IT-Sicherheit

Hochschule Osnabrück
University of Applied Sciences

Das wichtigste über Kryptologie

… mit ein wenig diskreter Mathematik

Prof. Dr.-Ing. Alfred Scheerhorn

a.scheerhorn@hs-osnabrueck.de

Kryptologie

Hochschule Osnabrück
University of Applied Sciences

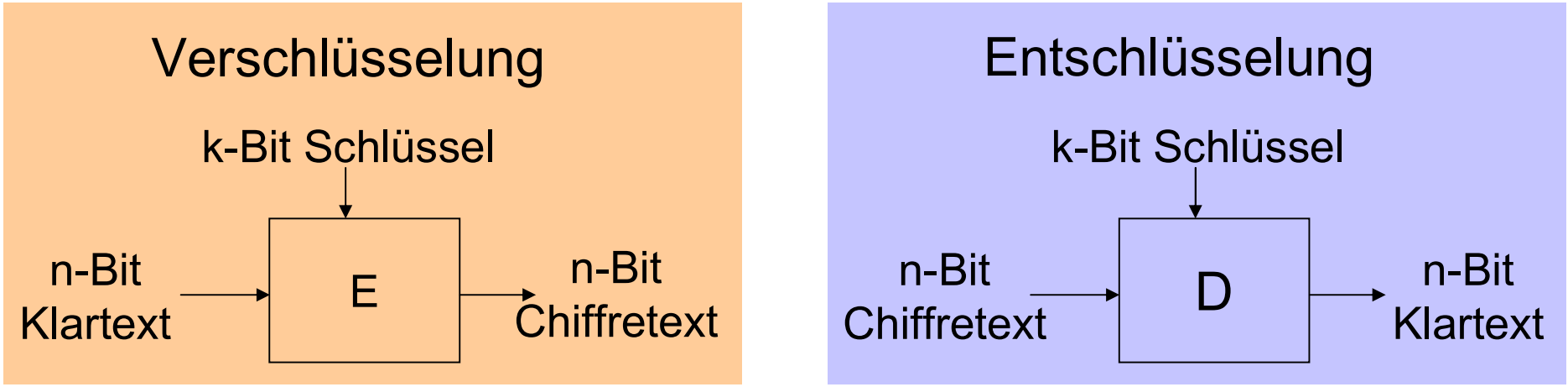
- Ursp: Lehre der Verschleierung von Nachrichten

- Krypto***graphie***: Entwicklung kryptographischer Verfahren
- Krypto***analyse***: Analyse / Brechen kryptographischer Verfahren

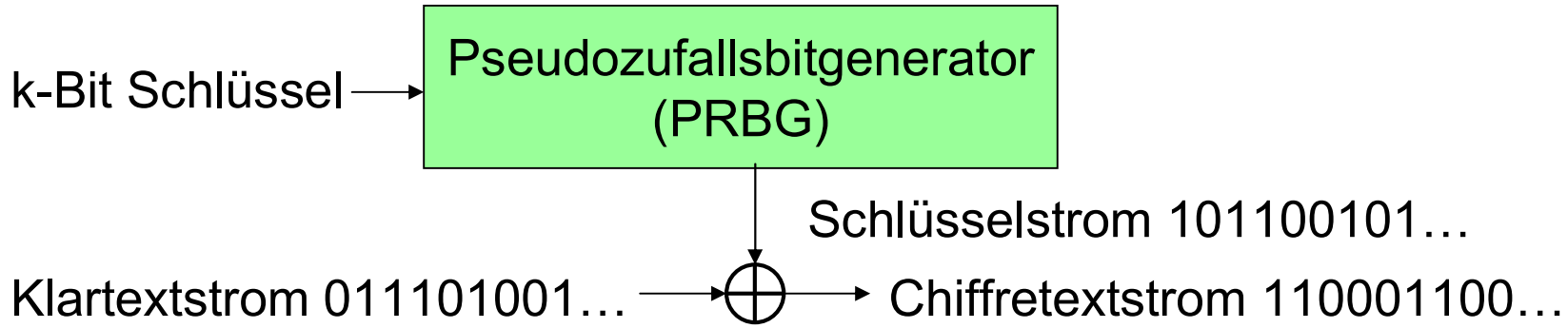
Arten von Chiffren

Hochschule Osnabrück
University of Applied Sciences

- **Blockchiffren**: Blocklänge n-Bit, Schlüssellänge k-Bit



- **Stromchiffren**: Zeichenweise Verschlüsselung (i.d.R. bitweise)



Kerckhoffsche Prinzip

Hochschule Osnabrück
University of Applied Sciences

- Brechen eines Kryptosystems
  - Berechnung des verwendeten geheimen Schlüssels oder
  - Rückrechnung von Klartexten aus Chiffretexten
- Komplexität:
  - Leicht / effizient berechenbar: In polynomialer Laufzeit in der Bitgröße n der Eingabelänge berechenbar: $O(n^k)$
  - Praktisch nicht berechenbar: Kein effizientes Berechnungsverfahren bekannt
    - Laufzeit i.d.R. exponentiell oder subexponentiell
- Kerckhoffsche Prinzip:

„Die Sicherheit eines Kryptoverfahrens soll nie von der Geheimhaltung des Verfahrens abhängen, sondern allein von der Geheimhaltung des verwendeten Schlüssels."

!

Angriffsklassen und –verfahren

Hochschule Osnabrück
University of Applied Sciences

Angriffsklassen: Wie viel weiss der Angreifer?

- **Ciphertext only Attack**: Angreifer kennt nur Chiffretext
- **Known Plaintext Attack**: Angreifer kennt einige vorgegebene Klartext-Chiffretext Paare zum aktuellen Schlüssel
- **Chosen Plaintext Angriff**: Angreifer kennt selbst gewählte Klartext-Chiffretext Paare zum aktuellen Schlüssel

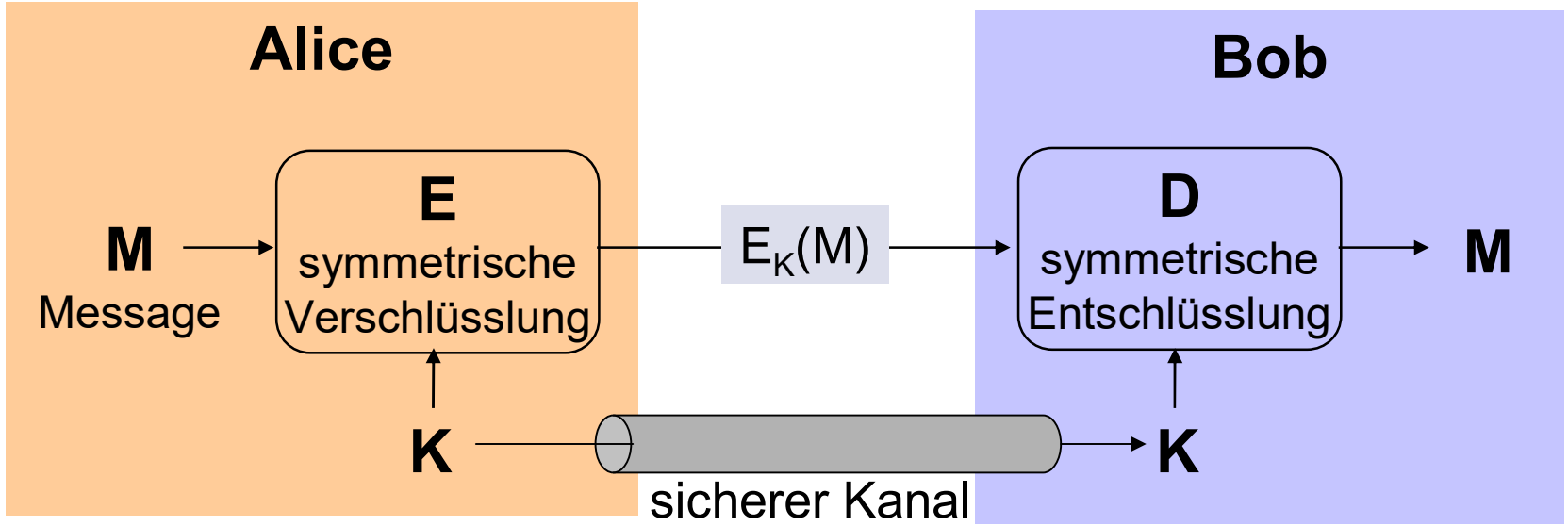
Angriffsverfahren

- Erschöpfende Schlüsselsuche (Brute Force Attack)
- Spezielle kryptoanalytische Angriffe
  - Häufig auf den jeweiligen Kryptoalgorithmus zugeschnitten
  - Z.B. Differenzielle- oder Lineare Kryptoanalyse

Symmetrische Verschlüsselung

Hochschule Osnabrück
University of Applied Sciences

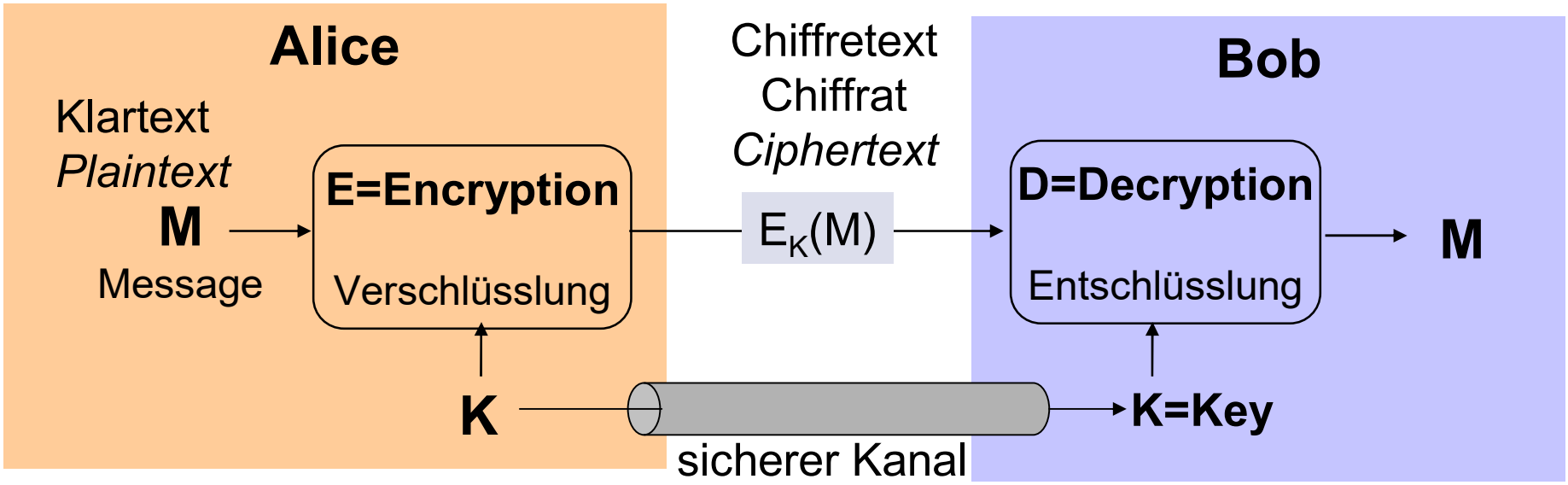
- Alice und Bob nutzen ***den selben geheimen*** Schlüssel **K**
- Schlüsselaustauschproblem:
  - Geheimer Schlüssel muss vor der Verschlüsselung vertraulich ausgehandelt / übertragen werden
- Verfahren: AES, 3DES, DES, …



