehen.

5. Was ist ein Message Authentication Code (MAC)? Gibt es Unterschiede in der Funktionalität zur digitalen Signatur, wenn ja, inwiefern? Beschreiben sie jeweilige Vor- und Nachteile? Warum wird bei IPSec zur Authentifizierung ein MAC und keine digitale Signatur Verwendet ? (2 Punkte)

Ein Message Authentication Code (MAC; deutsch Nachrichtenauthentifizierungscode) dient dazu, Gewissheit über den Ursprung von Daten oder Nachrichten zu erhalten und ihre Integrität zu überprüfen. MAC-Algorithmen erfordern zwei Eingabeparameter, erstens die zu schützenden Daten und zweitens einen geheimen Schlüssel, und berechnen aus beidem eine Prüfsumme, den Message Authentication Code.



Kryptographische Hashfunktionen können zur Berechnung von MACs genutzt werden, jedoch gehen MACs über die Verwendung einfacher Hashwerte hinaus: Wenn eine Nachricht nur mit ihrem Hashwert als MAC übertragen würde, wäre dies nicht sicher, da die Hashfunktion öffentlich bekannt ist. Ein Angreifer könnte die Nachricht modifizieren und einen neuen Hashwert für die neue Nachricht berechnen und diesen übertragen. Falls allerdings Sender und Empfänger ein Geheimnis teilen, kann dies in die Hashberechnung einfließen, so dass ein Angreifer nach Modifikation einer Nachricht nicht in der Lage ist, den passenden Hashwert zu berechnen.

Auch wenn die Manipulationssicherheit von MACs eine Verwandtschaft mit digitalen Signaturen nahelegt, bestehen Unterschiede der beiden Verfahren. MACs unterscheiden sich von digitalen Signaturen darin, dass die Überprüfung des MACs Kenntnis desselben geheimen Schlüssels erfordert, der zu seiner Berechnung genutzt wurde. Daher kann jeder, der einen MAC überprüfen kann, diesen auch berechnen; entsprechend ist er nicht in der Lage, gegenüber Dritten zu beweisen, von wem die Nachricht stammt. Im Gegensatz dazu werden digitale Signaturen mit Hilfe eines nur dem Absender bekannten Schlüssels erstellt und mit Hilfe eines öffentlichen Schlüssels überprüft. So ist sichergestellt, dass der Unterschreiber im Besitz des privaten Schlüssels ist, und es sich so mit hoher Wahrscheinlichkeit um den vorgegebenen Autor der Nachricht handelt.

Beschreiben sie jeweilige Vor- und Nachteile! Warum wird bei IPSec zur Authentifizierung ein MAC und keine digitale Signatur verwendet?

MAC ist viel schneller als eine digitale Signatur (mittels Public Key Verfahren), benötigt aber ein geteiltes Geheimnis, und der Empfänger ist nicht in der Lage, gegenüber Dritten zu beweisen, von wem die Nachricht stammt. Geschwindigkeit ist bei IPSec natürlich ein wichtiger Faktor, und wenn man schon ein Geheimnis nur mit dem Empfänger teilt, sollte auch die Quelle einer Nachricht gesichert sein. Die Vor- und Nachteile sind hier ähnlich wie bei hybriden Verfahren, wenn dies hier nicht sogar genau die gleiche Lage ist, was Geschwindigkeit/Sicherheit angeht.

6. Zum Verständnis von RSA: gegeben sind als public key $n = 15$ und $e = 7$. Sie fangen einen Ciphertext $c = 8$ ab. Ermitteln Sie durch eine Faktorisierungs Attacke den dazugehörigen Plaintext und erklären Sie die Rolle der Faktorisierung bei ihrer Attacke. (2 Punkte)

Unser Ziel ist es den Ciphertext $c = 8$ zu entschlüsseln.

Verschlüsselung: $c = m^e \pmod{n}$.

Entschlüsselung: $m = c^d \pmod{n}$.

Das RSA-Verfahren definiert $n = pq$, wobei p, q Primzahlen sind, und d ist das Inverse von e in $Z_{(p-1)(q-1)} = Z_{\phi(n)}$ da $\phi(n) = \phi(pq) = (p-1)(q-1)$. Wir suchen also ein d sodass $de \equiv 1 \pmod{\phi(n)}$.

Wir brauchen also p und q . Dazu müssen wir n faktorisieren. Da hier n sehr klein ist, erkennen wir durch bloßes Hinschauen, dass $n = 3 \cdot 5$ also $p = 3$ und $q = 5$. Nun können wir $\phi(n) = (p-1)(q-1)$ berechnen.

$$\phi(15) = (3-1)(5-1) = 2 \cdot 4 = 8.$$

Das Inverse d kann mit dem erweiterten euklidischen Algorithmus berechnet werden. Da hier $\phi(n)$ und e sehr klein sind, erkennen wir durch bloßes Hinschaun, dass $d = 7$.

Nichtsdestotrotz, so wird d berechnet:

Wir ermitteln nun das Inverse von $7 \pmod{8}$, also ein d sodass $d \cdot 7 \equiv 1 \pmod{8}$.

Gesucht sind also x und $k \in \mathbb{Z}$, sodass $7 \cdot x = 8 \cdot k + 1$.

Dazu berechnen wir zunächst mit dem Euklidischen Algorithmus den $\gcd(8, 7)$ (einfacher: wir beantworten die Frage "7 mal was hat einen Rest von 1 bei der Division durch 8?"):

$$8 = 1 \cdot 7 + 1$$

$$7 = 7 \cdot 1 + 0.$$

Da der $\gcd(8, 7) = 1$, existiert garantiert ein Inverses von $7 \pmod{8}$. Wir berechnen nun $x, y \in \mathbb{Z}$ sodass $1 = x \cdot 8 + y \cdot 7$:

$$1 = 8 - 1 \cdot 7 = 1 \cdot 8 - 1 \cdot 7 = 1 \cdot 8 + (-1) \cdot 7.$$

umgeformt (für $7 \cdot x = 8 \cdot k + 1$ bzw $8 \cdot k + 1 = 7 \cdot x$):

$$-1 \cdot 8 + 1 = -1 \cdot 7 \text{ also für } d \cdot 7 \equiv 1 \pmod{8}: -1 \cdot 7 = -7 \equiv 1 \pmod{8}.$$

Da $-1 \pmod{8}$ negativ ist, und es sich mit positiven Zahlen schöner rechnet, rechnen wir $-1 \pmod{8}$ in die positive Restklasse um, und das wäre 7. Das Inverse d von $7 \pmod{8}$ ist also 7. (Wir könnten aber auch mit -1 statt 7 weiterrechnen)

Nun können wir $m = c^d \pmod{n}$ berechnen:

$$m = 8^7 \pmod{15} = (8^2)^3 \cdot 8 \pmod{15} = (64 \pmod{15})^3 \cdot 8 \pmod{15} = (64 \pmod{15})^3 \cdot$$

$$8 \pmod{15} = 43 \cdot 8 \pmod{15} = (43 \pmod{15}) \cdot 8 \pmod{15} = (43 \pmod{15}) \cdot$$

$$8 \pmod{15} = 4 \cdot 8 \pmod{15} = 32 \pmod{15} = 2.$$

Die ursprüngliche Nachricht m ist also 2.

