Zusammenfassung Krypto

David Jäggli

15. Juni 2024

Inhaltsverzeichnis

1 Allg

Allgemeine Begriffe und Definitionen.

1.1 Terminologie

Kryptographie	Entwerfen von Krypto-Algorithmen
Kryptoanalyse	Brechen von Krypto-Algorithmen
Perfekte Sicherheit	Unendlich viele Ressourcen sind equivalent zu raten
Unkeyed Kryptographie	Hashfunktionen
Symmetrische Krypt.	Beide den gleichen Schlüssel - $\mathcal{O}(n^2)$
Asymmetrische Krypt.	Öffentlicher und privater Schlüssel - $\mathcal{O}(n)$
Stromchiffren	Verschlüsselung von einzelnen Zeichen
Blockchiffren	Verschlüsselung von Blöcken
MAC	Message Authentication Code
DES	Data Encryption Standard
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
CA	Certificate Authority
CRL	Certificate Revocation List - Zertifikatssperrliste
Mutual	Two-way Authentication. Beide Parteien authentifizieren sich
EWF (mit) Trapdoor	Es existert Geheimnis (Trapdoor) mit der sich $f^{-1}(y)$ einfach berechnen lässt. Die Sicherheit ist, dass dieser "Trick" nur einer Partei bekannt ist.

1.2 Trapdoor Weiterführung

Untenstehend sind die Unterschiede von Einwegfunktionen mit und ohne Trapdoor näher erläutert.

1.2.1 Einwegfunktion ohne Trapdoor

Beispiele und Erläuterungen für Einwegfunktionen ohne Trapdoor:

- Diffie-Hellman $y = f(x) = a^x \mod p$.
  - p prim ist rechenintensiv aber einfach.
  - Das Inverse (disk. Log.) $x \equiv f^{-1}(x) \equiv \log_a(y) \mod p$ ist für alle schwierig.
- Elliptische Kurven
  - Multiplikation $Q = k \cdot P$, k eine unbekannte Zahl, P,Q bekannte Punkte.
  - Das Bestimmen von k aus $Q = k \cdot P$ ist für alle schwierig.
  - Die Punkt-Division von $k = \frac{Q}{P}$ gibt es in diesem Sinne nicht.

1.2.2 Einwegfunktion mit Trapdoor

Beispiele und Erläuterungen für Einwegfunktionen mit Trapdoor:

- Bei RSA: Die e-te Wurzel mod N berechnen: Trapdoor
  - Das Potenzieren $y = f(x) \equiv x^e \mod N$ ist rechenintensiv aber einfach.
  - Das Inverse $x \equiv f^{-1}(x) \equiv y^{1/e} \equiv \sqrt[e]{y} \mod N$ ist extrem schwierig.
  - Das Berechnen der "e-ten Wurzel mod N" ist das eine Problem, das Faktorisieren von $N = p \cdot q$ das andere. Der RSA kann gebrochen werden, wenn nur eines der zwei Probleme gelöst ist.

2 Schutzmechanismen

Kleine Übersicht:

Sicherheitsdienst / Krypt. Mechanismus	Vertraulichkeit	Datenintegrität	Partnerauthentizität	Keiner dieser Dienste
Sym. Verschlüsselung	X			
Asym. Verschlüsselung	X			
Diffie-Hellman Schlüsselaustausch Protokoll				X
Hybride Verschlüsselung	X			
MAC- Berechnung		X		
Digitale Signatur		X		
C-R Protokoll mit MAC			X	
C-R Protokoll mit digitaler Signatur			X	

Abbildung 1: Mechansimus vs Sicherheitsdienst

2.1 Sicherheitsanforderungen

- Vertrauchlichkeit (Abhören)
- Integrity (Daten Verändern)
- Insertion (Daten Einfügen)
- Non repudiation of origin (Abstreiten die Meldung geschickt zu haben)
- Replay (Abfangen und wieder senden)
- Delete
- Non repudiation of receipt (Abstreiten die Meldung erhalten zu haben)
- Authentisierung

2.2 Geheimhaltung / Verschlüsselung

K_e = Key Encrypt
K_d = Key Decrypt

Ist $K_e = K_d \rightarrow$ symmetrische Verschlüsselung.
Ist $K_e \neq K_d \rightarrow$ asymmetrische Verschlüsselung.

2.3 Authentizität

K_g = Key Generate
K_v = Key Verify

Ist $K_g = K_v \rightarrow$ MAC.
Ist $K_g \neq K_v \rightarrow$ digitale Signatur.

3 Symmetrische Kryptographie

Note: Stromschiffren bieten keine Integrität.
Note: Blockchiffren bieten Integrität.
Ausserdem können Blockchiffren verwendet werden für:

- Hashfunktionen
- Pseudo random number generators
- Message Authentication Code (MAC)

3.1 Beschreibung von Blockchiffren

Grundsätzlicher Ablauf/Aufbau:

1. n-Bit Inputblockgrösse
2. n-Bit Outputblockgrösse
3. k-Schlüsselbit
4. x-Verschlüsselungsdurchläufe

übliche Blockgrössen: 64, 128, 256 Bit
Anzahl Runden: 10, 12, 14, 16
Übliche Schlüssellängen: k = 56 (DES), 112 (3DWS_2key),
k = 128 (AES), 168 (3DES_3key)
k = 192, 256 (AES) Bit

3.1.1 Eigenschaften von Blockchiffren

1. (Gleicher Schlüssel) Änderung eines Input bits ändert die Hälfte der Outputbits
2. (Gleicher Input) Änderung eines Schlüsselbits ändert die Hälfte der Outputbits
3. Mit Binomialkoeffizent kann gezeigt werden, dass diese Eigenschaften ideal sind.