Symmetrische Verschlüsselung

Hochschule Osnabrück
University of Applied Sciences

- Alice und Bob nutzen ***den selben geheimen*** Schlüssel **K**



Advanced Encryption Standard

Hochschule Osnabrück
University of Applied Sciences

- NIST iniitiert 1997 öffentliche Ausschreibung eines Algorithmus für den AES, u.a. mit folgenden Anforderungen
  - Sicherheit gegenüber bekannten Angriffen
  - Schlüssellängen 128 Bit und Blocklängen von 128, 192 und 256 Bit
  - Geringer Implementierungsaufwand und hohe Effizienz in Software und Hardware
  - Einfaches algorithmisches Design
  - lizenzkostenfreie Verfügbarkeit und Nutzung
- Nach mehreren Auswahlrunden wurde im Jahr 2000 Algorithmus Rijndael (J. Daemen und V. Rijmen) wurde als AES ausgewählt
- AES unterstützt Schlüssellängen von 128, 192 und 256 Bit

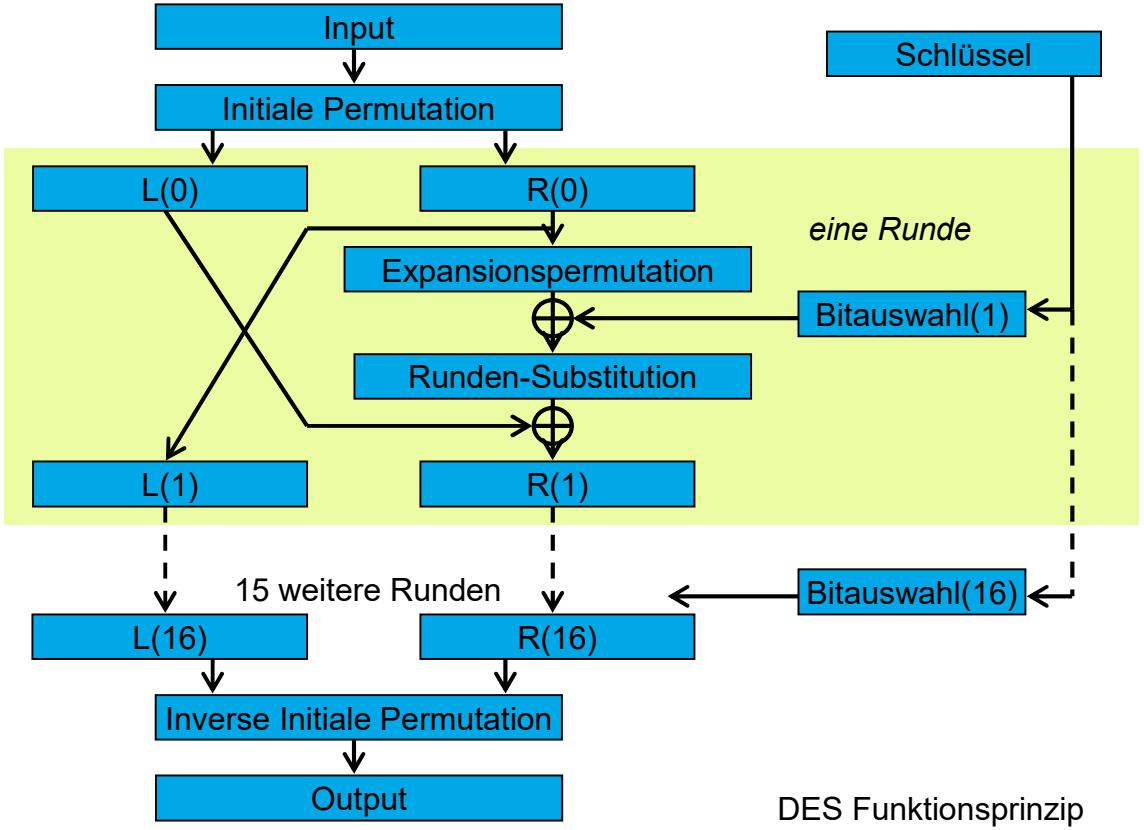
Data Encryption Standard

Hochschule Osnabrück
University of Applied Sciences

- 64 Bit Blocklänge, 56 Bit effektive Schlüssellänge
- In 70er Jahren von IBM für NSA entwickelt. Seit 1981 ANSI Standard
- Triple DES, 3DES. Varianten mit unterschiedlichen Schlüssellängen
  - 112 Bit:

    C=E(K1,D(K2,E(K1, P)))
  - 168 Bit:

    C=E(K1, D(K2, E(K3, P)))



DES Funktionsprinzip

Wie groß müssen Schlüssel sein?

Hochschule Osnabrück
University of Applied Sciences

Schlüssellänge	Anzahl Schlüssel	Zeit bei 10^6 Berechnungen pro Sek.	Zeit bei 10^{12} Berechnungen pro Sek.
32	$4.3 * 10^9$	35.8 Minuten	2.15 ms
40	$1.1 * 10^{12}$	6.3 Tage	0.5 Sek
56	$7,2 * 10^{16}$	1142 Jahre	10 Stunden
128	$3,4 * 10^{38}$	$5 * 10^{24}$ Jahre	$5 * 10^{18}$ Jahre

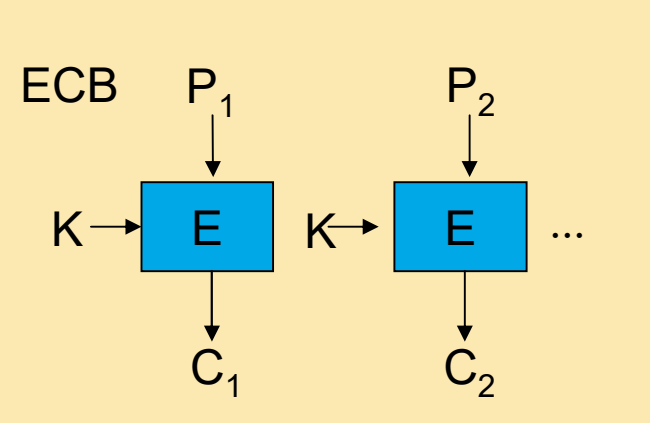
Größenordnungen	Anzahl	Bit
Sekunden pro Jahr	$3.3 * 10^7$	25
Sekunden seit Entstehung des Sonnensystems	$2.0 * 10^{17}$	57
Taktzyklen pro Jahr bei 4GHz	$1.3 * 10^{17}$	56
Anzahl 75-stelliger Primzahlen	$5.2 * 10^{72}$	241
Anzahl Elektronen im Universum	$8.4 * 10^{77}$	258
8 stellige Passwörter aus 26 Zeichen	$2.1 * 10^{11}$	37
12 stellige Passwörter aus 62 Zeichen	$3.2 * 10^{21}$	71

Betriebsarten von Blockchiffren (1)

Hochschule Osnabrück
University of Applied Sciences

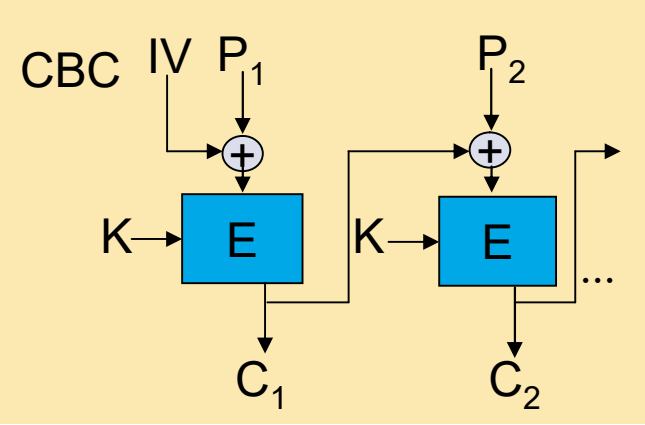
Electronic Code Book Mode (ECB)

- Jeder Block unabhängig verschlüsselt
- Nachteil: $P_i = P_j \rightarrow C_i = C_j$



Cipher Block Chaining (CBC)

- IV=Initialisierungsvektor
- IV wird im Klartext mitgeschickt
- Verschlüsselung:
  C_n von allen P_i, $i \leq n$ abhängig
- Entschlüsselung:
  P_n nur von C_n und C_{n-1} abhängig



Betriebsarten von Blockchiffren (2)

Hochschule Osnabrück
University of Applied Sciences

Output Feedback Mode (OFB)

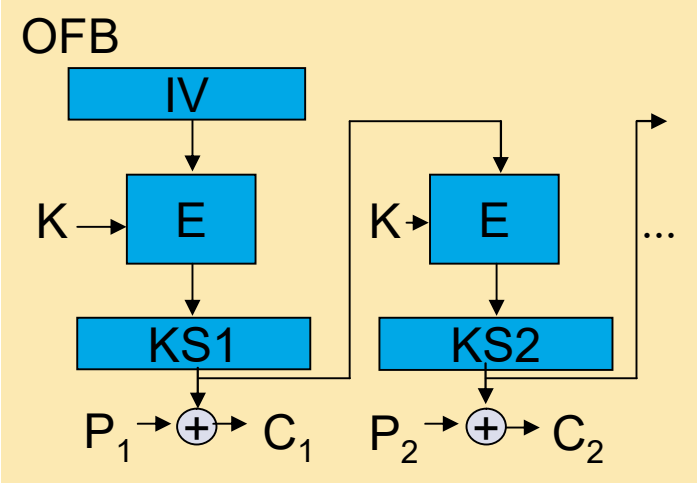
- Schlüsselstrom KS durch iterierte Verschlüsselung des IV