Erklären Sie die Rolle der Faktorisierung bei Ihrer Attacke.

Um $\phi(n)$ und d berechnen zu können, brauchen wir p und q . Dazu müssen wir n faktorisieren. Darin liegt die Sicherheit des RSA-Verfahrens: Die Faktorisierung von n bei sehr großen p und q ist ein "computational unfeasible problem", d.h. es gibt bis jetzt keinen effizienten Algorithmus für diese Faktorisierung.

7. Welches Problem wird bei SET durch sog. duale Signaturen gelöst und wie funktioniert das (z.B. bei der Bank wenn die Zahlungsinformation entschlüsselt wird) ? (2 Punkte)

Ein wichtiges Detail von SET ist die sogenannte „duale Signatur“ Zwei Nachrichten werden verbunden die für verschiedene Empfänger bestimmt sind. In diesem Fall geht es um die Bestellungsinformation die an den Händler geht und die Bezahlungsinformation die an die Bank geht sie müssen verbunden sein um bei Streifällen die Aktion rekonstruieren zu können (diese Bezahlung ist für diese Ware und nicht für eine andere), die Bank und der Händler sollen aber andererseits die jeweils anderen Informationen nicht lesen können.

Das wird wie folgt realisiert: der Kunde berechnet einen Hash der OI und PI, fügt diese zusammen und berechnet vom Ergebnis wieder einen Hash. Das Ergebnis wird mit dem privaten Signatur-Key signiert. Hat der Händler nun diese duale Signatur DS, die OI, und den Hash der PI kann er $H(PIMD||H(OI))$ und $D_{KU_C}(DS)$ berechnen. Sind diese Ausdrücke gleich, hat der Handler die Unterschrift verifiziert. Die Bank kann das analog machen mit vertauschten Rollen von PI und OI.

Ablauf

1. Kunde eröffnet ein kreditkartenfähiges Konto bei einer SET unterstützenden Bank
2. Kunde erhält ein digitales Zertifikat (bestätigt den public key und das Ablaufdatum)
3. Händler haben auch Zertifikate (X.509v3 digitale Zertifikate) - für jede Karte die er akzeptiert für zwei Schlüssel: zum Signieren und zum Key Exchange.
4. Der Kunde tätigt eine Bestellung
5. Zusätzlich zum Bestellformular schickt der Händler eine Kopie seiner Zertifikats, daß er ein echter Händler ist.
6. Der Kunde schickt die Bestellung und Zahlungsinformation an den Händler zusammen mit seinem Zertifikat.
7. Der Händler verlangt Zahlungsauftragserstellung (das Payment Gateway muß bestätigen, daß die Kreditkarte o.k. ist)
8. Der Händler bestätigt die Bestellung
9. Die Ware oder Dienstleistung wird vom Händler bereitgestellt
10. Der Händler verlangt Bezahlung (vom payment Gateway).

8. Erklären Sie warum bei OTP Verschlüsselung ein OTP nur 1x als keystream verwendet werden darf. Welche Attacke wird ermöglicht wenn noch zusätzlich einer von zwei involvierten Plaintexten (die mit dem gleichen OTP verschlüsselt wurden) zur Verfügung steht ? (1 Punkt)

Sei K der OTP bzw. der Key für die OTP Verschlüsselung, mit dem der Plaintexte P_1 und P_2 verschlüsselt werden.

Seien gegeben zwei Ciphertexte $C_1 = P_1 \oplus K$ und $C_2 = P_2 \oplus K$. Eve hat C_1 und C_2 abgefangen und rechnet $C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K$. Durch die Kommutativität der XOR-Operation ergibt sich $P_1 \oplus P_2 \oplus K \oplus K$ und wegen $a \oplus a = 0$ bleibt $P_1 \oplus P_2$. Daraus kann Eve wieder durch Methoden der Häufigkeitsanalyse P_1 und P_2 isolieren.

Ist zusätzlich ein dazugehöriger Plaintext verfügbar (z.B. P_2 , also eine known-Plaintext Attacke), kann durch Berechnung von $C_1 \oplus C_2 \oplus P_2 = P_1 \oplus P_2 \oplus P_2$ der andere Plaintext P_1 direkt berechnet werden.

9. Beschreiben Sie ein Verfahren und eine Anwendung von Secret Splitting für 2 und mehr Personen. Was ist der Unterschied zu Secret Sharing ? (1 Punkt)

Secret Splitting bedeutet, daß eine Nachricht in mehrere Teile aufgeteilt wird. Jede einzelne Teil ist für sich alleine bedeutungslos. Alle gemeinsam ergeben erst die Nachricht.

Das Verfahren ist einfach. Trent splittet M zwischen Alice und Bob:

1. Trent generiert einen zufälligen String R gleich lang wie M
2. Trent wendet ein XOR auf M und R an, $\rightarrow S$:

$$M \oplus R = S$$

3. Trent schickt R an Alice und S an Bob.

Das ganze Verfahren ist eigentlich eine Verschlüsselung mit einem One-Time Pad , wo Alice das Pad besitzt und Bob den Ciphertext.

1. Trent generiert drei Strings R, S, T .

2. Berechnen:

$$M \oplus R \oplus S \oplus T = U$$

3. ...

Unterschied zu Secret Sharing:

(m, n)-threshold Verfahren: ein einfaches Beispiel solcher Verfahren: Die Nachricht wird in m Stücke (shadows) geteilt, sodaß mit n Teilen die Nachricht rekonstruiert werden kann.

10. Erklären Sie anhand eines einfachen Protokolls den Diffie-Hellman Key-exchange Algorithmus. Wählen Sie $n = 5$ und rechnen Sie ein konkretes Beispiel durch. Was ist eine Primitivwurzel und warum ist deren Verwendung hier wichtig ? (2 Punkte)

Benötigt werden n primzahlen und g . Wobei g eine Primitivwurzel modulo n ist.

1. Alice wählt eine zufällige große Zahl x und schickt Bob $X = g^x \text{ mod } n$.
2. Bob wählt eine zufällige große Zahl y und schickt Alice $Y = g^y \text{ mod } n$.
3. Alice berechnet $K = Y^x \text{ mod } n$.
4. Bob berechnet $K' = X^y \text{ mod } n$.

$K = K' = g^{xy} \text{ mod } n$. Niemand kann das berechnen, weil nur n, g, X, Y über den Kanal geschickt wurden. Es müßte ein diskreter Logarithmus berechnet werden, um x oder y zu erhalten. K ist also der geheime Schlüssel, den beide unabhängig berechnet haben.

Wichtig ist, dass n sehr groß gewählt wird und auch $\frac{n-1}{2}$ eine Primzahl ist.

Konkretes Beispiel mit $n = 5$:

Als erstes muß eine Primitivwurzel modulo n gefunden werden:

$$2^1 = 2 \text{ mod } 5$$

$$2^2 = 4 \text{ mod } 5$$

$$2^3 = 8 \text{ mod } 5 = 3 \text{ mod } 5$$

$$2^4 = 16 \text{ mod } 5 = 1 \text{ mod } 5$$

Alle Zahlen von 1 bis $(5 - 1)$ sind vorhanden $\Rightarrow 2$ ist eine Primitivwurzel modulo 5.