Allgemein ist folgender Binom Ideal wenn $m = n/2$:

$$\binom{n}{m} = \frac{n!}{m! \cdot (n-m)}$$

3.1.2 Good to know

- 3DWES_2key (112 Bit Schlüssellänge) hat Sicherheit von 57-60 Bit
- 3DES_3key (168 Bit Schlüssellänge) hat Sicherheit von 112 Bit
- AES braucht für 128 Bit 10 Runden, für 192 Bit 12 Runden und für 256 Bit 14 Runden
- AES braucht für Entschlüsselung doppelt so lange wie für Verschlüsselung
- AES hat nicht den gleichen Algorithmus für Ver-/Entschlüsselung
- PRESENT ist optimiert für 8-Bit Prozessor (IoT)
- PRESENT braucht weniger Energie, SW & HW

3.2 Stromchiffren

XOR - Ziemlich straight forward, aufgrund von Key wird ein Pseudo-Random Sequence generiert, mit der dann der Plaintext verschlüsselt (XORd) wird. Entschlüsselung genau gleich
Abkürzungen: C = Cipher
M = Message
S = Sequence

Verschlüsselung: $C = M \oplus S$
Entschlüsselung: $M = C \oplus S = (M \oplus S) \oplus S = M \oplus (S \oplus S) = M \oplus 0 = M$

Charakteristiken:

- Keygenerator
  - (Bankenwelt) of ein Blockverschlüssler
  - (Netwerkwelt) of spezielle Konstruktion mit Hashfunktionen wie SHA-1
- Symmetrisch
- Vor allem bei Link-Verschlüssler als HW erhältlich.
- Keine Authentizität/Integrität
- Ein Verändern im Chiffretext verändert nur die betroffenen Zeichen
- Wird jedoch ein Bit hinzugefügt oder entfernt, ist die Hölle los

3.2.1 One-Time-Pad (OTP)

True-random Key, welcher gleich lang ist wie der Plaintext → perfekte Sicherheit.

OTP ist eine Stromchiffre, nicht jede Stromchiffre ist ein OTP.

3.2.2 XOR

Zwei gleiche Ausdrücke XORen ergibt 0.
Beispiel:
$6B \oplus A5 = CE = (6XORA)(BXOR5)$

3.2.3 Beurteilung von Stromchiffren

1. Schutzziele: Schutz gegen das Abhören der Meldung
2. Mit was für Typen / Angriffen muss gerechnet werden: gegen intelligente Gegner, welche passiven Angriff (abhören) durchführen
3. Stromchiffre Mechanismus: Symmetrische Verschlüsselung

3.3 Blockchiffren

Guess what, verschlüsseln immer einen Block und hängen diese hintereinander.

3.3.1 ECB-Modus für mehrere Blöcke

Electronic Code Book Mode, einfach alle Blöcke einzeln verschlüsseln.

$$C_i = E(M_i, K)$$

Gleiche Blöcke ergeben gleiche Chiffretexte.

3.3.2 CBC-Modus für mehrere Blöcke

Cipher Block Chaining, XOR des vorherigen Chiffretextes (oder IV) mit dem Plaintext. Respektive braucht einen IV (Initialisierungsvektor) für den ersten Block. Danach dient der Chiffretext des vorherigen Blocks als IV für den nächsten Block.

$$C_i = E(M_i \oplus C_{i-1}, K) \text{ mit } C_0 = IV$$

Gleiche Blöcke ergeben nicht mehr gleiche Chiffretexte.

Wenn die Länge des Plaintextes nicht durch die Blockgrösse teilbar ist, muss der letzte Block anders behandelt werden. Zum Beispiel einfach abschneiden. Die Entschlüsselung ändert nicht.
Die Formel für den letzen Block ist: $C_4 = E(C_3, K) \oplus M_4$

3.3.3 OFB-Modus = Output Feedback Mode

Ist grundsätzlich eine Stromchiffre. Hier wird Blockchiffre wird nur als Zufallszahlengenerator verwendet.

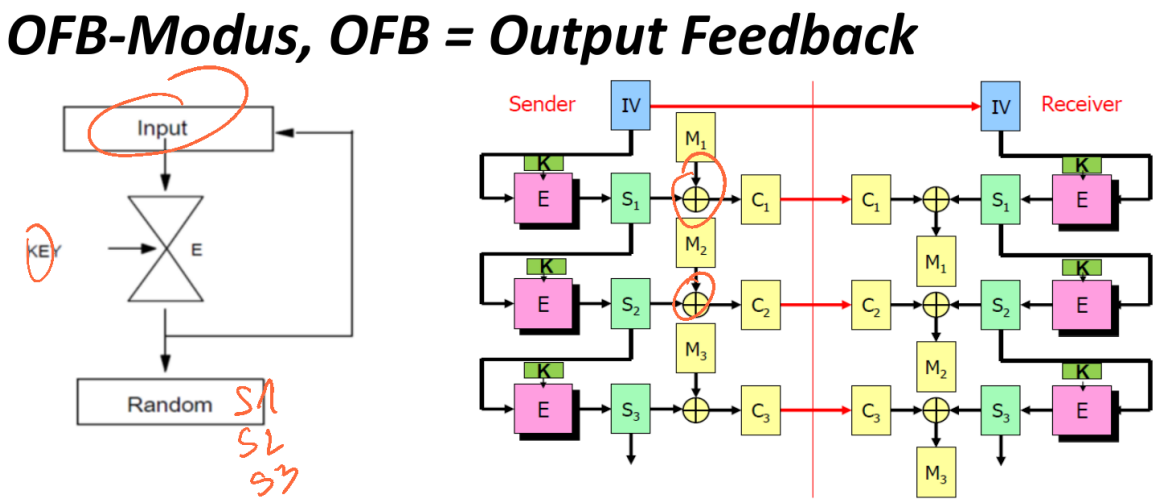

Abbildung 2: OFB Mode

Verschlüsseln:

$$C_i = S_i \oplus M_i \text{ mit } S_i = E(S_{i-1}, K) \text{ und } S_0 = IV$$

Entschlüsseln:

$$M_i = S_i \oplus C_i \text{ mit } S_i = E(S_{i-1}, K) \text{ und } S_0 = IV$$

3.3.4 CTR-Modus (Counter Mode)

Der XOR Schlüssel ist nicht mehr der vorherige (verschlüsselte) Block sondern einfach $IV + i$.
Hat Vorteile von beiden:

1. Ist parallelisierbar
2. Verschlüsselung eines grossen Files / Harddisk möglich
3. Gleicher Klartext ergibt nicht gleicher Output
4. Vorausberechnung von Schlüsselmaterial ist möglich
5. Vertauschen von Chiffratblöcken ist nicht mehr einfach.