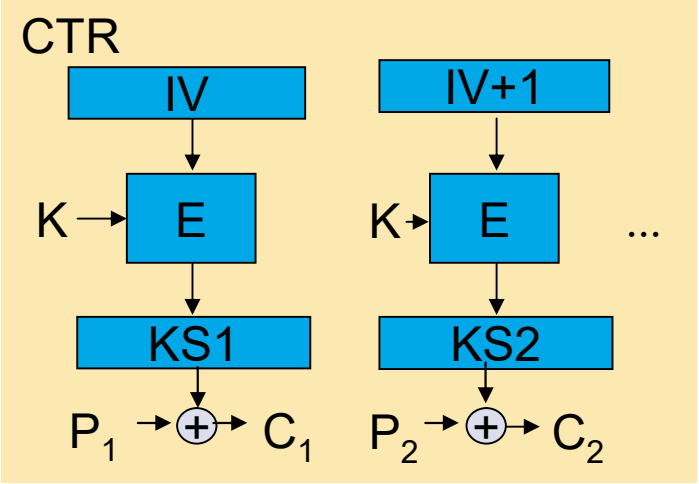
Counter Mode (CTR)

- Schlüsselstrom KS durch Verschlüsselung von IV, IV+1, IV+2, …

Einsatz (OFB, CTR)

- zur Generierung von Pseudozufallsbitfolgen
- als Stromchiffre





Beispiele für Stromchiffren

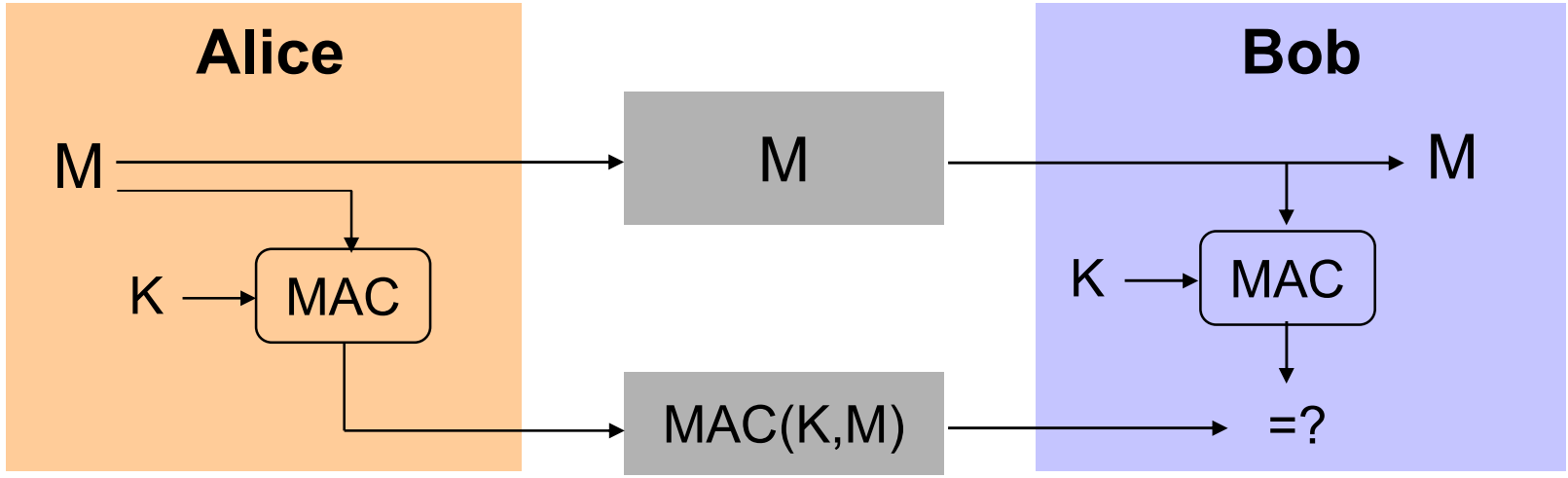
Hochschule Osnabrück
University of Applied Sciences

- Betrieb einer gegebenen Blockchiffre
  - im Output Feedback Mode (OFB-Mode)
  - im Counter Mode (CTR-Mode)
- RC4: Ron‘s Cipher4 (Ron Rivest, RSA)
  - nicht länger sicher
  - per RFC 7465 (Februar 2015) für TLS verboten
- ChaCha20 (RFC7439)

Message Authentication Codes

Hochschule Osnabrück
University of Applied Sciences

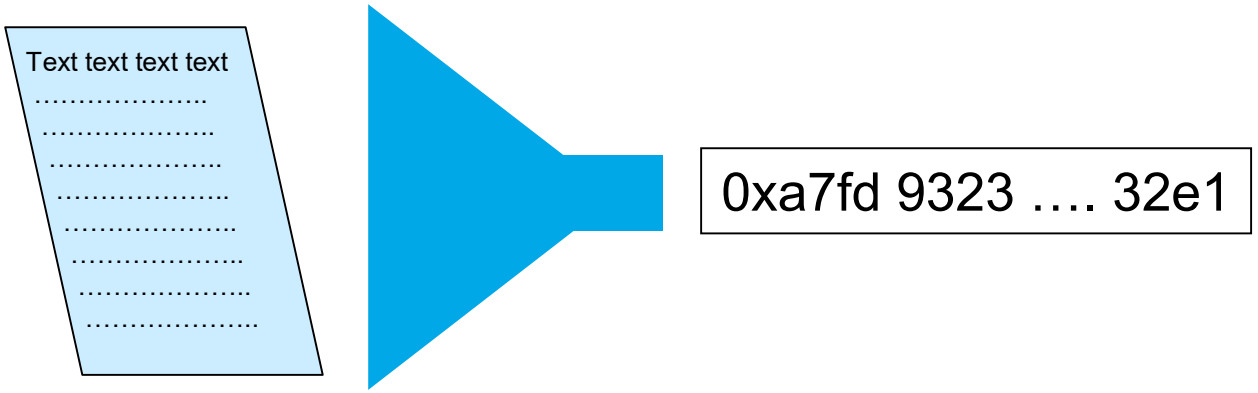
- Kryptographische Prüfsummen (symmetrisches Verfahren)
  - Funktionsweise s. Abbildung
- Sicherheitsanforderungen
  - Ohne K soll es einem Angreifer praktisch unmöglich sein, für eine Nachricht M eine gültige Prüfsumme zu berechnen.
  - Bei gegebener Nachricht M und Prüfsumme MAC(K,M) darf es einem Angreifer praktisch weder möglich sein, K zu berechnen, noch einen zweite Nachricht M' mit identischer Prüfsumme zu finden.



Hashfunktionen

Hochschule Osnabrück
University of Applied Sciences

- Eine Hashfunktion bildet Daten beliebiger Länge auf einen Hashwert einer festen Länge ab



Hashfunktionen

Hochschule Osnabrück
University of Applied Sciences

Eine Funktion $H:\{0,1\}^* \rightarrow \{0,1\}^n$ heißt kryptographisch starke Hashfunktion mit Hashwertlänge n, gdw.

1. Zu *x* ist *z*=H(*x*) leicht berechenbar. Zu gegebenen *z* ist es praktisch unmöglich, ein *x* mit H(*x*)=*z* zu finden. (H ist eine Einwegfunktion.)
2. Zu gegebenem *x* ist es praktisch unmöglich ein *y* zu finden, so dass H(*x*) = H(*y*). (schwache Kollisionsresistenz)
3. Es ist praktisch unmöglich *x* und *y* zu finden, so dass H(*x*) = H(*y*). (starke Kollisionsresistenz)

Beispiele für Hashfunktionen:

- MD-Familie: MD2, MD4, MD5 (inzwischen alle gebrochen)
- SHA-1 (SHA = Secure Hash Algorithm, 160 Bit Hashwert, gebrochen)
- SHA-256, SHA-384, SHA-512 (SHA-2 Familie)
- RIPE MD 160 / 256 / 320
- NIST: Keccak (2013, SHA-3 Familie) 224 / 256 / 384 / 512 Bit Hashwert

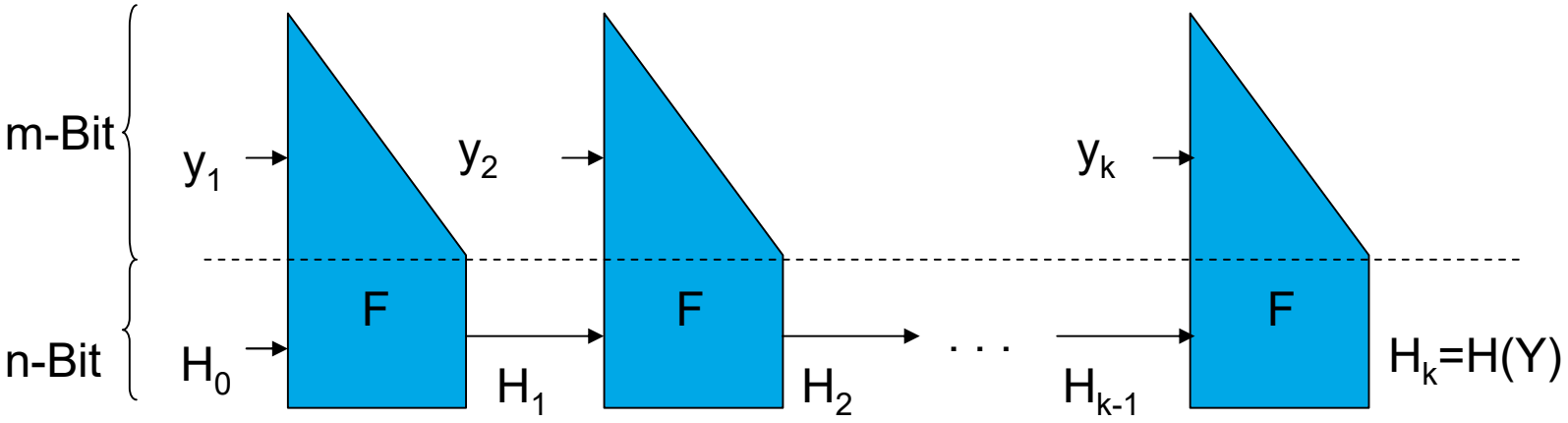
Konstruktion Hashfunktionen

Hochschule Osnabrück
University of Applied Sciences

- Der zu hashende Text Y wird in m-Bit Blöcke y_i, i=1,...,k unterteilt

$$Y=(y_1,y_2,...,y_k)$$ (ggf. Padding erforderlich)

- Der Hashwert von Y ergibt sich durch iterierte Anwendung einer Einwegfunktion $F:\{0,1\}^{m+n} \rightarrow \{0,1\}^n$
  - H_0 ist n-Bit Konstante



Birthday Paradox

Hochschule Osnabrück
University of Applied Sciences

- Sei H eine Hashfunktion mit Hashwertlänge *n*-Bit.
- Brute-Force Kollisions-Suche: Wähle x_i, i=0,1,2,… zufällig, berechne $h_i=H(x_i)$ und vergleiche h_i mit allen bisherigen h_j, j < i.
- Eine Kollision $h_i = h_j$ tritt bei n-Bit Hashwertlänge im Mittel auf bei

$$i = 1.17 \cdot SQRT(2^n) = 1.17 \cdot 2^{n/2}$$

⇒ Hashwertlänge so wählen, dass $1.17 \cdot 2^{n/2}$ zu groß für erschöpfende Suche ist.