Nun muß das Protokoll durchgerechnet werden:

1. Alice wählt eine zufällige große Zahl $x = 3$ und schickt Bob $X = 2^3 \text{ mod } 5 = 3$.
2. Bob wählt eine zufällige große Zahl $y = 4$ und schickt Alice $Y = 2^4 \text{ mod } 5 = 1$.
3. Alice berechnet $K = Y^x \text{ mod } n = 1^3 \text{ mod } 5 = 1 = K$.
4. Bob berechnet $K' = X^y \text{ mod } n = 3^4 \text{ mod } 5 = 81 \text{ mod } 5 = 1 = K'$.

11. Wie funktioniert bei GSM Netzen die sichere Kommunikation zwischen Benutzer und Basisstation? Welches Problem wird dabei unerfreulicherweise nicht gelöst ? (2 Punkte)

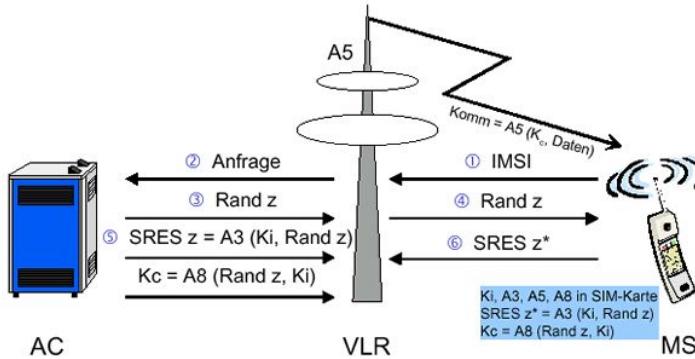


Abb. 11: Graphische Darstellung des Authentisierungsverfahrens in GSM-Netzen

In GSM Netzen wird mit dem A5 Algorithmus verschlüsselt, der als eher schwacher Algorithmus gilt (politische Diskussion bei der Entscheidung und der Kryptoanalyse von A5). Das ist ein Schieberegistergenerator mit linearer Rückkopplung (alle Crays haben einen Spezialbefehl zur LFSR Auswertung zur "staatlichen" Kryptoanalyse). A5 benutzt tatsächlich 3 LFSR mit unterschiedlichen Registerlängen.

In jeder SIM Karte ist ein Chip mit einer Seriennummer und einer geheimen Zahl Ki , die nicht ausgelesen werden kann (von Amateuren :-)). Im Chip existieren auch zwei Algorithmen A3 und A8 die jeder Netzbetreiber geheim hält. Ki haben auch die Netzbetreiber in ihren Rechnern gespeichert. Startet ein Teilnehmer einen Anruf, so sendet der Chip die Seriennummer - das Netzwerk sucht die entsprechende Zahl Ki und schickt eine Zufallszahl RAND als Antwort. Mittels A3 berechnet der Chip aus RAND und Ki eine 32-Bit Antwort SRES und schickt diese an die Basisstation. Diese verifiziert das, ist alles korrekt, geht der Anruf o.k. Aus RAND und Ki berechnen nun der Chip und die Basisstation einen 64-Bit Sessionkey Kc unter Verwendung von A8. Dieser Schlüssel wird für A5 Ent- und Verschlüsselung benutzt. Durch die korrekte Authentifizierung ist der Schlüssel natürlich identisch. Durch die Verwendung von verschiedenen Schlüsseln für jedes Gespräch relativiert sich die Unsicherheit von A5, da die Entschlüsselung sehr schnell sein müsste (1997 hätte ein Rechner mit 50 Spezialchips ca. 3 Stunden zum Knacken gebraucht - also auch nur eine Frage des Geldes !).

Problem:

Die Verschlüsselung endet und beginnt bei der Basisstation, also gibt es keine verschlüsselte End-to-End Kommunikation. Außerdem kann der Staat bei der Basisstation mithören.

Ein weiteres Problem wäre die mögliche Positionsbestimmung.

12. Erklären Sie wie die Methode „counting coincidences“ verwendet werden kann um die key-length bei Verschlüsselung mit XOR und einem kurzen Schlüssel zu ermitteln. Begründen sie auch warum diese Methode funktioniert. Geben sie ein Beispiel für eine Methode der Entschlüsselung wenn die Schlüssellänge ermittelt wurde. (2 Punkte)

Die Key length kann durch counting coincidences bestimmt werden. Dabei wird ein XOR des Ciphertextes auf geshiftete (verschobene) Varianten des Ciphertextes angewendet und die gleich bleibenden Bytes gezählt.

Mehrfaeches der Key length: 6% gleichbleibende Bytes

SONstige Positionen: 0.4% gleichbleibende Bytes

⇒ Index of coincidence.

Der kleinste Shift bis 6% ergibt die Keylength.

Begründung warum es funktioniert:

???

Mit bekannter Key-länge gibt es nun u.a. folgende zwei Methoden zur Entschlüsselung:

1. Bei gefundener Key-länge L liegt offenbar ein polyalphabetischer Cipher mit L Alphabeten vor, es gibt also offenbar L entsprechende monoalphabetische Ciphers. Werden die Ciphertext character jedes dieser Cipher zusammengruppiert lässt sich auf jeden dieser L monoalphabetischen Cipher eine (triviale) Häufigkeitsanalyse durchführen.
2. Wie oben ausgeführt wird durch das XOR zwischen geshifteten Ciphertextvarianten der Key entfernt wenn der Shift ein Vielfaches der Key-länge beträgt. Übrig bleibt der Plaintext ge-XORED mit einer geshifteten Variante der Plaintexts. Dies entspricht einem sog. “Text Autokey Cipher”, bei dem der Key für die XOR Berechnung aus dem Plaintext selbst abgeleitet wird. Klassischerweise wird vor dem Plaintext noch ein Key unbekannter Länge eingefügt, bevor dieser mit dem Plaintext ge-XORED wird. In unserem Fall ist der Key bekannt und besteht nur aus dem Shift um L character. Das Entschlüsseln der vorliegenden Daten basiert auf folgender Beobachtung and wird sukzessive durchgeführt:
 - Werden die Daten mit einem Key ge-XORED und der gewonnene Plaintext ist um L character in die andere Richtung verschoben identisch, so ist dieser Key korrekt ($P \oplus K \oplus K = P$). Durch eine dictionary Attacke mit charactern und Silben kommt man schnell zum Erfolg.

13. Erklären sie die Funktionsweise der S-Box Substitution in DES und vergleichen sie deren Funktionsweise mit der in AES verwendeten Substitution. In welcher Hinsicht besteht ein grundlegender Unterscheid (bezogen auf die Substitution aber auch anderer Elemente) zwischen den Designprinzipien von DES und AES ? (2 Punkte)

DES:

Die Substitutionsboxen (S-Boxen) beim DES sind standardisiert. Um aus den jeweiligen Tabellen der S-Boxen den Ausgabewert zu erhalten, wird der 6-Bit Eingabewert gesplittet. So bildet das erste und letzte Bit zusammen die Zeile, und die Spalte ergibt sich aus den übrigen 4 Bits. Eine Änderung dieser Boxen reduziert die Sicherheit drastisch!