Mathematische Form: $C_i = S_i \oplus M_i$ mit $S_i = E(IV + (i-1), K)$

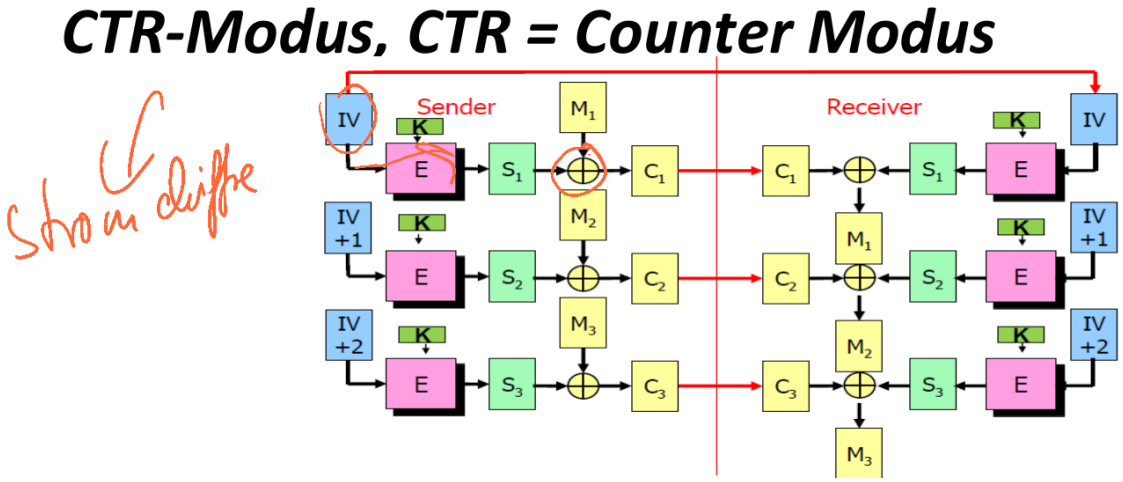

Abbildung 3: CTR Mode

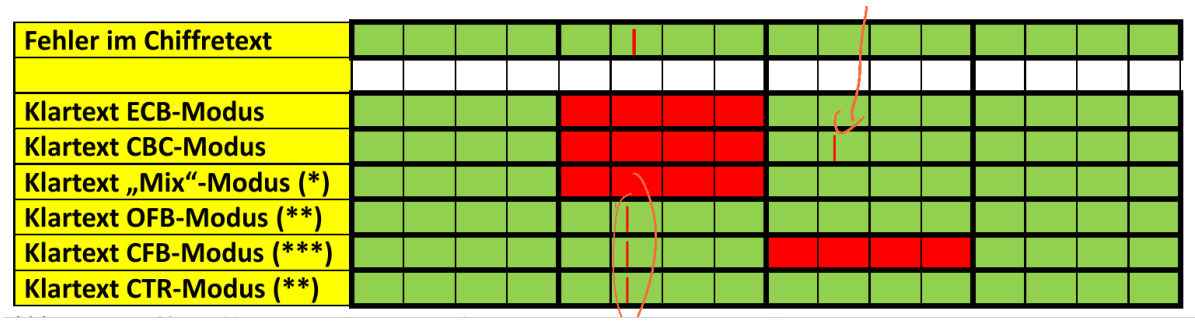

Abbildung 4: Auswirkung auf Klartext bei Bit Manipulation im Chiffretext

3.4 Auswirkungen bei Bit Manipulation im Chiffretext

3.5 Integritätsschutzmechanismen

MAC & HMAC sind Integritätsschutzmechanismen.

MAC halt irgendwie mit CBC oder CTR Mode und HMAC mit Hashing Algorithmus.

4 Asymmetrische Kryptographie

4.1 Allgemein

Sicherheitsvergleich klassisch:

Symmetric	**56**	**80**	**112**	**128**	**192**	**256**
RSA n	512	1024	2048	3072	7680	15360
ECC p	112	160	224	256	384	512
Key-Ratio	5:1	6:1	9:1	12:1	20:1	30:1

Sicherheit mit Quantencomputer:

1. Für Faktorosierung bei RSA K braucht ca $2k$ Qubits
2. Für DL bei ECC von k-Bits brauchts ca. $K \approx 5k + 8\sqrt{k} + 4 \cdot \log_2(k)$ Qubits

4.2 Einwegpermutationen

RSA, ECC und Diffie-Hellman sind genau genommen Einwegpermutationen.

Die Farben blau, grün und rot kann man auf $3! = 6$ Arten anordnen:

(B, G, R), (B, R, G), (G, B, R), (G, R, B), (R, B, G), (R, G, B)

Im RSA-System:
$N = 3 \cdot 11$ und $e = 13 \Rightarrow d = 13^{-1} \mod \varphi(N) = 17 = 13^{-1} \mod 20$

Nun kann man alle Werte $x \in \mathbb{Z}_{33} = \{0, \ldots, 32\}$ mit $y \equiv x^{13} \mod 33$ berechnen verschlüsseln.

Beispielaufgabe:
Die 3-stellige Zahl x wird mit der Formel $y \equiv (a \cdot x + b) \mod N$ verschlüsselt. Die Entschlüsselungsfunktion lautet: $x \equiv a^{-1} \cdot (y - b) \mod N$. Dabei ist $N = 11 \cdot 23 \cdot 41$ ein Produkt von drei Primzahlen; der Wert N ist öffentlich bekannt. Die Werte a und b bilden in der Form $(a; b)$ den geheimen Schlüssel. Die Werte a und b sind für die Verschlüsselung und Entschlüsselung geeignete Werte aus der Menge $\{2; 3; \ldots; N - 1\}$

Aus wie vielen möglichen Schlüsseln der Form $(a; b)$ können bei dieser Verschlüsselung ausgewählt werden? Es ist die exakte Zahl anzugeben.

Allgemein:

- Für b sind alle möglichen Werte aus der Menge $\{2; 3; \ldots; N-1\}$ erlaubt. Das sind $N - 2$ Möglichkeiten.
- Für a sind aus der Menge $\{2; 3; \ldots; N-1\}$ alle Werte erlaubt, die teilerfremd zu N sind. Da die Zahl 1 aber nicht drin sein darf lautet die Anzahl der möglichen Werte für a somit: $\varphi(N)-1 = \varphi(r \cdot s \cdot t)-1 = \varphi(r) \cdot \varphi(s) \cdot \varphi(t)-1 = (r-1) \cdot (s-1) \cdot (t-1)-1$
- somit gibt es total $(N - 2) \cdot [(r - 1) \cdot (s - 1) \cdot (t - 1) - 1]$ Möglichkeiten

Mit Zahlen:
$r = 11; s = 23; t = 41 \Rightarrow N = 11 \cdot 23 \cdot 41 = 10'373$

$\Rightarrow N - 2 = 10'371$
$\Rightarrow \varphi(N) - 1 = \varphi(r \cdot s \cdot t) - 1 = (11 - 1) \cdot (23 - 1) \cdot (41 - 1) - 1 = 10 \cdot 22 \cdot 40 - 1 = 8'799$