⇒ Falls symmetrische Schlüssellänge von n-Bit sicher ist, sollten Hashfunktionen eine Hashwertlänge von 2n Bit aufweisen

- AES 128 Bit sicher => Hashwertlänge 256 Bit

Verfahren zur MAC-Berechnung

Hochschule Osnabrück
University of Applied Sciences

- CBC Verschlüsselung: Letzter Chiffretext-Block C_N hängt von allen Klartextblöcken ab und ist als MAC geeignet.

$$C_N = E_K(P_N \oplus E_K(P_{N-1} \oplus E_K(\ldots \oplus E_K(P_1) \ldots)))$$

- Hashfunktionen: Sei H eine Hashfunktion
  - H(K|M) ist grundsätzlich als MAC geeignet, weist jedoch bei üblichen Hashfunktionen Schwächen auf.
- Standardisiert: Hashed MAC (**HMAC**) (ursprünglich RFC 2104)

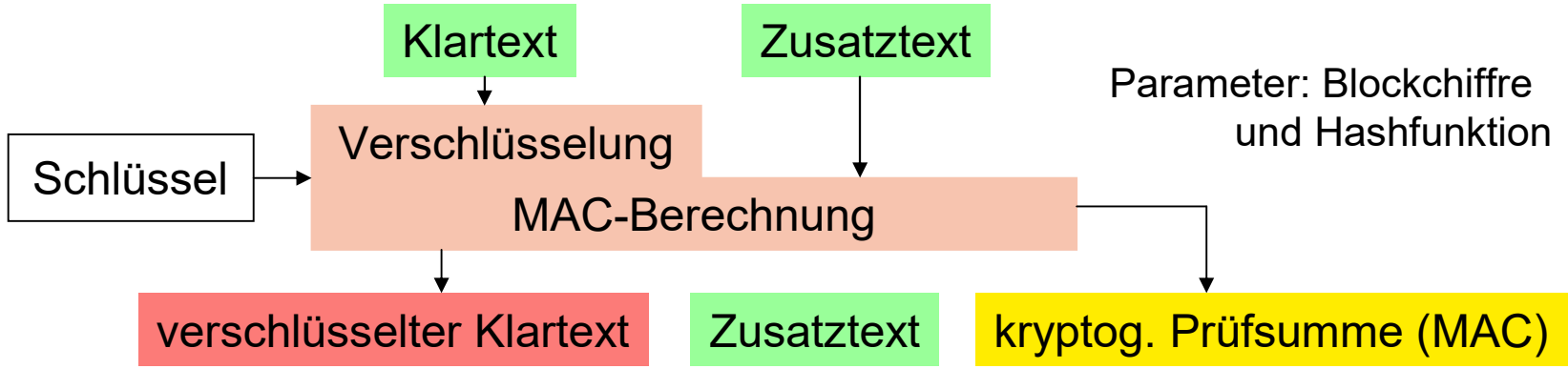
$$MAC(K, M) := H(K \oplus p1 \mid H(K \oplus p2 \mid M))$$

  - p1 und p2 sind feste Füllmuster (Padding), um die Schlüssel auf die Länge eines Blocks der Hashfunktion aufzufüllen
  - Hashfunktion ist mit anzugeben
- RFC 8439: POLY1305 MAC
  - Polynom durch Textblöcke bestimmt (P_1,…P_N) wird in gegeben Wert r modulo einer gegebenen Primzahl p ausgewertet

Authenticated Encryption

Hochschule Osnabrück
University of Applied Sciences

- „Zusammenschaltung“ von Verschlüsselung und Nachrichtenauthentisierung
  - aufgrund von Schwächen in getrennter Ausführung (MAC-then-Encrypt)
- Betriebsarten für Authenticated Encryption with associated Data (AEAD)
  - Kombination von Verschlüsselung und gleichzeitiger Berechnung eines Message Authentication Codes (MAC)
  - Einsatz in IPSec, TLS, SSH
- Bsp.1/2: GCM – Galois Counter Mode, CCM – Counter with CBC-MAC
  - Klartext wird verschlüsselt, Klartext und Zusatztext werden authentisiert

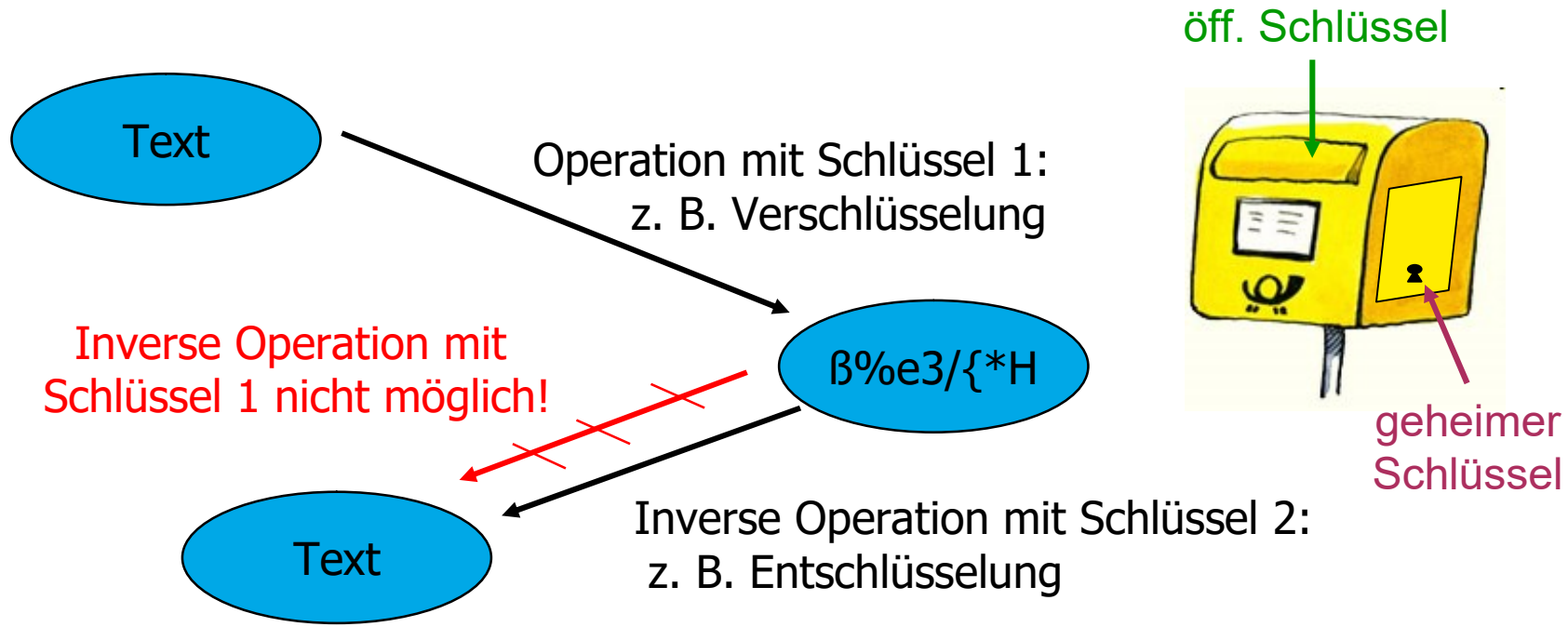


- Bsp.3: Kombination von Stromchiffre ChaCha20 mit MAC POLY1305

Asymmetrische Kryptographie

Hochschule Osnabrück
University of Applied Sciences

- Engl.: **public key cryptography**
- Jeder Teilnehmer hat ein Schlüsselpaar: (Schlüssel 1, Schlüssel 2)
- Ein Schlüssel ist aus dem anderen praktisch nicht berechenbar



Einsatz: Asymmetrische Kryptographie

Hochschule Osnabrück
University of Applied Sciences

Asymmetrische Verschlüsselung

- Alice verschlüsselt Nachricht M mit Bobs öffentlichen Schlüssel
- (Nur) Bob kann M mit seinem privaten Schlüssel entschlüsseln

Digitale Signatur

- Alice signiert einer Nachricht mit ihrem privaten Schlüssel
- Bob prüft Signatur von Alice mit öffentlichen Schlüssel von Alice

Schlüsselaustausch

- Alice und Bob senden sich gegenseitig öffentliche Schlüsselanteile
- Beide können damit einen gemeinsamen geheimen Schlüssel berechnen

Gruppen mit endlich vielen Elementen

Hochschule Osnabrück
University of Applied Sciences

- Gruppe ≈ „Operationen können rückgängig gemacht werden“ und es gibt ein neutrales Element.
- Bsp: Ganze Zahlen mit Addition; Rationale Zahlen mit Multiplikation
- $Z_m := \{0,1, \ldots, m\text{-}1\}$ ist die Menge der Zahlen von 0 bis m-1 ($m \geq 2$)
- Addition, Subtraktion und Multiplikation wird modulo m gerechnet
  - Wenn Ergebnis größer, gleich oder negativ, so oft m addieren bzw. subtrahieren, bis Ergebnis in 0… m-1

Aufgabe:
Bitte ausfüllen

Addition in Z_5

+	0	1	2	3	4
0					
1					
2					
3					
4					

Multiplikation in Z_5 (ohne 0)

·	1	2	3	4
1				
2				
3				
4				

Rechnen in Z_m

Hochschule Osnabrück
University of Applied Sciences

- Beim Rechnen (+, -, ·) in $Z_m = \{0,1, \ldots, m-1\}$ darf man jederzeit Zwischenergebnisse modulo *m* reduzieren, ohne das sich das Ergebnis ändert.
  - das erleichtert Rechnungen
- Testen:

$15 + 22 \bmod 7$ $\quad$ $(5+9) \cdot 9 \bmod 13$ $\quad$ $15 \cdot 32 \bmod 17$

Ausrechnen und reduzieren:

Zwischendurch reduzieren.