Als Beispiel hier eine S-Box:

S₁		Mittlere 4 Bits des Eingabewertes															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Außere Bits	00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
	01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
	10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
	11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

Für die Zeile 10 und die Spalte 0100 müsste der Input also 1 0100 0 lauten.

AES:

- Byte Sub: jedes Byte im 128 Bits Block wird ersetzt durch einen in einer S-Box vordefinierten Wert ersetzt. Die S-Box entsteht durch ein Ersetzen eines Byte durch dessen Inverses im Galoisfeld GF(2⁸), anschließender bitweiser Matrixmultiplikation modulo 2 und abschließendem XOR mit Hexadezimal 63. Diese Operation weist hohe Nicht-Linearität auf und wird als Lookuptable implementiert.

Unterschiede die AES sicherer als DES machen:

- Während DES nur 56 Bit Schlüssel anbietet, gibt es bei AES 128, 192, oder 256 bit lange Schlüssel, was auch zu der höheren Sicherheit von AES beiträgt
- Die Block Size von DES beträgt nur 64 Bit, während AES mit 128 Bit arbeitet.
- Andere Implementierung der S-Box.
- AES wurde aus den Erfahrungen aus DES implementiert.

14. Erklären Sie die Grundidee der Quantenkryptographie und warum es beim beschriebenen Verfahren möglich ist zu erkennen ob ein Lauscher am Kanal war. (2 Punkte)

In der Quantenkryptographie wird mit der Polarisierung von Photonen gearbeitet. Die Polarisierung kann 4 verschiedene Ausprägungen haben: oben/unten, rechts/links, hauptdiagonal, nebendiagonal. Ein Polarisationsfilter erlaubt manchen Photonen unveränderten Durchtritt und bei manchen wird die Polarisierung verändert. Es gibt Rechtecksfilter (für oben/unten und rechts/links Polarisation) und Diagonalfilter. Wird ein hauptdiagonal oder nebendiagonal polarisiertes Photon durch einen Rechtecksfilter geschickt, wird dessen Polarisierung zufällig auf oben/unten oder rechts/links verändert.

Zusätzlich erlaubt Quantenkryptographie Intrusion Detection, also das Aufspüren eines möglichen Angreifers. Wenn Eve beim Lauschen am Photonenkanal an einer Position wo Bob korrekt misst den falschen Filter verwendet, verändert sie die Polarisierung des Photons. Alice und Bob müssen nun nur noch ein Subset des Keymaterials vergleichen (über den unsicheren Kanal) – stimmt es überein und ist das Subset ausreichend groß, hat niemand gelauscht.

15. Erklären Sie welche Bereiche der IP Pakete im Transport Mode beim ESP Protokoll von IP Secure authentifiziert bzw. verschlüsselt werden. Was ist Tunnel und Transport Mode? Was ist der Unterschied zwischen ESP und AH? (2 Punkte)

AH (Authentication Header): AH-Authentifizierungsprotokoll: hier werden die Daten und die invarianten Felder des äußeren IP Headers mit einem MAC - wegen der Geschwindigkeit statt einer Signatur - authentifiziert

ESP (Encapsulating Security Payload): kombiniertes Verschlüsselungs/Authentifizierungsprotokoll, bei ESP Authentifizierung wird der Header nicht authentifiziert, sondern nur die Payload.

Transport Mode: sichere End-to-End Kommunikation, Header bleibt unverändert, IP Payload wird geschützt.

Tunnel Mode: das gesamte IP Paket wird geschützt und mit einer neuen IP Header versehen. Wird für das Tunneling zwischen Routern verwendet.

16. Was ist Zero-Knowledge Protokoll und was ist der Unterschied zu z.B. klassischen Authentifizierungsprotokollen. Beschreiben sie detailliert die Funktionsweise des Feige-Fiat Schamir Identifikations Schemas und erklären sie die zero-knowledge Eigenschaft. (2 Punkte)

Ein Zero-Knowledge-Protokoll stellt eine Möglichkeit zur Verfügung, eine Partei von der Gültigkeit einer Aussage zu überzeugen, ohne dabei etwas anderes als die Gültigkeit der Aussage selbst preiszugeben.

Im Gegensatz zu klassischen Authentifizierungsprotokollen muß das Geheimnis in keiner Weise preisgegeben werden.

Funktionsweise des Feige-Fiat Schamir Identifikations Schemas:

Ein zufälliger Modul $n = pq$, $n \geq 512$ Bits.

v muß ein quadratisches Residuum modulo n sein ($x^2 = v \bmod n$ hat eine Lösung und v^{-1} existiert)

Keys:

- Peggies Public Key: v
- Peggies Private Key: s mit $s \equiv \sqrt{v^{-1}} \bmod n$

Protokollablauf:

1. Peggy wählt ein zufälliges $r < n$ und berechnet $x = r^2 \bmod n$. Dann schickt sie x an Viktor.

2. Viktor schickt Peggy ein zufälliges Bit b .

3. $b = 0$: Peggy schickt r .

$b = 1$: Peggy schickt $y = rs$.

4. $b = 0$: Victor überprüft, ob $x = r^2 \bmod n$ ist. D.h. Peggy kennt \sqrt{x}

$b = 1$: Victor überprüft, ob $x = y^2 v \bmod n$ ist. D.h. Peggy kennt $\sqrt{v^{-1}}$

Kennt Peggy s nicht, so kann sie r so wählen, dass die Viktor täuscht, wenn er $b = 0$ oder $b = 1$ schickt, aber nicht beides!. D.h. die Chance ist 50%.

Peggy und Viktor wiederholen das Protokoll, bis Viktor überzeugt ist, dass Peggy s kennt.

Zero-knowledge eigenschaft:

Peggy kann beweisen, dass sie s kennt, ohne s preis geben zu müssen!

17. Wie und mit welchen kryptographischen Primitiven wird eine digitale Signatur typischerweise erstellt und wie wird sie verifiziert? Erklären sie die Rolle der Komponenten und den Grund für ihre Verwendung. Geben sie konkrete Beispiele, welche Algorithmen für die verschiedenen Komponenten bei der Signaturerstellung verwendet werden können. (2 Punkte)

Digitale Unterschriften mit symmetrischen Kryptographiesystemen und einem Vermittler.

In diesem Fall garantiert ein außenstehender Vertrauensmann, dass eine Nachricht wirklich vom angegebenen Absender stammt. Angenommen Alice und Bob wollen Nachrichten untereinander austauschen. Der Vertrauensmann teilt einen geheimen Key K_A mit Alice und K_B mit Bob. Wenn Alice eine Nachricht an Bob schicken will, verschlüsselt sie diese mit K_A und schickt sie dem Vertrauensmann. Der Vertrauensmann entschlüsselt die Nachricht mit K_A und verschlüsselt sie anschließend mit K_B und schickt sie Bob, der sie dann mit K_B entschlüsselt. In diesem Szenario garantiert also der Vertrauensmann, dass die Nachricht wirklich von Alice ist.