4.3 Elliptic Curve Cryptography (ECC)

- Was: mathematische Objekte, die man als Public-Key Kr. verwenden kann
- Warum:
  - wenige und schnelle Operationen (statt viele langsame wie RSA) wegen multiplikationen statt exponentiationen
  - wenig Speicherplatz (für Chipkarten ein Kriterium)
  - Es gibt Standards
  - Patent-freie Algorithmen
- Man kann:
  - Signieren
  - Schlüsselaustauschen
  - Hybrid Verschlüsseln
  - (Wie RSA)
- Könnte RSA aufgrund der langen Schlüssel ablösen (vielleicht auch Quantenalgorithmen)
- Es wird addiert und verdoppelt statt multipliziert und potenziert (RSA)
- Anzahl der Punkte kann Atome im Weltall übertreffen
- Ist eine Einwegfunktion ohne Trapdoor
- Die Sicherheit liegt im diskreten Logarithmus Problem
- Diskreter Logarithmus: für P, Q bei $q = i \cdot P$ kann i nicht einfach berechnet werden
- Der Schnittpunkt des Koordinatensystems ist $P(0,0)$, $P(0,0)$ kann Inhalt von EC sein.
- Der unendlich Ferne Punk $\mathcal{O}$ ist das neutrale (null-)Element
- Die Gruppenordnung ist Anzahl der Punkte (somehow NICHT p)
- Für gleiche Sicherheit: Ein 3072 Bit RSA entspricht etwa einer 256 Bit EC.

4.3.1 Requirements

Hat immer die Form:

$$y^2 = x^3 + ax + b$$

und ist NICHT

$$y = f(x)$$

Nichtsingularitätsbedingung:

$$4a^3 + 27b^2 \not\equiv 0 \mod p$$

Diese muss erfüllt sein, ansonsten gibt es mehrere Wurzeln (Ergebnisse).
Und ist sie erfüllt, hat die Kurve sowas wie eine Spitze oder Überschneidungen.

Diese beiden Bedingungen stellen sicher, dass die Verbindung zwischen zwei Punkten genau einen weiteren Punkt schneidet. Bei Tangenten wird der Berührpunkt doppelt gezählt.

4.3.2 Punktaddition

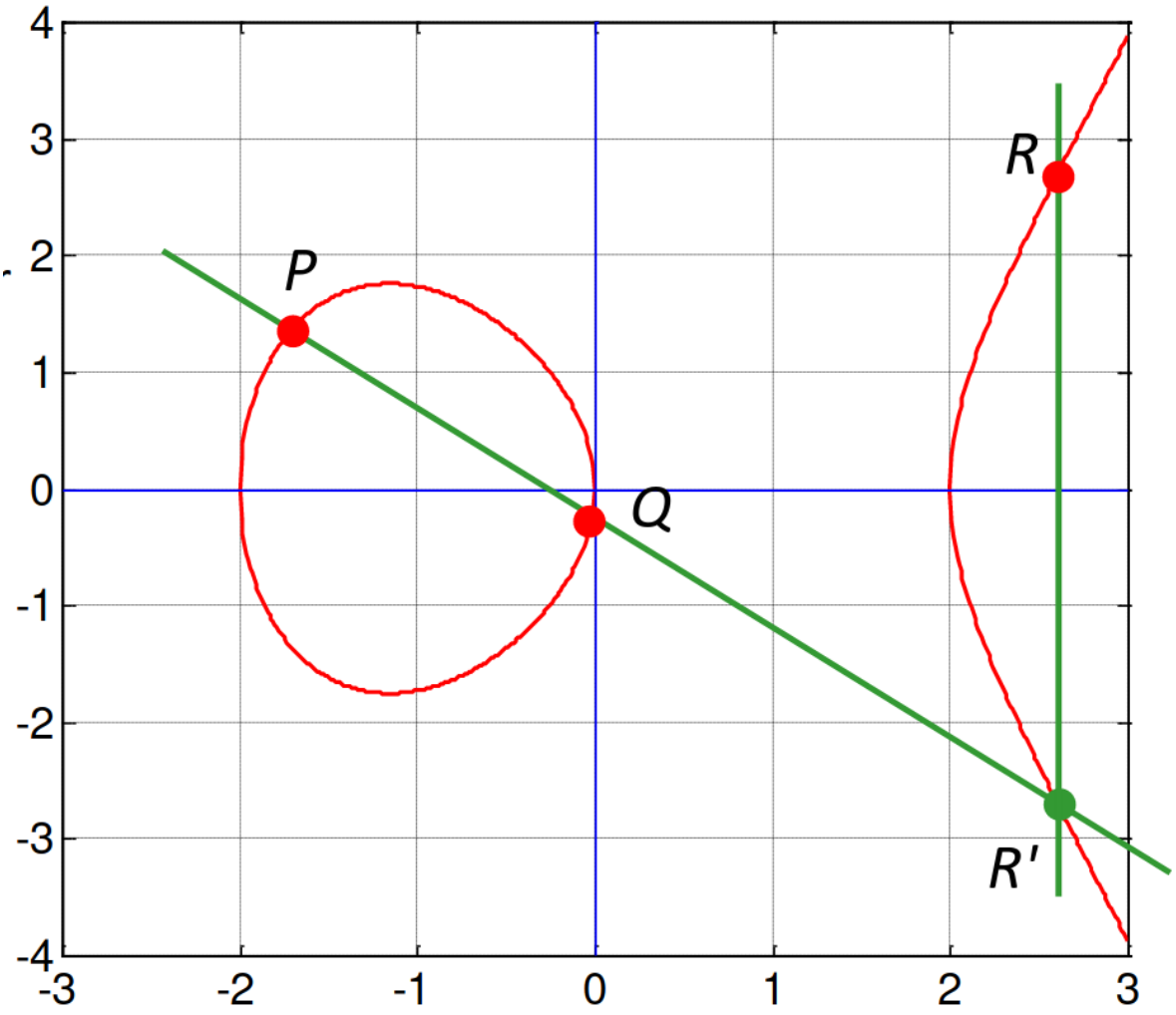

Abbildung 5: Elliptic Curve Punkt Addition

$$P + Q = R$$

4.3.3 Neutrales und Inverses Element

Inverses Element resp. Spiegelung an der x-Achse:

$$P'(x, -y) = -P(x, y)$$

Punktaddition:

$$P' + P = P + P' = \mathcal{O}$$

Das neutrale Element ist der Punkt im unendlichen, also

$$\mathcal{O}(x, \infty)$$

Dann einfach noch die einzelnen Koordinaten mit p modulo rechnen.