Multiplikative Inverse in Z_m

Hochschule Osnabrück
University of Applied Sciences

- Voraussetzung für Inverses *(ggT=größter gemeinsamer Teiler)*
  - *a* aus Z_m := {0,1, …, *m*-1} ist multiplikativ invertierbar ⇔ ggT(*a*,*m*) = 1
- Effiziente Berechnung über erweiterten Euklidischen Algorithmus
- Berechnung im Kopf:
  - Suche *b*, so dass $a \cdot b = x \cdot m + 1$

  (Falls $a \cdot b = x \cdot m - 1$, dann ist $(-b) \bmod m = (m - b)$ Inverses von *a*)

- Aufgabe: Berechnen Sie Inverse in Z_{10}

a	1	2	3	4	5	6	7	8	9
a^{-1}									

Primkörper

Hochschule Osnabrück
University of Applied Sciences

Sei p eine Primzahl. Da p (außer 1 und p) keine Teiler hat,

=> ggT(a,p) = 1 für alle a aus {1,…, p-1}

=> Jede Zahl aus Z_p\{0} hat damit ein multiplikatives Inverses

=> Z_p = GF(p) ist ein Körper (Galois Field*) !!

Körper ≈ „wir können wie gewohnt addieren, subtrahieren, multiplizieren und dividieren (außer durch 0)“

Bsp. für Körper: Rationale Zahlen, reelle Zahlen

Aufgabe: Berechnen Sie Inverse in GF(7)

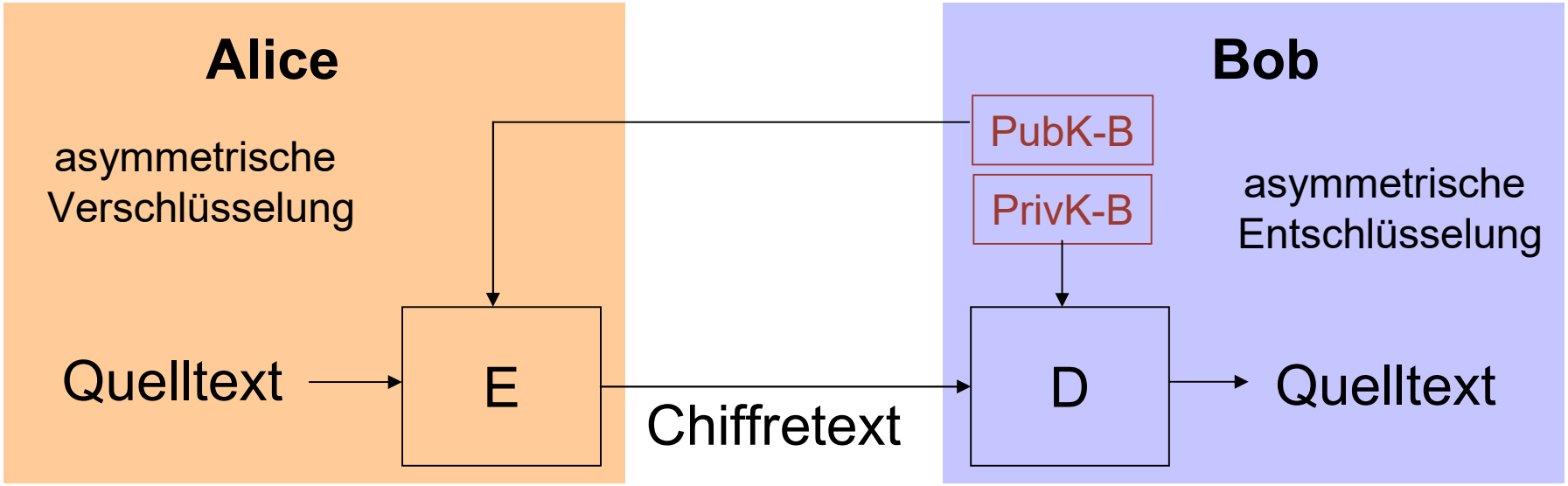
a	1	2	3	4	5	6
a^{-1}						

(* Evariste Galois, 1811-1832)

Asymmetrische Verschlüsselung

Hochschule Osnabrück
University of Applied Sciences

- Verwendet wird das Schlüsselpaar des Empfängers (Bob)
- Schlüsselpaar von Bob besteht aus:
  - öffentlicher Schlüssel PubK-B (public key), zur Verschlüsselung genutzt
  - privater / geheimer Schlüssel PrivK-B (private key), zur Entschlüsselung verwendet



RSA Verfahren

Hochschule Osnabrück
University of Applied Sciences

- Rivest, Shamir, Adleman (1978)
- Vorbereitung des Empfängers:
  - Modul $m = p \cdot q$ ist Produkt zweier großer Primzahlen
  - Öffenlicher Exponent $e < \varphi(m)$, geheimer Exponent $d < \varphi(m)$
    - $e \cdot d \bmod \varphi(m) = 1 \qquad \varphi(m) = (p-1) \cdot (q-1) \qquad (\mathrm{ggT}(e, \varphi(m))$ muss 1 sein!)
  - Öffentlicher Schlüssel (m,e), privater Schlüssel (p,q,d)
- RSA Verschlüsselung und Entschlüsselung:
  - Verschlüsselung von P, $0 \le P < m$: $C = (P^e) \bmod m$
  - Entschlüsselung von C: $P = (C^d) \bmod m$
- RSA Schlüssellänge = Anzahl der Bits des Moduls m

Sicherheit des RSA-Verfahrens

Hochschule Osnabrück
University of Applied Sciences

- RSA-Sicherheit beruht auf Faktorisierungsproblem (Berechnung von p und q zu gegebenem m)
  - es sind keine effizienten Faktorisierungsalgorithmen bekannt
  - Modul *m* Mindestlänge: 2048 Bit
  - 768 Bit Modul Januar 2010 geknackt, 795 Bit Dezember 2019
- Quantencomputer könnten dem RSA zukünftig gefährlich werden
  - Post Quantum Crypto: Krypto, die Quantencomputern stand hält
- Als Öffentlicher Exponent e wird häufig e = 65537 = $2^{16}+1$ gewählt
- Teilnehmer müssen unterschiedliche Primzahlen verwenden!
- Es gibt zahlreiche andere Dinge, die bei der Implementierung zu beachten sind

  => RSA nicht selbst implementieren! Bibliotheksfunktionen nutzen (PKCS#1)

Potenzieren

Hochschule Osnabrück
University of Applied Sciences

- Wie viele Multiplikationen werden zur Berechnung von a^n benötigt?
  - Naives Vorgehen: $a^n = a \cdot a \cdot \ldots \cdot a$ => n-1 Multiplikationen
- Aufgabe: Versuchen Sie es mit weniger Multiplikationen

	#Mult	Rechenweg
a^2		
a^4		
a^8		
a^{16}		

	#Mult	Rechenweg
$a^{(2^n)}$		
a^9		
a^{21}		

- Anzahl Mult. allgemein:
- Potenzen sind über Square-and-Multiply Algorithmus effizient berechenbar

Generatoren

Hochschule Osnabrück
University of Applied Sciences

- Ein Element g aus $Z_p\backslash\{0\}$ heißt *Generator* oder *Primitivwurzel*, wenn die Potenzen von g jedes Element aus $Z_p\backslash\{0\}$ erzeugen
- Beispiel: 6 ist ein Generator von $Z_{11}\backslash\{0\}$

a	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}
a=6	1	6	3	7	9	10	5	8	4	2	1

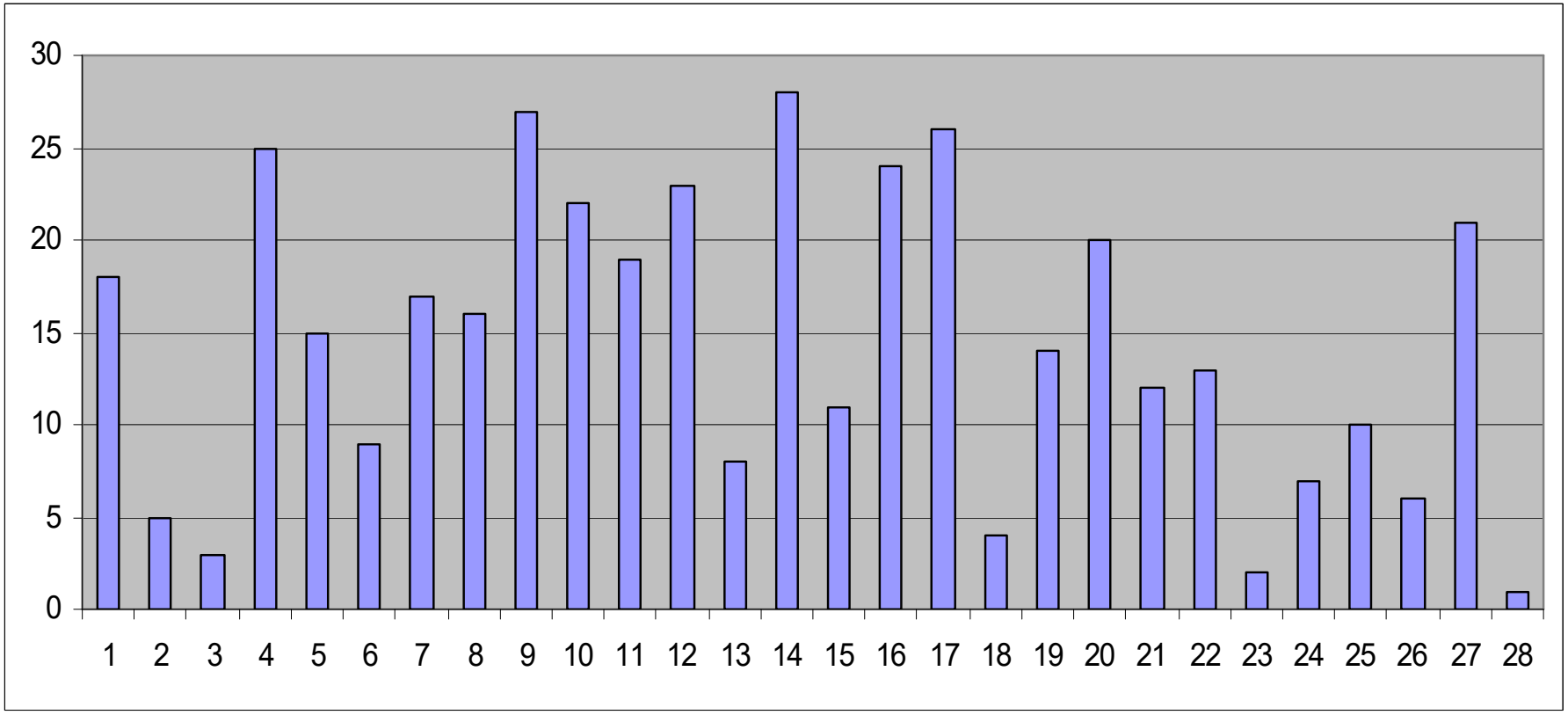
- Satz: Für jede Primzahl p gibt es in $Z_p\backslash\{0\}$ Generatoren
- Die Umkehrfunktion zum Potenzieren ist der diskrete Logarithmus
  - Potenzen: Effizient über Square-and-Multiply Algorithmus berechenbar
  - Diskrete Logarithmen: keine effizienten Algorithmen bekannt
    (Für eine 795 Bit Primzahl im Dez. 2019 geknackt)
  - DLP = Discrete Logarithm Problem