Digitale Unterschriften mit Public Key

In diesem Fall ist der Privat Key für das Verschlüsseln und der Public Key für das Entschlüsseln. Die Unterschrift ist das Verschlüsseln mit dem Private Key. Die Verifizierung der Unterschrift ist das Entschlüsseln mit dem Public Key. Da nur der Absender die Nachricht mit seinem Private Key verschlüsseln kann, stellt das Entschlüsseln mit dem Public Key des Absenders sicher, dass die Nachricht wirklich vom Absender stammt.

Zusätzlich werden auch One-Way Hash Funktionen verwendet: Der Absender berechnet den Hash Value des Dokumentes, verschlüsselt diesen mit seinem Private Key und schickt das Dokument mit dem Unterschriftshash an den Empfänger. Der Empfänger berechnet nun den Hash Value des erhaltenen Dokuments, entschlüsselt den mitgeschickten Hash Value und vergleicht die beiden Hash Values. Wenn sie identisch sind, dann ist die Unterschrift gültig.

Digitale Unterschriften mit Verschlüsselung

Wie bei digitalen Unterschriften mit Public Key unterschreibt der Absender die Nachricht indem er diese mit seinem Private Key verschlüsselt. Zusätzlich verschlüsselt er die unterschriebene Nachricht mit dem Public Key des Empfängers:

$E_B(S_A(m))$. Der Empfänger entschlüsselt zuerst die Nachricht mit seinem Private Key und verifiziert die Unterschrift mit dem Public Key des Empfängers: $V_A(D_B(E_B(S_A(m))))$.

18. Was ist die „man in the middle attack“ (wie funktioniert sie) und wie kann eine Abhilfe aussehen? (2 Punkte)

Zur Erinnerung eine kurze Wiederholung:

1. Alice hat Bobs Public Key vom KDC.
2. Alice generiert den Session Key, verschlüsselt ihn mit Bobs Public Key und schickt ihn an Bob.
3. Bob entschlüsselt Aices Nachricht mit seinem Private Key.
4. Beide verwenden den Schlüssel zur Kommunikation.

Dabei gibt es eine bekannte Attacke Man in the Middle Attack:

1. Alice schickt Bob ihren Public Key. Mallory fängt den Key ab und schickt Bob seinen eigenen Public Key.
2. Bob schickt Alice seinen Public key. Wie vorher fängt Mallory den Key ab und schickt Alice seinen eigenen Public Key.
3. Wenn Alice an Bob eine Nachricht schickt, fängt Mallory diese ab, entschlüsselt sie, liest sie, verschlüsselt mit Bobs Key und schickt das Resultat an Bob.
4. Wenn Bob an Alice etwas schickt, handelt Mallory analog.

Abhilfe:

Interlock Protokoll:

1. Alice schickt Bob ihren Public Key.
2. Bob schickt Alice seinen Public Key.
3. Alice verschlüsselt ihre Nachricht mit Bobs Key und schickt die halbe Nachricht an Bob.
4. Bob verschlüsselt seine Nachricht mit Alices Key und schickt die Hälfte an Alice.
5. Alice schickt die zweite Hälfte.
6. Bob setzt die beiden Hälften zusammen und entschlüsselt sie. Dann schickt er seine zweite Hälfte.
7. Alice setzt die beiden Hälften zusammen und schlüsselt Bobs Nachricht.

Dieses Verfahren verhindert die „Man in the Middle“ Attacke, da die Hälfte einer Nachricht ohne die andere Hälfte nicht entschlüsselt werden kann.

19. Erklären sie die Common Modulus Attacke gegen RSA und wie sie verhindert werden kann. (2 Punkte)

Einige mögliche RSA Implementierung gibt jedem/r das gleiche n aber verschiedene e und d . Wird die gleiche Nachricht mit zwei relativ primen Exponenten verschlüsselt, kann der Plaintext ohne Private Key entschlüsselt werden.

$$\begin{aligned} c_1 &= m^{e_1} \pmod{n} \\ c_2 &= m^{e_2} \pmod{n} \end{aligned}$$

Der/die FeindIn kennt n, e_1, e_2, c_1, c_2 . $(e_1, e_2) = 1 \Rightarrow \exists r$ und s , sodaß $re_1 + se_2 = 1$, $r < 0$. Dann ist nur noch das folgende zu berechnen: c_1^{-1} und $(c_1^{-1})^{-r} \cdot c_2^s = m^{e_1 r} \cdot m^{e_2 s} = m \pmod{n}$.

Alternative Erklärung:

Angenommen eine Nachricht m wird zweimal, mit gleichem Modulus n aber verschiedenen Public Keys e_1, e_2 , verschickt. Wenn e_1, e_2 relativ prim zueinander sind ($\gcd(e_1, e_2) = 1$), dann $\exists x, y \in \mathbb{Z}$ sodass $x \cdot e_1 + y \cdot e_2 = 1$, welche mit dem erweiterten Euklidischen Algorithmus berechnet werden können. Die Nachricht m kann nun ohne Private Key entschlüsselt werden, weil $(m^{e_1})^x \cdot (m^{e_2})^y = m^{x \cdot e_1} \cdot m^{y \cdot e_2} = m^{x \cdot e_1 + y \cdot e_2} = m^1 = m \pmod{n}$.

Verhinderung:

Abhilfe dafür schafft die Vermeidung verschiedener e_1, e_2 für die gleiche Nachricht oder die Bedingung, dass $\gcd(e_1, e_2) \neq 1$.

20. Erklären sie die Grundfunktionalitäten von DNSSEC (was wird wie abgesichert, was nicht – im Gegensatz zu IPSec, welche zusätzlichen DNS records gibt es dafür) und beschreiben sie die wie die Problematik der Überprüfung der Korrektheit eines public domain-keys gelöst wird. (2 Punkte)

Die Domain Name System Security Extensions (DNSSEC) sind eine Reihe von Internetstandards, die das Domain Name System (DNS) um Sicherheitsmechanismen zur Gewährleistung der Authentizität und Integrität der Daten erweitern. Ein DNS-Teilnehmer kann damit verifizieren, dass die erhaltenen DNS-Zonendaten auch tatsächlich identisch sind mit denen, die der Ersteller der Zone autorisiert hat. DNSSEC wurde als Mittel gegen Cache Poisoning entwickelt. Es sichert die Übertragung von Resource Records durch digitale Signaturen ab. Eine Authentifizierung von Servern oder Clients findet nicht statt.

Vertraulichkeit ist bei DNSSEC nicht vorgesehen, DNS-Daten werden daher nicht verschlüsselt.

Zusätzliche DNS records:

- RRSIG (resource record signature)
- DNSKEY
- DS (delegation signer)
- NSEC (next secure record)
- NSEC3 (next secure record version 3)
- NSEC3PARAM (next secure record version 3 parameters)

Um eine sichere Authentifizierung zu gewährleisten, muss der Public Key einer Zone (bzw. dessen Hash) in den zentralen Server, der den Namen auflösen soll, manuell eingebracht werden. Da normalerweise jede Zone einen anderen Schlüssel besitzt, der sich zudem regelmäßig ändert, kann die Schlüsselverwaltung sehr aufwändig werden.