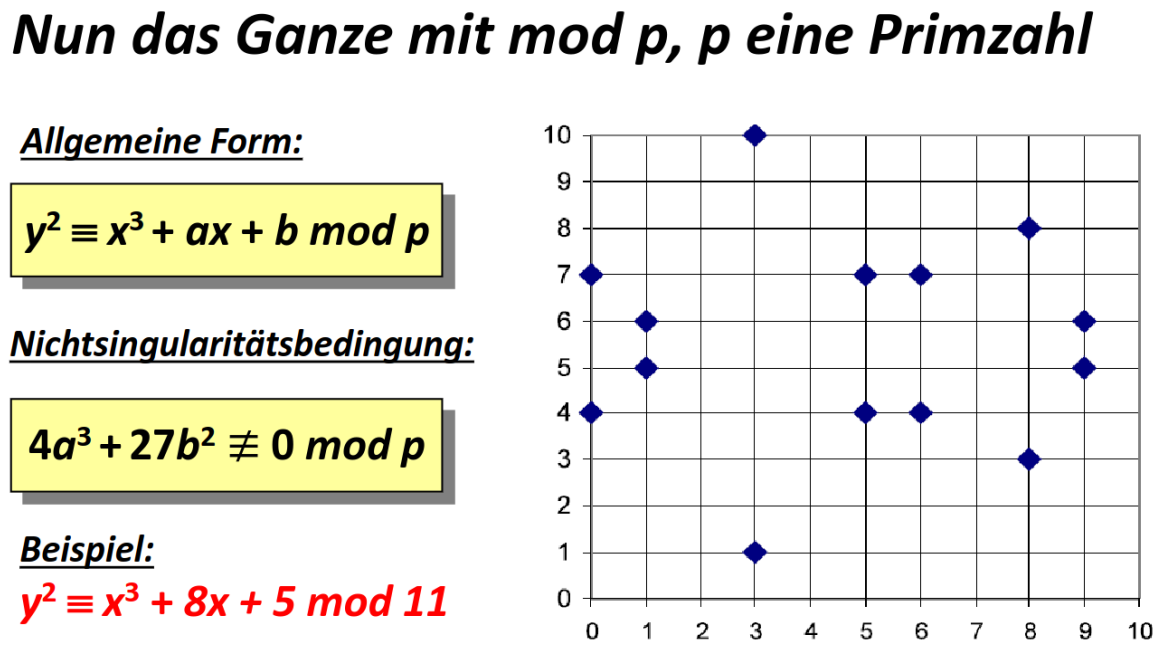

Abbildung 6: Elliptic Curve with Modulo

Ein Punkt P mit welchem man alle Punkte ausrechnen kann ist ein Basispunkt. Hier als Beispiel $(k\cdot)P(0, 4)$. Wenn wir P mit sich selbst addieren oder den Faktor k vorndran 1,2,3 ... ausrechnen, erhalten wir alle Punkte der Kurve.

Die Koordinaten muss man dabei jweils mit p modulo rechnen.

Das k vor dem P muss dan module die Gruppenordnung (wie viele verschiedene Punkte dass es gibt) gerechnet werden. Da sich die Punkte wiederholen.

4.3.4 Allgemeine Form & Zusammenfassung:

Remember:

- allgemeine Form: $E = \{(x; y)|y^2 = x^3 + ax + b \mod p\}$
- Nichtsingularitätsbedingung: $4a^3 + 27b^2 \neq 0$
- E = Zusammenfassung aller Punkte, welche die Bedingung erfüllen
- p muss zwischen 256 und 512 Bit sein
- a & b können beliebig sein, müssen aber die beiden Gesetze erfüllen
- Eine Gruppe heist zyklisch wenn es einen erzeugenden Punkt P gibt.
- Mit einen eurzeugenden Punkt P kann man alle anderen Punkte erzeugen $k \cdot P$
- ist die Gruppenordnung prim $rightarrow$ alle Elemente ausser $\mathcal{O}$ sind erzeugend

Bereich der Punkte kann berechnet werden mit:

$$p + 1 - 2\sqrt{p} \leq |E| \leq p + 1 + 2\sqrt{p}$$

Die effektive Range wird durch a und b bestimmt, mit p kann nur die Randzahlen berechnet werden.

4.3.5 Formel zur Addition von 2 Punkten

Zuerst Steigung berechnen, danach können x und y berechnet werden.

$$\begin{aligned} s &\equiv \frac{y_2 - y_1}{x_2 - x_1} \mod p \\ x_3 &\equiv s^2 - x_1 - x_2 \mod p \\ y_3 &\equiv s(x_1 - x_3) - y_1 \mod p \end{aligned}$$

Falls man P + P (Verdoppelung) rechnen muss, resp wenn $x_1 = x_2; y_1 = y_2$ dann ändert sich die Steigungsformel zu:

$$s \equiv \frac{3 \cdot x_1^2 + a}{2 \cdot y_1} \mod p$$

Remember: bei $x \cdot y \mod p \equiv (x \mod p \cdot y \mod p) \mod p$
Beispiel mit $P(0;4)$ und $Q(3;1)$ in Kurve $y^2 \equiv x^3 + 8x + 5 \mod 11$:

$$\begin{aligned}
s &\equiv \frac{1-4}{3-0} \mod 11 \\
&\equiv \frac{-3}{3} \mod 11 \\
&\equiv (-3) \cdot 3^{-1} \mod 11 \\
&\equiv [(-3) \mod 11 \cdot 3^{-1} \mod 11] \mod 11 \\
&\equiv 8 \cdot 4 \mod 11 \\
&\equiv 32 \mod 11 = 10
\end{aligned}$$

Danach Punke:

$$\begin{aligned}
x_3 &\equiv 10^2 - 0 - 3 \mod 11 = 9 \\
y_3 &\equiv 10 \cdot (0 - 9) - 4 \mod 11 = 5
\end{aligned}$$

4.3.6 Double and add Algorithmus

Analog kann zum Square and Multiply Algorithmus effizient gerechnet werden.