=> Das Potenzieren / die Exponentiation ist eine sog. Einwegfunktion

Potenzen eines Generator

Hochschule Osnabrück
University of Applied Sciences

- Das Potenzieren würfelt gut durcheinander
- Beispiel: Potenzen von 18 mod 29



Diskreter Logarithmus

Hochschule Osnabrück
University of Applied Sciences

- Die Umkehrfunktion zum Potenzieren in endlichen Gruppen ist der *Diskrete Logarithmus*

$b = a^n \pmod p \quad \Leftrightarrow \quad n = \log_a(b)$

- Tabelle mit Potenzen von 6 in $Z_{11}\backslash\{0\}$

a	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9
a=6	1	6	3	7	9	10	5	8	4	2

- Aufgabe: Tragen Sie die diskreten Logarithmen in die Tabelle ein!

a	1	2	3	4	5	6	7	8	9	10
$\log_6(a)$										

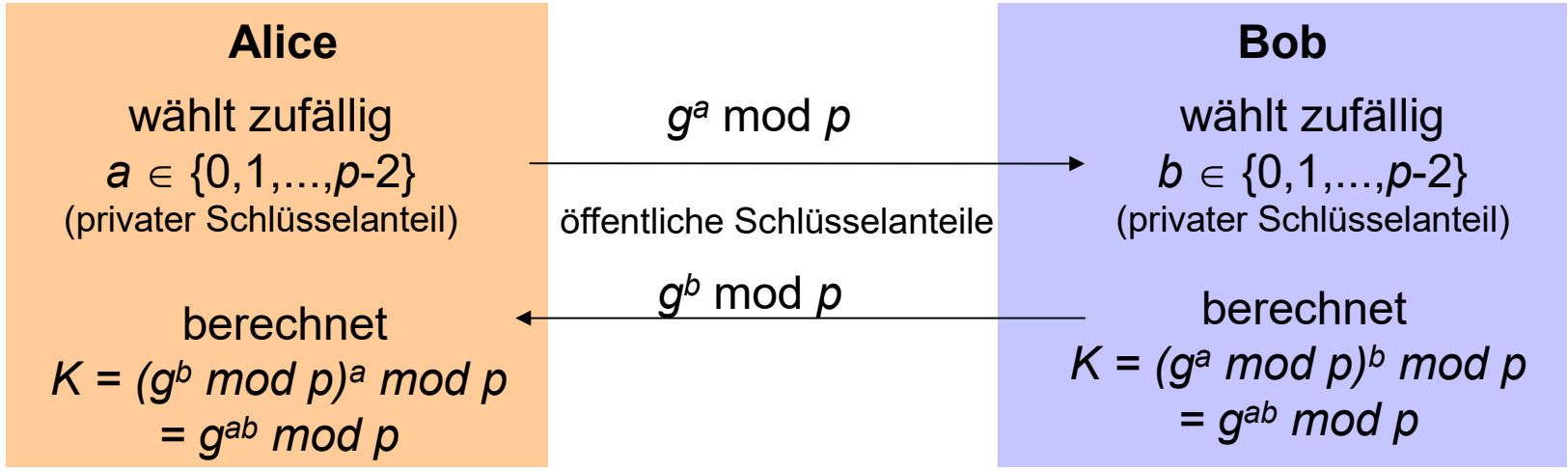
Diffie-Hellman Schlüsselaustausch

Hochschule Osnabrück
University of Applied Sciences

- Sicherheit beruht auf Problem des Diskreten Logarithmus
- Gegeben: Große Primzahl p, Generator g aus $Z_p\backslash\{0\}$
  - DH-Schlüssellänge = Anzahl Bits von p
- DH bietet *keine Authentisierung!*

(W. Diffie, M.E: Hellman, 1976)

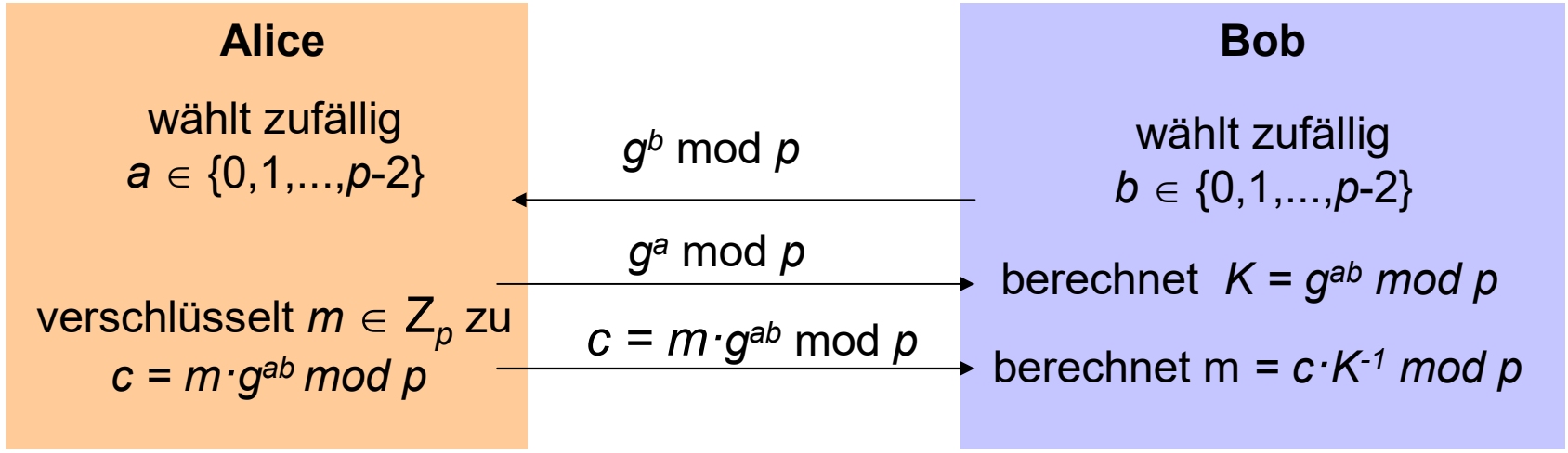
- Alice und Bob berechnen wie folgt gemeinsamen Schlüssel *K:*



ElGamal Verschlüsselung

Hochschule Osnabrück
University of Applied Sciences

- Gegeben: Große Primzahl p, Generator g aus $Z_p \backslash \{0\}$
- ElGamal = Diffie-Hellman Schlüsselaustausch und zusätzliche Verschlüsselung durch Multiplikation mod p
  - Bob: private key b, public key $g^b \bmod p$
  - Alice: private key a, public key $g^a \bmod p$



(Taher ElGamal, 1985)

Problem großer Schlüssellängen

Hochschule Osnabrück
University of Applied Sciences

- Nachteil: Sicherer Einsatz asymmetrischer Verfahren wie RSA oder Diffie-Hellmann über GF(*p*) erfordert große Schlüssellängen

- Idee: Wähle „kompliziertere Gruppe“

  => Algorithmen zur Berechnung diskreter Logs sind auch „komplizierter“ bzw. funktionieren in der komplizierteren Gruppe nicht mehr

  => geringere Schlüssellängen ausreichend

=> Verwendung von Punktemengen *elliptischer Kurven* über GF(*p*) als Gruppe

Vergleich Schlüssellängen

Vergleich Schlüssellängen

Schlüssellänge symmetrisch	RSA / DH	Elliptische Kurven ECDH, ECDSA,...
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Kryptosysteme über elliptischen Kurven (EC, elliptic curve)

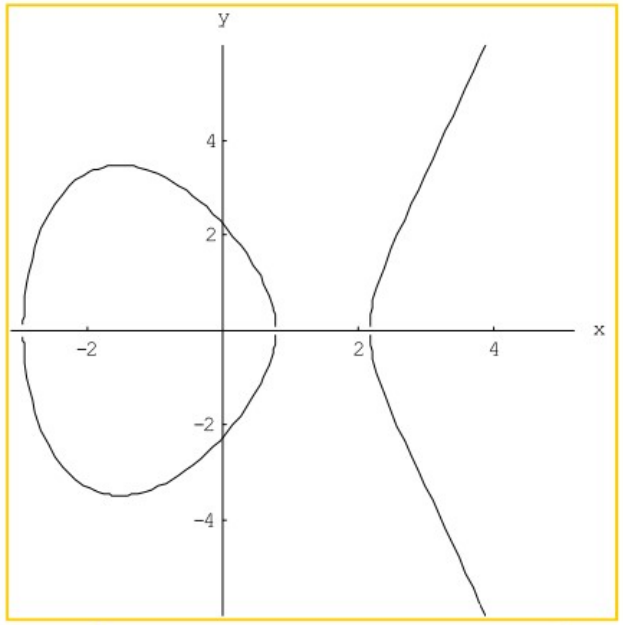
- Koblitz, Miller, Vanstone, Menezes (1985-1990)
- ECDH = Elliptic Curve Diffie Hellman (Schlüsselaustausch)
- ECIES = Elliptic Curve Integrated Encryption Scheme