Die Bildung von Vertrauensketten (engl.: Chains of Trust) erleichtert das Keymanagement dramatisch. Eine möglichst hoch im DNS-Baum angesiedelte Zone enthält die Public Keys ihrer delegierten Subzonen und unterschreibt diese digital. Die Subzonen können wiederum die signierten Public Keys ihrer untergeordneten Zonen enthalten usw. Für eine derartige Chain of Trust muss im Resolver eines zentralen Nameservers lediglich der Public Key der obersten Zone bekannt sein. Die Gesamtmenge der durch einen einzigen Schlüssel gesicherten Menge von Zonen wird auch als Sicherheitsinsel (engl.: Island of Security) bezeichnet. Im Idealfall existiert nur eine einzige derartige Island of Security für den gesamten Namensraum und damit nur ein einziger Trusted Key.

21. Beweisen sie die Formel für die Euler'sche Phi-Funktion $\Phi(n)$ für eine Zahl n die das Produkt von zwei Primzahlen $n = p \cdot q$ ist: $\Phi(n) = (p - 1) \cdot (q - 1)$. (2 Punkte)

Beweis .

Es gibt q Vielfache von p , und p Vielfache von q innerhalb von $\{1, 2, \dots, p \cdot q\}$ und das einzige gemeinsame Vielfache von p und q ist $p \cdot q$.

Also $\phi(p \cdot q) = p \cdot q - p - q + 1 = (p - 1)(q - 1)$.

22. Beschreiben sie die in der VO besprochene Protokollattacke gegen (automatisierte) digitale Empfangsbestätigungen (verschlüsselt & signiert) und diskutieren sie deren Realitätsnähe. (2 Punkte)

Immer, wenn Bob eine Nachricht empfangen hat, schickt er sie als Bestätigung wieder zurück.

1. Alice unterschreibt, verschlüsselt und schickt $E_B(S_A(m))$.
2. Bob entschlüsselt und überprüft die Unterschrift $V_A(D_B(E_B(S_A(m)))) = M$.
3. Bob unterschreibt M mit seinem Private Key, verschlüsselt dann mit Alices Public Key und schickt das ganze an Alice zurück $E_A(S_B(m))$.
4. Alice entschlüsselt und überprüft die Unterschrift. Stimmen die Nachrichten überein, hat Bob die Nachricht korrekt bekommen.

Wenn für die Verschlüsselung und die Unterschriftenprüfung der gleiche Algorithmus verwendet wird, gibt eine mögliche Attacke:

$$V_x = E_x \quad \text{und} \quad S_x = D_x$$

Mallory besitzt einen eigenen Private und Public Key. Er liest Bobs Mail und behält die Nachricht von Alice an Bob. Nach einiger Zeit schickt er diese an Bob mit der Behauptung, daß sie von ihm käme. Bob entschlüsselt die Nachricht mit seinem Private Key und versucht Mallorys Unterschrift mit dessen Public Key zu überprüfen. Das ergibt das folgende Resultat:

$$E_M(D_B(E_B(D_A(m)))) = E_M(D_A(m))$$

Bob folgt dem Protokoll und schickt Mallory eine Empfangsbestätigung (receipt).

$$E_M(D_B(E_M(D_A(m))))$$

Mallory entschlüsselt nun mit seinem Private Key, verschlüsselt mit Bobs Public Key, entschlüsselt mit seinem eigenen Private Key und verschlüsselt mit Alices Public Key. Damit hat er die Nachricht M .

Realitätsnähe:

Es ist nicht unrealistisch bei automatischen Empfangsbestätigungen und deswegen gibt es diverse Verbesserungen wie:

- Verwendung verschiedener Operationen (Algorithmus oder Key),
- Timestamps,
- Unterschriften mit One-Way Hash Funktionen.

23. Wie funktioniert die Geburtstagsattacke gegen one-way Hash functions? Worauf beruht sie? Abhilfe ? (2 Punkte)

- Alice besitzt zwei Versionen eines Vertrages, wobei eine die echte ist, und die andere zu Ungunsten von Bob verändert worden ist.
- Alice verändert beide geringfügig und berechnet die Hash-Werte der veränderten Dokumente (2^{32} verschiedene Dokumente).
- Alice sucht ein Paar von Hash-Werten, die gleich sind, und rekonstruiert die Dokumente.
- Alice und Bob unterschreiben den Vertrag, der für Bob in Ordnung ist mit einem Protokoll, bei dem er den Hash des Dokumentes unterschreibt.
- Irgendwann ersetzt Alice die Originalversion mit der anderen - equivalent unterschrieben von Bob!!

Worauf beruht sie:

Der Hashwert ist m Bits lang. Um einen vorgegebenen Hashwert zu erzeugen, müssen 2^m Nachrichten erzeugt werden. Damit zwei beliebige Nachrichten den selben Wert haben, muss man nur $2^{\frac{m}{2}}$ erzeugen.

Abhilfe:

- Hashlänge von 64 Bit auf 128 Bit erhöhen
- Hash erzeugen und an Nachricht anhängen (Kann beliebig oft wiederholt werden)

24. Erklären Sie die Betriebsweise von Blockciphern (ECM, CBC, CFB, OFB), deren Vor- und Nachteile und ihre Verhalten bei Ciphertextfehlern bzgl. Fehlerausbreitung. (2 Punkte)

ECM Electronic Codebook Mode

Ein Block Plaintext wird in einen Block Ciphertext überführt.

Vorteile :

- Die Reihenfolge muß bei der Ver- bzw. Entschlüsselung nicht eingehalten werden. Dies eignet sich daher besonders gut für Verwendung in Datenbanken bzw. für die Parallelisierung.
- Fehler im Code betreffen nur einen spezifischen Block. Wenn Blockgrenzen kontrolliert werden, ist auch ein Fehler von Bits kein Problem.

Nachteile :

- Der Vorteil der einzelnen Blöcke macht diesen Modus anfälliger für diverse Attacken. Es kann ein Codebook aller Plaintext- Ciphertext Paare erstellt werden. Große Gefahr ergibt sich daher vor allem am Anfang und am Ende von Nachrichten, da sich hier bei bestimmten Texttypen oft wiederholende Stereotype befinden, z.B. Begrüßung.
- Block Replay: Mallory könnte die verschlüsselten Nachrichten verändern, ohne daß es bemerkt wird.

CBC Cipher Block Chaining Mode

Ein zusätzlicher Feedback Mechanismus wird eingefügt. Die Ergebnisse der Verschlüsselung des vorherigen Blocks werden zur Verschlüsselung des aktuellen Blocks verwendet. Jeder Ciphertext Block hängt nicht nur vom entsprechenden Plaintext Block ab, sondern auch von allen vorherigen Plaintext Blöcken.

Auf den Plaintext Block wird mit dem vorhergehenden Ciphertext Block ein XOR angewendet, ehe er verschlüsselt wird.