Beispiel: $28 \cdot P$

- $28 = 16 + 8 + 4 = 2^4 + 2^3 + 2^2 = 11100$
- Mit der ersten 1 macht man nichts
- wegen 1: p $\rightarrow$ 2p $\rightarrow$ 2p + p = 3p | double and add
- wegen 1: 3p $\rightarrow$ 6p $\rightarrow$ 6p + p = 7p | double and add
- wegen 0: 7p $\rightarrow$ 14p | double
- wegen 0: 14p $\rightarrow$ 28p | double

4.3.7 Bestimmung aller Punkte

A) Bestimmung aller Punkte

x	0	1	2	3	4	5	6
$x^3 + 3x + 2 \bmod 7$	2	5	2	3	1	2	5

y	0	1	2	3	4	5	6
$y^2 \bmod 7$	0	1	4	2	2	4	1

Abbildung 7: Bestimmung aller Punkte

Für x = 0: $(0; 3)$ und $(0; 4)$
Für x = 1: keine da $y^2 \mod 7 \neq 5$
Für x = 2: $(2; 3)$ und $(2; 4)$
Für x = 3: keine
Für x = 4: $(4; 1)$ und $(4; 6)$
Für x = 5: $(5; 3)$ und $(5; 4)$
Für x = 6: keine

4.3.8 Bestimmung ob Punkt auf Graf ist

Beispiel:

Kurve: $E : y^2 \equiv x^3 + x + 1$ über $\mathbb{Z}_{19}$ Punkt : $Q(15, 13)$

1. Punkt in Gleichung einsetzen
2. Links von Gleichung ausrechnen mod p
3. Rechts von Gleichung ausrechnen mod p
4. Vergleichen

$$
\begin{aligned}
y^2 &\equiv x^3 + x + 1 && \text{mod } 19 && && (1)\\
16^2 &\equiv 15^3 + 15 + 1 && \text{mod } 19 && && (2)\\
16^2 &\equiv 256 && \text{mod } 19 \equiv 9 && \text{mod } 19 \text{ und} && (3)\\
15^3 + 15 + 1 &\equiv 3391 && \text{mod } 19 \equiv 9 && \text{mod } 19 && (4)
\end{aligned}
$$

Da beide Seiten gleich sind, ist der Punkt auf der Kurve.

4.4 RSA

RSA braucht mehr Rechenleistung (ist sicherer?).

RSA Keys sind grösser als ECC Keys für gleiche Sicherheit.

Ablauf beim Signieren:

1. Hashwert berechnen (Integrität)
2. Hashwert mit privatem Schlüssel signieren (Authentizität)

4.4.1 Anzahl der Primzahlen 1-x

Ist eine Annäherungsformel, berechnet die Anzahl der Primzahlen in 1 bis x berechnen:

$$\pi(x) \approx \frac{x}{\ln(x)}$$

Nur ungerade Primzahlen von 1 bis x:

$$APZ(x) = \frac{Anz.PZ}{Anz.ungeradeZ.} \approx \frac{\frac{x}{ln(x)}}{\frac{x}{2}} = \frac{2}{ln(x)}$$

Beispiel:

$$\pi(10^{150}) \approx \frac{10^{150}}{\ln(10^{150})} = \frac{10^{150}}{150 \cdot ln(10)} \approx \frac{10^{150}}{345} \approx \frac{10^{150}}{3.5 \cdot 10^2} = \frac{10 \cdot 10^{149}}{3.5 \cdot 10^2} = \frac{10 \cdot 10^{147}}{3.5} \approx 2.9 \cdot 10^{147}$$

$$APZ(10^{150}) \approx \frac{2}{ln(10^{150})} = \frac{2}{150 \cdot ln(10)} \approx \frac{2}{345} = 0.0058 = 0.58\% = 5.8‰$$

4.4.2 Anzahl n-stelligen Primzahlen

Ist die genaue Formel, nicht unbedingt nötig, Annäherungsformel ist genau genug.

Anzahl der n-stelligen Primzahlen:

$$\pi(10^n) - \pi(10^{n-1})$$

Anteil der n-stelligen PZ. in den n-stelligen ungeraden Zahlen:

$$APZ(n) = \frac{Anz.PZ}{Anz.ung.Z.} \approx \frac{\pi(10^n) - \pi(10^{n-1})}{\frac{0.9 \cdot 10^n}{2}} = \frac{\pi(10^n) - \pi(10^{n-1})}{0.45 \cdot 10^n}$$

Beispiel:
$n = 150 \Rightarrow x = 10^{150}$

$$\pi(10^{150}) - \pi(10^{149}) = \frac{10^{150}}{ln(10^{150})} - \frac{10^{149}}{ln(10^{149})} \approx 2.6 \cdot 10^{147}$$

4.4.3 Dezimalstellen ausrechnen:

Bits: $N = 1024$
$D = N \cdot \log_2(10)$
Dezimalstellen: $10^D \approx 10^{308}$

4.5 Diskreter Logarithmus bei ECC

Diffie-Hellman: bei $z \equiv g^x \mod p$ ist auf x schwer zu schliessen.
Sofern p minestend 600 stellige Primzahl und g ein Generator ist.

Bei EC ist gleichung $Q = k \cdot P$ schwerz auf k zu schliessen.

4.5.1 Key exchange

Diffie-Hellman (K = Secret):

$$\begin{aligned} A &= g^a \mod p \\ B &= g^b \mod p \\ K &= A^b = B^a = g^{ab} \mod p \end{aligned}$$

EC Key exchange (K = Secret):

$$\begin{aligned} A &= a_{Alice} \cdot P = a_A \cdot P \\ B &= b_{Bob} \cdot P = b_B \cdot P \\ K &= b_B \cdot A = (b_B \cdot a_A) \cdot P = a_A \cdot B = (a_A \cdot b_B) \cdot P \end{aligned}$$

4.6 Verschlüsselung mit EC nach Volker Müller

1. Schlüsselgenerierung (Bob ist Empfänger)
   a) Elliptische Kurve wählen
   b) Mit Primzahl ca. 256, 384 oder 512 Bit gross
   c) Koeffizienten a und b wählen
   d) Einem Punkt P der eine zyklische Untergruppe der Primordnung q erzeugt.
   e) d wählen wobei $0 < d < q$ und 512 Bit lang
   f) Berechne $Q = d \cdot P$
   g) Damit sind die Beiden Schlüssel erzeugt:

      $K_{pub} = (p, a, b, q, P, Q)$

      $K_{priv} = d$

2. Verschlüsselung (Sender Alice)
   a) Wähle i mit $1 < i < q - 1$
   b) Berechne Einmal-Key $K_E = i \cdot P$
   c) Berechne Masking-Key $K_M = i \cdot Q$
   d) Verschlüssle Meldung T mit $Y = T \oplus x$-Wert von K_M
   e) Meldung mit Einmal Key schicken also (Y, K_E)