Elliptic Curve Crypto

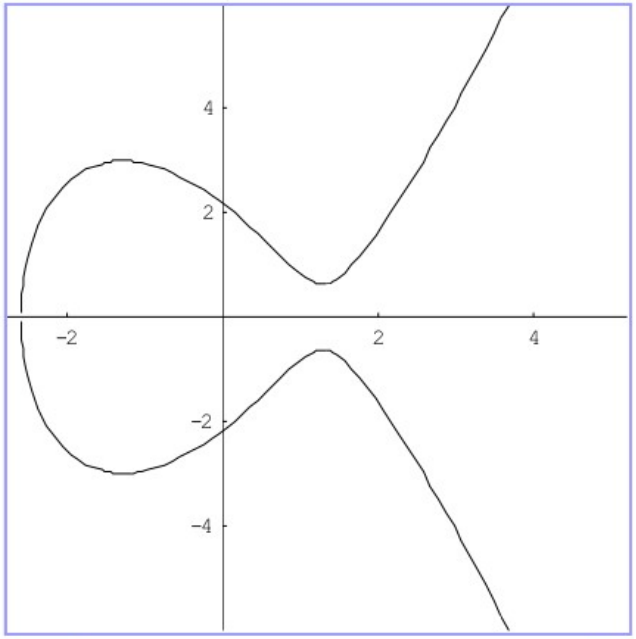
Hochschule Osnabrück
University of Applied Sciences

- Kryptographie über elliptischen Kurven (EK)
- EK: Menge von Paaren (x,y) reeller Zahlen, die Kurvengleichung erfüllt.

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



$y^2 = x^3 - 7x + 5$

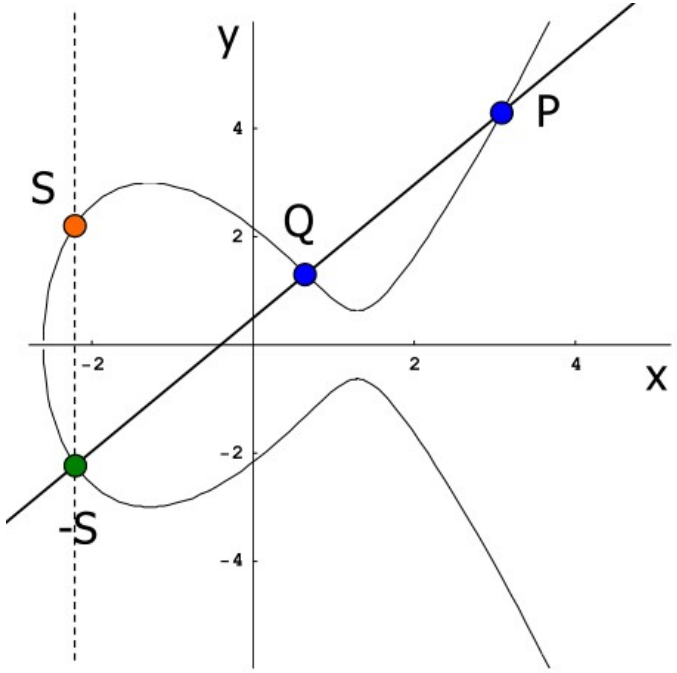


$y^2 = x^3 - 5x + 4.7$

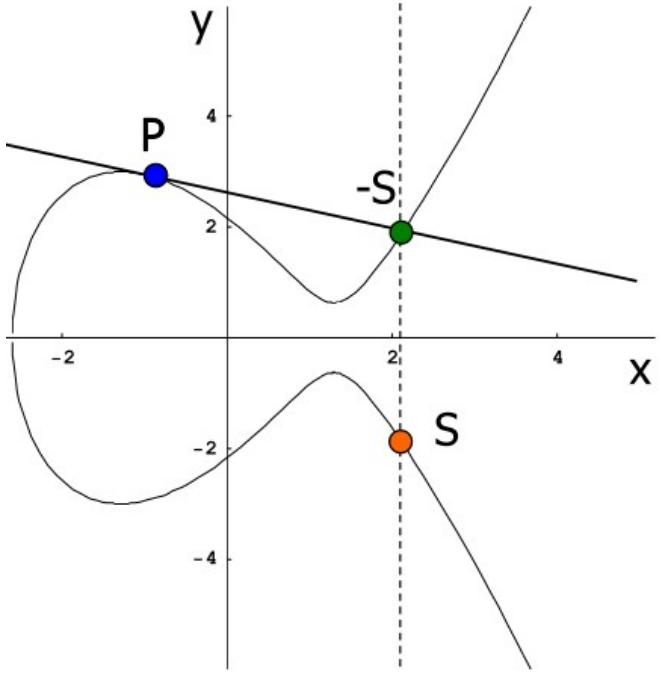
EK: Punkteaddition und Punktverdopplung

Hochschule Osnabrück
University of Applied Sciences

- Punktaddition: S = P + Q
- Gerade durch P und Q hat weiteren Schnittpunkt (-S) mit EK
- Schnittpunkt an x-Achse spiegeln ergibt S = P+Q



- Punktverdoppelung: S = P + P = 2P
- Tangente an P hat weiteren Schnittpunkt (-S) mit EK
- Schnittpunkt an x-Achse spiegeln ergibt S = 2P



EK: Inverse, neutrales Element, und Subtraktion

Hochschule Osnabrück
University of Applied Sciences

- **Inverses** = Spiegelung an x-Achse

  P = (x,y) => – P = (x,-y)

- Gerade durch P und -P ist senkrecht

  => kein weiterer Schnittpunkt mit EK

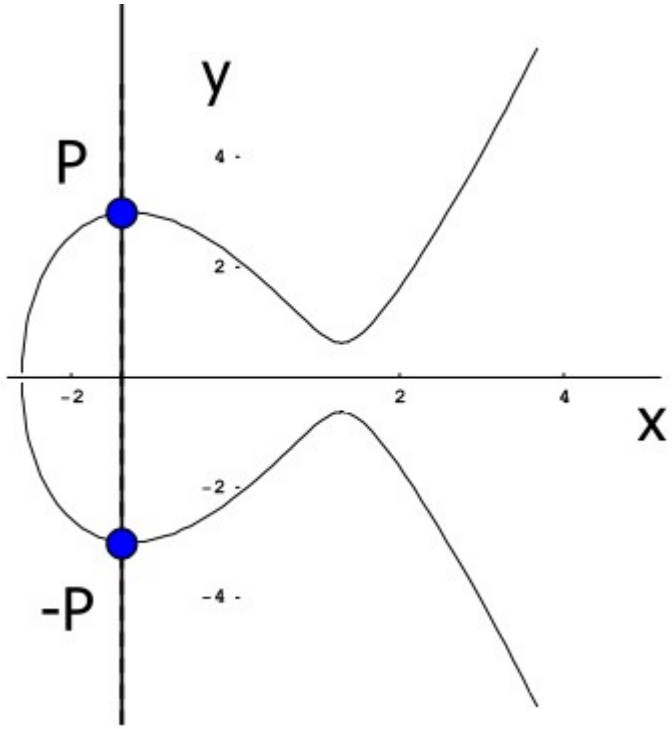
  - Gerade schneidet EK „im Unendlichen“ im abstrakten Punkt $\mathcal{O}$

- Punkt $\mathcal{O}$ ist **neutrales Element** der Gruppe

  $P - P = P + (-P) = \mathcal{O}$

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

Subtraktion: P – Q = P + (– Q)



EK: Formeln für Punktarithmetik

Hochschule Osnabrück
University of Applied Sciences

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

Punktaddition: $P = (x_P, y_P)$, $Q = (x_Q, y_Q)$ $(x_{P+Q}, y_{P+Q}) = P+Q$, $x_P \neq x_Q$

Steigung berechnen: $d = (y_Q - y_P) / (x_Q - x_P)$,

3. Schnittpunkt der Gerade mit der Kurve berechnen und spiegeln ergibt

=> $x_{P+Q} = d^2 - x_P - x_Q$ $y_{P+Q} = d \cdot (x_P - x_S) - y_P$

Punktverdopplung: $P = (x_P, y_P)$, $2P = (x_{2P}, y_{2P}) = P+P$,

Tangentensteigung: $d = (3x_P^2 + a) / (2y_P)$,

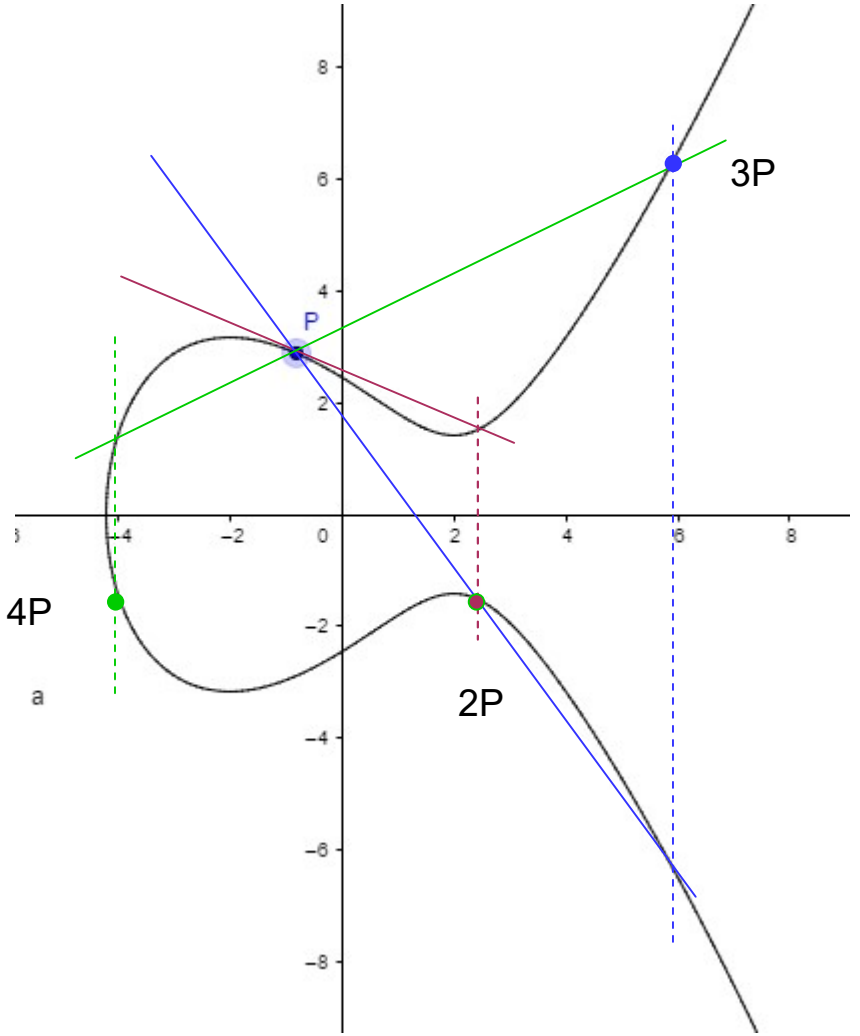
Schnittpunkt der Tangente mit der Kurve berechnen und spiegeln ergibt