Nachteile Zwei identische Texte werden gleich verschlüsselt. Wenn sie sich ab einem bestimmten Punkt unterscheiden, unterscheiden sich auch die beiden Ciphertexte an der genau gleichen Stelle.

Errorpropagation Ein Fehler im Plaintext betrifft zwar den gesamten Ciphertext, beim Entschlüsseln bleibt aber trotzdem nur der ursprüngliche Fehler zurück.

Ciphertext Fehler: Ein Bit zerstört den gesamten Block und ein Bit des nächsten Plaintextes an der Fehlerstelle. Der zweite Block nach dem fehlerhaften Bit ist allerdings wieder korrekt („self recovering“).

CFB Cipher Feedback Mode

Im CBC Mode muß gewartet werden, bis der gesamte Block vorhanden ist. Bei dieser Variante genügt es, wenn n Bits zur Verfügung stehen.

Ein Block Algorithmus im CFB Mode arbeitet mit einer Queue die genau Blockgröße hat. Zu Beginn wird die Queue mit einem Initialisierungsvektor gefüllt. Die Queue wird verschlüsselt und auf die n -Bit am linken Rand der Queue wird mit dem Plaintext ein XOR angewendet und an das rechte Ende der Queue gestellt. Alle anderen Bits der Queue werden nach links verschoben. Dann kommt der nächste n -Bit Block an die Reihe.

Die Entschlüsselung ist genau umgekehrt, aber beide im Verschlüsselungsmodus.

Der Initialisierungsvektor muß bei verschiedenen Anwendungen immer verschieden sein!

Errorpropagation Analog zum CBC Mode betrifft ein Plaintextfehler den gesamten Ciphertext. Nach der Entschlüsselung ist der Fehler aber wieder auf die Ausgangsstelle reduziert.

Ciphertext Fehler: Ein Fehler im Ciphertext verursacht einen Fehler im Plaintext. Die Fehlerstelle wird im Shift-Register aber langsam über äußerem Rand geschoben. Nach dem „Hinauswurf“ des Fehlers, erholt sich das System wieder.

OFB Output Feedback Mode

Das Verfahren ist dem CFB sehr ähnlich und verwendet ebenfalls Feedback. Allerdings werden hier n -Bits des verschlüsselten Shift Registers wieder im Shift Register verwendet. Die Entschlüsselung erfolgt wieder genau umgekehrt zur Verschlüsselung. Hier ist der Feedback u. a. von Plain- und Ciphertext, wird auch als „internal feedback“ bezeichnet.

Vorteile Der gesamte Verschlüsselungsvorgang kann geschehen, bevor der Plaintext vorhanden ist, da nur noch ein XOR angewendet wird.

Errorpropagation Hier gibt es keine Error Extension. Ein Bit-Fehler im Ciphertext bewirkt einen Bit-Fehler im Plaintext.

25. Was ist die „meet in the middle attack“ (wie funktioniert sie) und was ist die Konsequenz ihrer Existenz ? (2 Punkte)

CryptoanalystIn kennt P_1, C_1, P_2, C_2 , sodaß

$$C_1 = E_{K_2}(E_{K_1}(P_1))$$

$$C_2 = E_{K_2}(E_{K_1}(P_2))$$

Für alle möglichen $K(K_1$ oder $K_2)$ wird $E_K(P_1)$ berechnet und im Speicher abgelegt. Danach wird $D_K(C_1)$ für alle K berechnet und das identische Ergebnis im Speicher gesucht. Wird es gefunden, kann der aktuelle Schlüssel K_2 sein und der im Speicher K_1 .

Wenn nun versucht wird, P_2 mit K_1 und K_2 zu verschlüsseln und man bekommt C_2 , so ist ziemlich sicher, daß K_1 und K_2 gefunden wurden. Wenn nicht, wird einfach weiter gesucht. Ein Problem ist allerdings der Speicherbedarf von 2^n Blocks. Bei DES : 2^{56} 64-Bit Blöcke. Das sind etwa 10^{17} Bytes.

Konsequenz ihrer Existenz

???

26. Erklären sie wie das Knapsack-Problem verwendet werden kann, um ein public-key Verschlüsselungsverfahren zu realisieren. Warum werden solche Systeme heute nicht mehr eingesetzt ? (2 Punkte)

1. Suche ein einfaches Knapsack Problem.
2. Wähle zwei Zahlen n und m mit $m > \sum M_i$ und $(n, m) = 1$.
3. Multipliziere alle M_i mit $n \bmod m$. Die daraus resultierenden M'_i bilden dann ein schwieriges Knapsack Problem mit der selben Lösung.

Beispiel: $K : \{2, 3, 6, 13, 27, 52\}, m = 105, n = 31$
 $\rightarrow K' : \{62, 93, 81, 88, 102, 37\}$.

Verschlüsselung: Die Nachricht wird in Blöcke in der Länge des Knapsacks aufgeteilt.

Entschlüsselung: Der Private Key (=superincreasing Knapsack) sowie n und m müssen bekannt sein. Somit kann n^{-1} ($nn^{-1} \equiv 1 \bmod m$) bestimmt und jeder Ciphertextwert mit $n^{-1} \bmod m$ multipliziert werden.

Beispiel: Für die Verschlüsselung muß der Ciphertext mit dem Public Key erzeugt werden. $K' : \{62, 93, 81, 88, 102, 37\}$:

$$\begin{array}{ll} 011000 & 93 + 81 = 174 \\ 110101 & 62 + 93 + 88 + 37 = 280 \Rightarrow \text{Ciphertext } 174, 280, 333. \\ 101110 & 62 + 81 + 88 + 102 = 333 \end{array}$$

Für die Entschlüsselung werden Private Key n und m benötigt:

$$\begin{array}{llll} K : \{2, 3, 6, 13, 27, 52\}, m = 105, n = 31, n^{-1} = 61 & & & \\ 174(61 \bmod 105) & = & 9 & = & 3 + 6 & 011000 \\ 260(61 \bmod 105) & = & 70 & = & 2 + 3 + 13 + 52 & 110101 \\ 333(61 \bmod 105) & = & 48 & = & 2 + 6 + 13 + 27 & 101110 \end{array}$$

Diese Art von Verschlüsselung ist sehr unsicher. Die Sicherheitslücke ist die Erzeugung des schweren Knapsack Problems. Es wurde demonstriert, dass der Merkle-Hellman Algorithmus in 15 Minuten geknackt werden kann (Shamir & Zippel). Selbst ein iteriertes Verfahren, bei dem die Gewichte mehrfach mit unterschiedlichen Paaren von Multiplikatoren und Moduln transformiert werden, um das 'schwierige' Rucksackproblem zu generieren, kann erfolgreich mit polynomialem Aufwand angegriffen werden.

27. Erklären sie die Begriffe „Ciphertext-only Attacke“ und „Known-Plaintext Attacke“ und erklären sie genau welches Wissen und welche Voraussetzungen man für diese Attacken braucht. Gegen welche Art von Attacke sind Public-key Verfahren grundsätzlich immer anfällig und warum? (2 Punkte)

Ciphertext-only Attack: Der Feind kennt n Ciphertexte von n Plaintexten, die mit dem gleichen Algorithmus verschlüsselt wurden.