3. Entschlüsselung (Empfänger Bob)
   a) Masking-Key berechnen $K_M = d \cdot K_E$
   b) Meldung entschlüsseln $T = Y \oplus x$-Wert von K_M

5 Blinde Signaturen

Generelle Beschreibung: Anna weis nicht WAS sie unterschreibt, wenn sie das Dokument später sieht, weis sie aber DASS sie es unterschrieben hat.
Nutzen:

- Unverfällschbarkeit - kein falsches Geld erzeugen
- Anonymität - gegenüber Bank
- Unlinkbarkeit - keinen Zusammenhang zwischen s und m

Beispiel-Ablauf:

1. Kunde zieht geld ab m ist Identifikationsnummer
2. Kunde bildet m' und fragt Bank um Unterschrift
3. Bank rechnet s' aus m' und bucht Geld ab und schickt s' zurück (Signatur)
4. Kunde berechnet s aus s' für den Wert m
5. Kunde bezahlt im Shop mit m und s
6. Shop kann mit Public-Key und Signatur kontrollieren ob von Bank signiert
7. Shop liefert Geld mit ID m und Signatur s bei Bank ein.
8. Bank prüft ebenfalls Signatur.
9. Bank weis nun, dass dieses Geld abgehoben wurde, aber nicht von wem.

Allgemeiner mathematischer Ablauf:

1. Kunde kenn Public-Key N und e
2. Bank kenn Private-Key d
3. Kunde: Wahl der Nachricht $m \rightarrow$ eindeutige Seriennummer
4. Kunde: Nachricht "blinden":
   a) Zufällige wahl von $r \in_R \mathbb{Z}_N^*$
   b) $N = p \cdot q$ mit p, q Primzahlen
   c) $ggT(r, N) = 1$
   d) $ggT(r, N) \neq p$
   e) $ggT(r, N) \neq q$
   f) Berechnung von $m' \equiv r^e \cdot m \mod N$
5. Kunde: "geblindete"Nachricht m' schicken
6. Bank: Nachricht m' signieren:
   $s' \equiv (m')^d \mod N$
7. Bank: Signatur s' schicken
8. Kunde: Berechnung der Signatur s:
   $s \equiv s' \cdot r^{-1} \mod N$
9. Kunde: Überprüfen der Signatur:
   $m \equiv s^e \mod N \rightarrow$ ok?
10. Kunde: Bezahlen mit m und s
11. Shop und Bank überprüfen Echtheit der Münze nach Schritt 7.

5.1 Beweis der Korrektheit

$$\frac{s'}{r} \equiv \frac{(m')^d}{r} \equiv \frac{(r^e \cdot m)^d}{r} \equiv \frac{r^{ed} \cdot m^d}{r} \equiv \frac{r \cdot m^d}{r} \equiv m^d \equiv s \mod N$$

5.2 Beispiel mit Zahlen

Parameter:

- Öffentlicher Schlüssel: $(N, e) = (91, 5)$
- Privater Schlüssel: $(p, q, d) = (7, 13, 29)$

Kontrolle der Paramter:
$\varphi(N) = \varphi(7 \cdot 13) = \varphi(7) \cdot \varphi(13) = 6 \cdot 12 = 72 = (2 \cdot 2 \cdot 2 \cdot 3 \cdot 3)$
$e = 5$ ist teilerfremd zu 72.
$d = 5^{-1} \mod 72 \equiv 29$, da $5 \cdot 29 \equiv 145 \equiv 2 \cdot 72 + 1 \equiv 1 \mod 72$

Sei $m = 11$ die Nachricht.
Wir wählen zufällig $r = 16$
es muss gelten: $ggT(N, r) = ggT(91, 16) = 1$

Erwartete Signatur:
$s \equiv m^d \mod N \equiv 11^{29} \mod 91 = 72$

Kunde kennt Public Key (667, 9)	Meldung	Bank kennt Secret Exponent d = 137
1. Wahl der Nachricht m: m = 38		
2. Nachricht m „blinden": r = 63 $m' \equiv 63^9 \cdot 38 \; mod \; 667 \equiv 36$		
3. geblindete Nachricht m' schicken:	m' = 36 ---------->	
		4. Nachricht m' signieren: $s' \equiv (36)^{137} \; mod \; 667 \equiv 487$
	s' = 487 <----------	5. Signatur s' zurückschicken:
6. Signatur s aus s' extrahieren: $s \equiv 487 \cdot 63^{-1} \equiv 487 \cdot 180 \equiv 283 \; mod \; 667$		
7. Signatur s prüfen: $\underbrace{m \equiv s^e \equiv 283^9 \equiv 38 \; mod \; 667}_{OK}$		
8. Mit Münze (11, 72) bezahlen:		
9. resp. 10 Signatur s prüfen: Shop wie Bank überprüfen die Echtheit der Münze, cf. Schritt 7.		

Abbildung 8: Blinde Signaturen

5.3 Basis-Test

- geheimer Exponent muss aufgeteilt werden
- Man kann ihn additiv oder multiplikativ aufteilen
- Kann beliebig additiv aufgeteilt werden
- Kann NICHT beliebig multiplikativ aufgeteilt werden:
  Der erste Teil der Aufteilung muss teilerfremd zu $\varphi(N)$ sein, also $ggT(d_1; \varphi(N)) = 1$.
- Ablauf der Erstellung einer Doppelsignatur ist bei additiv und multiplikativ nicht dieselbe
  Bei multiplikativ muss z.B. die Reihenfolge eingehalten werden.
- Blinde Signaturen werden z.B. für anonymes, digitales Geld verwendet.

6 Einführung in die Public-Key Infrastruktur (PKI)

6.1 Verschlüsseln und Signieren (repetition)

6.1.1 Verschlüsseln

6.1.2 Signieren

Ablauf signieren:

1. Dokument von Alice ist Ausgangswert
2. Hash berechnen → Hashwert
3. chiffrieren (mit private key und Hash) → Signatur
4. Dokument & Signatur + Zertifikat → signiertes Dokument

Achtung: Schlüsselverwendung bei Signatur ist umgekehrt wie bei Verschlüsselung. Private Key wird für die Erstellung der Signatur verwendet und Public-Key um zu validieren.

Warum Zertifikat? → um sicherzustellen, dass der öffentliche Schlüssel auch wirklich von Alice ist.