Gegeben: Ciphertexte C_1, \dots, C_n .

Gesucht: Plaintexte P_1, \dots, P_n oder ein Algorithmus mit dem P_{n+1} aus C_{n+1} berechnet werden kann.

Known Plaintext Attack: Der Feind kennt n Paare von Plaintexten und Ciphertexten.

Gegeben: Plaintext/Ciphertext Paare $(P_1, C_1), \dots, (P_n, C_n)$.

Gesucht: Schlüssel oder ein Algorithmus mit dem P_{n+1} aus C_{n+1} berechnet werden kann.

Public Key Verfahren sind grundsätzlich immer gegen Known Plaintext Attacken anfällig, weil der Public Key sowie das Kryptographiesystem bekannt ist, kann der Feind beliebige Plaintexte mit dem Public Key verschlüsseln und daraus die Ciphertexte gewinnen.

28. Erklären sie drei „computational infeasible problems“ die die Grundlage von public-key Verfahren bilden (auch warum das schwierige Probleme sind) und beschreiben sie in welcher Form sie bei je einem konkreten Algorithmus vorkommen (also warum sie konkret die Sicherheitsgrundlage dieser Verfahren bilden). (3 Punkte)

Problem der Faktorisierung

Gegeben: Sehr großes $n \in \mathbb{N}$. Gesucht: Primfaktoren von n . Problem: Bis heute gibt es keinen effizienten Algorithmus für die Faktorisierung von n . Besonders schwer zu faktorisieren ist das Produkt zweier sehr großer Primzahlen.

Public Key Verfahren das auf dem Problem der Faktorisierung aufbaut: RSA

Problem der Berechnung von diskreten Logarithmen

Gegeben: Elemente a, b einer Gruppe. Gesucht: Eine ganze Zahl x sodass $a^x = b$.

Problem: Bis heute gibt es keinen effizienten Algorithmus für die Berechnung allgemeiner diskreter Logarithmen. Besonders schwer zu berechnen sind diskrete Logarithmen in der Restklassengruppe mit Modulo Multiplikation.

Public Key Verfahren das auf dem Problem der Berechnung diskreter Logarithmen aufbaut: Diffie-Hellman

Knapsack Problem

Gegeben: Eine Folge von Werten (M_1, M_2, \dots, M_r) und eine Zahl S . Gesucht: Eine Kombination von Werten der Folge sodass deren Summe, S ist. Das Knapsackproblem ist NP-vollständig, d.h. es gibt keinen Algorithmus der die Lösung in polynomialer Zeit findet.

Public Key Verfahren das auf dem Knapsack Problem aufbaut: Merkle-Hellman Algorithmus

29. Welche Zahl im Restklassensystem Modulo 5 entspricht der Ausdruck $-\frac{7}{4} \pmod{5}$?

Erklären Sie die Berechnungsschritte. (1 Punkt)

1. Als erstes hole ich den Ausdruck in den Positiven Bereich:

$$-\frac{7}{4} \pmod{5} = -7 \cdot \frac{1}{4} \pmod{5} = (-7 + 5 + 5) \cdot 4^{-1} \pmod{5} = 3 \cdot 4^{-1} \pmod{5}$$

2. Dann muss das Inverse von 4 ($\pmod{5}$) berechnet werden:

$$4 \cdot 4 = 16 \pmod{5} = 1 \pmod{5} \rightarrow 4 = 4^{-1} \pmod{5}$$

3. Zum Schluß wird alles zusammengerechnet:

$$3 \cdot 4 \pmod{5} = 12 \pmod{5} = 2 \pmod{5}$$

30. Erklären sie den algorithmischen Ablauf einer MD-5 Hash-Generierung. Wie ist es uns aus heutiger Sicht um die Sicherheit von MD-5 bestellt? Gibt es Alternativen ? (2 Punkte)

MD5 basiert auf der Merkle-Damgård Konstruktion um aus einer Nachricht variabler Länge eine Ausgabe fester Länge (128 Bit) zu erzeugen. Zuerst wird eine Eins an die Ausgangsnachricht angehängt. Danach wird die Ausgangsnachricht mit Nullen so aufgefüllt, dass ihre Länge 64 Bits davon entfernt ist durch 512 teilbar zu sein. Nun wird eine 64-Bit-Zahl, die die Länge der Ausgangsnachricht codiert, angehängt. Die Nachrichtenlänge ist jetzt durch 512 teilbar.

Der Hauptalgorithmus von MD5 arbeitet mit einem 128-Bit-Puffer, der in vier 32-Bit-Wörter A, B, C und D unterteilt ist. Diese werden mit bestimmten Konstanten initialisiert. Auf diesen Puffer wird nun die Komprimierungsfunktion mit dem ersten 512-Bit-Block als Schlüsselparameter aufgerufen. Die Behandlung eines Nachrichtenblocks geschieht in vier einander ähnlichen Stufen, von Kryptografen „Runden“ genannt. Jede Runde besteht aus 16 Operationen, basierend auf einer nichtlinearen Funktion „F“, modularer Addition und Linkssrotation. Es gibt vier mögliche „F“-Funktionen, in jeder Runde wird davon eine andere verwendet:

$$F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$

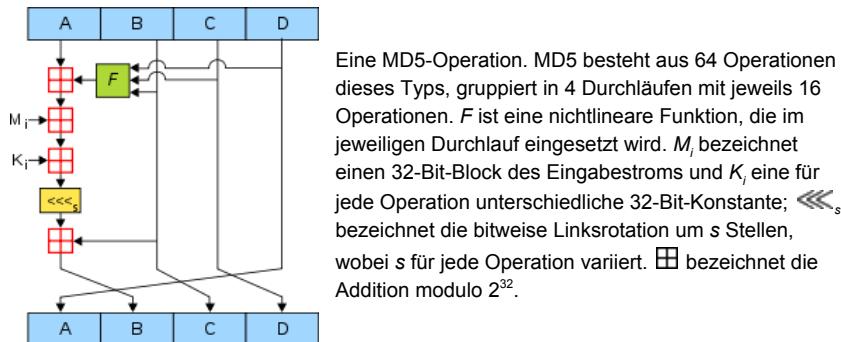
$$G(X, Y, Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$

$$H(X, Y, Z) = X \oplus Y \oplus Z$$

$$I(X, Y, Z) = Y \oplus (X \vee \neg Z)$$

\oplus , \wedge , \vee , \neg stehen jeweils für XOR, AND, OR und NOT-Operationen.

Auf das Ergebnis wird dieselbe Funktion mit dem zweiten Nachrichtenblock als Parameter aufgerufen usw., bis zum letzten 512-Bit-Block. Als Ergebnis wird wiederum ein 128-Bit-Wert geliefert – die MD5-Summe.



Derzeit ist MD5 nur bezüglich der Kollisions-Angriffe gebrochen. Deswegen besteht noch keine akute Gefahr für Passwörter, die als MD5-Hash gespeichert wurden. Diese Kollisionen sind eher eine Gefahr für digitale Signaturen.

Aktuell wird empfohlen auf SHA-2 bzw. SHA-3 umzusteigen.