Ablauf Signatur prüfen:

1. Dokument von Alice ist Ausgangswert
2. Dokument entpacken (Signatur und Dokument)
3. Signatur mit öffentlichem Schlüssel entschlüsseln → Hashwert
4. Hashwert von Dokument berechnen
5. Hashes vergleichen
6. Zertifikat Überprüfen
7. Wenn alles ok dann ist Signatur gültig.

6.2 Zertifikate

6.2.1 Herstellung eines Zertifikats

Allgemeiner Ablauf:

- Antragssteller identifiziert sich bei CA
- Aus dessen Informationen wird Datensatz gebildet (Zertifikatsinhalt)
- Datensatz wird mit privatem Schlüssel von CA signiert → Zertifikat
- CA veröffentlicht Zertifikat

Technisches vorgehen:

1. Zertifikatsinhalt
   - Version
   - Serial Number
   - Subject
   - Public Key
2. Inhalt hashen
3. Hash signieren
4. Signitierter Hash + Zertifikatsinhalt → Zertifikat

6.2.2 Installation eines neues (Root-)Zertifikates

1. Root-CA-Zertifikat Echtheit überprüfen
2. Echtheitsprüfung wird mit Fingerprint gemacht
3. Lokal angezeigter Fingerprint wird mit vertrauenswürdiger Referenzquelle verglichen

6.2.3 Überprüfung der Echtheit eines Zertifikates vom Betriebssystem

1. Applikation überprüft Signatur auf Zertifikat mithilfe des Root-CA-Zertifikates.
2. Zertifizierungsstelle muss im System hinterlegt sein; sie wird zum Trust Anchor

6.2.4 Zertifikatsklassen

- **Klasse 1:** wenig Sicherheit, keine Identitätsprüfung
- **Klasse 2:** mittlere Sicherheit, schwache Identitätsprüfung
- **Klasse 3:** hohe Sicherheit, strenge Identitätsprüfung
- **Qualified Certificate:** höchste Stufe, werden nur für natürliche Personen ausgestellt

7 Protokolle

7.1 User Authentication

- Username / Password
- One-Time Password
- Symmetric Algorithms
- Public-Key Algorithms
- Biometric Authentication

7.2 False-rates

Es gibt zwei Arten von False-rates:

- False Acceptance Rate (FAR)
- False Rejection Rate (FRR)

Beide sollten so tief wie mögliche sein. Aber es ist ein Tradeoff zwischen den beiden. Senkt man die Eine erhöht sich die Andere.

7.3 Verifikationen

7.3.1 One to many

Handy: überprüfe ob ich derjenige bin, der ich vorzugeben behaupte.

7.3.2 Many to one

Bank: überprüfe ob ich derjenige bin, der ich vorzugeben behaupte.

7.4 Paralellsession Attacke

Zwei Sessions eröffnen, dann muss nichts gerechnet werden und Zufalls-/Chiffrierzahl kann kopiert werden.

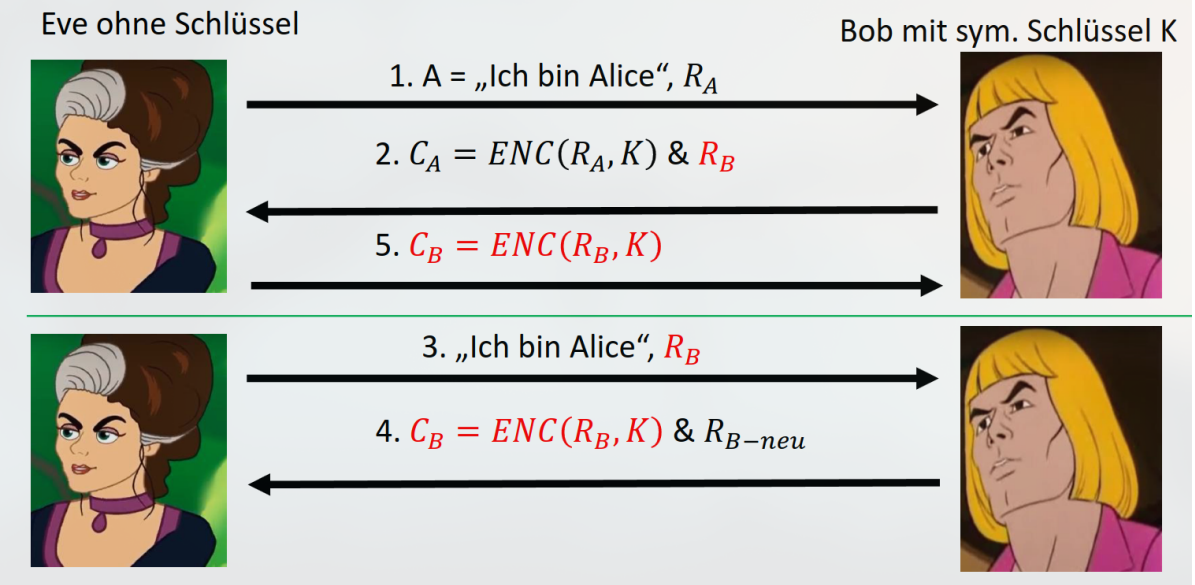

Abbildung 9: Paralellsession Attacke

8 Quantenkryptographie

Quantenkryptographie und Quantencomputer sind zwei verschiedene Dinge.

8.1 Polarization

4 Zustände, jedem muss 1 oder 0 zugewiesen werden.

- Senkrecht
- Wagrecht
- Schräg links
- Schräg rechts

Darauf basierend können Filter kreiert werden. Sollte filter nicht auf Teilchen passen, ist es 50/50 in welchem Zustand es durchgelassen wird.

8.2 Quantum Key Exchange

1. Sender Alice wählt zufällige Bits
2. Alice sendet Photonen mit gewählter Polarization
3. Empfänger Bob wählt zufällige Filter
4. Bob erhlält gefilterte Photonen
5. Bob kennt die Kodierung und ordnet 1 oder 0 zu
6. Bob sendet zurück, welche Filter er benutzt hat
7. Alice sagt, welche Filter korrekt waren

Im statistischen Mittle werden in 50% die falschen Filter gewählt. Zudem werden die Hälfte davon werden aufgedeckt um zu prüfen ob man abgehört wurde. Es müssen also ca. 4 mal so viele geschickt werden.

Key-takeaway: immer die Bits verwenden, welche durch den Filter nicht verändert werden. Alle anderen haben einen nicht vorhersagbaren Zustand.