=> $x_{2P} = d^2 - 2x_P$ $y_{2P} = d\,(x_P - x_{2P}) - y_P$

Beispiel:

Hochschule Osnabrück
University of Applied Sciences

- Berechnung von 2P, 3P und 4P



EK Arithmetik und DLP

Hochschule Osnabrück
University of Applied Sciences

- Addieren: P + Q
- Inverses: P=(x,y) => – P = (x, -y)
- Neutrales Element: $\mathcal{O}$
- Subtrahieren: P – Q = P + (– Q), P-P = 0P = $\mathcal{O}$
- Verdoppeln: P + P = 2P, 2P + 2P = 4P, 4P+4P = 8P, …
- Mult. mit ganzen Zahlen: 11P = P + 2P + 8P, -11P = –(11P)
  - Vorgehen wie bei der Potenzierung
  - Berechnung von nP erfordert im Mittel 1,5 $\log_2(n)$ Punktadditionen
- Diskretes Logarithmus Problem auf elliptischen Kurven:
  - Zu gegebenen P und nP sind für die Berechnung von n keine effizienten Algorithmen bekannt

Elliptische Kurven über GF(*p*)

Hochschule Osnabrück
University of Applied Sciences

- Beispiel: EK definiert durch $y^2 = x^3 + 2x + 5$ über GF(13)

EK = { (2,2), (2,-2), (3,5), (3,-5), (4,5), (4,-5), (5,6), (5,-6), (6,5), (6,-5), (8,0), ∞ }
EK hat 12 Punkte

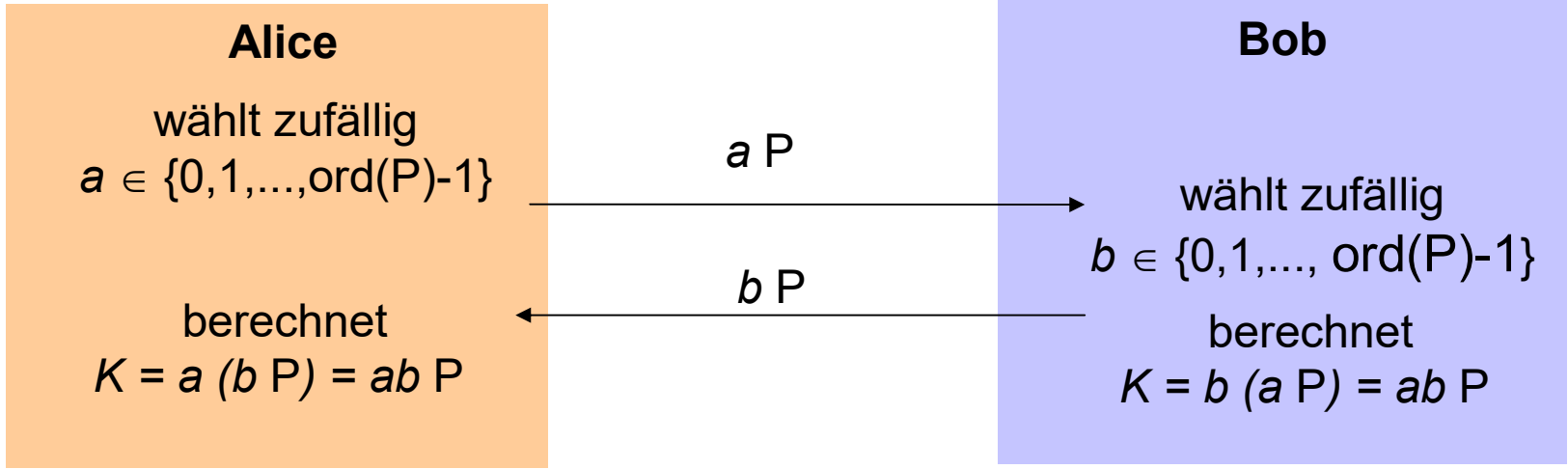
- Arithmetik funktioniert ! (Geometrische Darstellung geht nicht mehr)
- Anzahl der Punkte auf einer ellip. Kurve über GF(*p*) ist ungefähr *p*
  - Hasse (1936): Anzahl der Punkte liegt im Bereich *p*+1 ± SQRT(*p*)

ECDH: Ellip. Curve Diffie-Helmann

Hochschule Osnabrück
University of Applied Sciences

- Diffie-Hellman Schlüsselaustausch über elliptischen Kurven
- Gegeben: ellip. Kurve EK, Punkt P mit großer Ordnung ord(P)
- *Keine Authentisierung!*

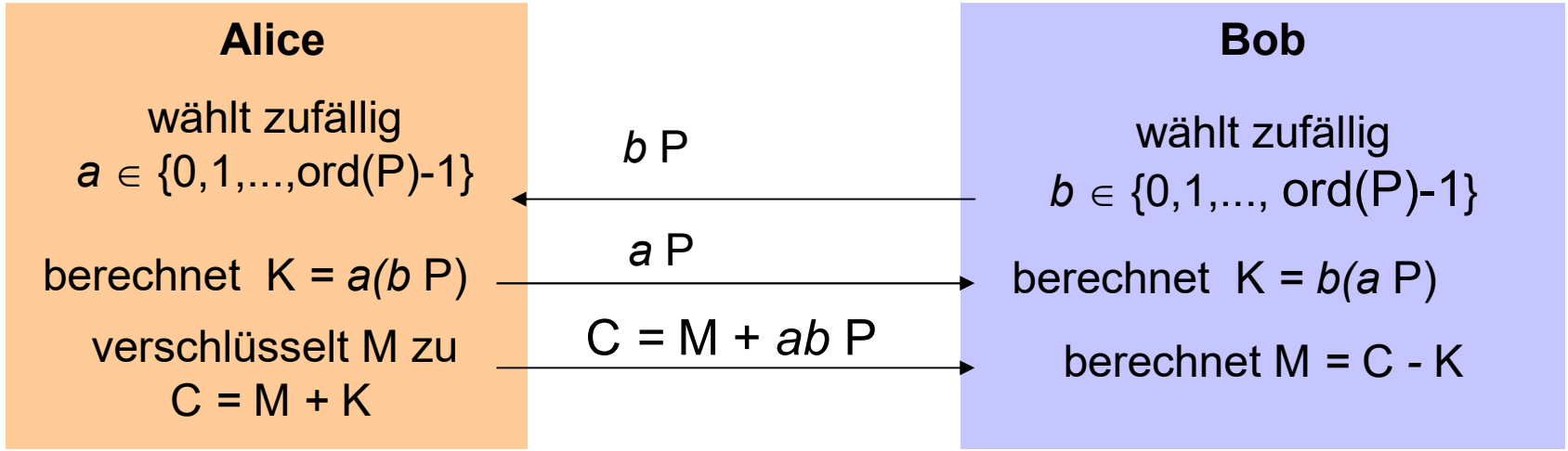
- Alice und Bob berechnen wie folgt gemeinsamen Schlüssel *K:*



EC ElGamal Verschlüsselung

Hochschule Osnabrück
University of Applied Sciences

- Gegeben: ellip. Kurve EK, Punkt P mit großer Ordnung ord(P)
  - Erinnerung: ElGamal = DH + Verschlüsselung durch Punktaddition
  - Bob: private key *b*, public key *b*P; Alice: private key *a*, public key *a*P
- Nachricht M ist hier ein Punkt der Kurve EK



Beispielkurven

Hochschule Osnabrück
University of Applied Sciences

- NIST Kurven: P-192,…,P-521
- SECG Kurven: secp192k1, …, secp521r1
- Brainpool Kurven: brainpoolP160r1, …, brainpoolP512t1
- Curve 25519 (Bernstein, 2005, RFC 7748 (2016))
  - Primzahl $p = 2^{255}-19$ (255 Bit Kurve)
  - EK: $y^2 = x^3 + 486662\,x^2 + x$
  - P = (9, $14_{78161.9447589544.7910205935.6840998688.7264606134.6164752889.6488183775.55862374}01$)
  - P hat Primzahl Ordnung

    $$ord(P) = 2^{252} + 0x14def9dea2f79cd65812631a5cf5d3ed$$

  - Erinnerung: Beim Diffie-Hellmann werden *a* und *b* zufällig gewählt aus dem Bereich

    $$0,1,2,\ldots\ ord(P)-1$$

Auswirkungen Quantencomputer

Hochschule Osnabrück
University of Applied Sciences

- Die Sicherheit asymmetrische Kryptoverfahren beruht auf math. Problemen, zu deren Lösung es derzeit keine effizienten Algorithmen gibt
  - Faktorisierung ganzer Zahlen, Diskreter Logarithmus
- Asymmetrische Verfahren sind langsamer als symmetrische
  - Performance-Faktor: ~ 1000

- Auswirkungen Quantencomputer
  - Asym. Verfahren (bis auf Ausnahmen) in Polynomzeit lösbar
    - Shor Algorithmus (Peter Shor, 1994)
  - Symmetrische Verfahren: Sicherheit auf ½ Schlüssellänge reduziert
  - *Post Quantum Cryptography:* Suche nach asym. Verfahren, die resistent gegen Quantencomputer sind

Vergleich sym.↔asym. Verschlüsselung

Hochschule Osnabrück
University of Applied Sciences

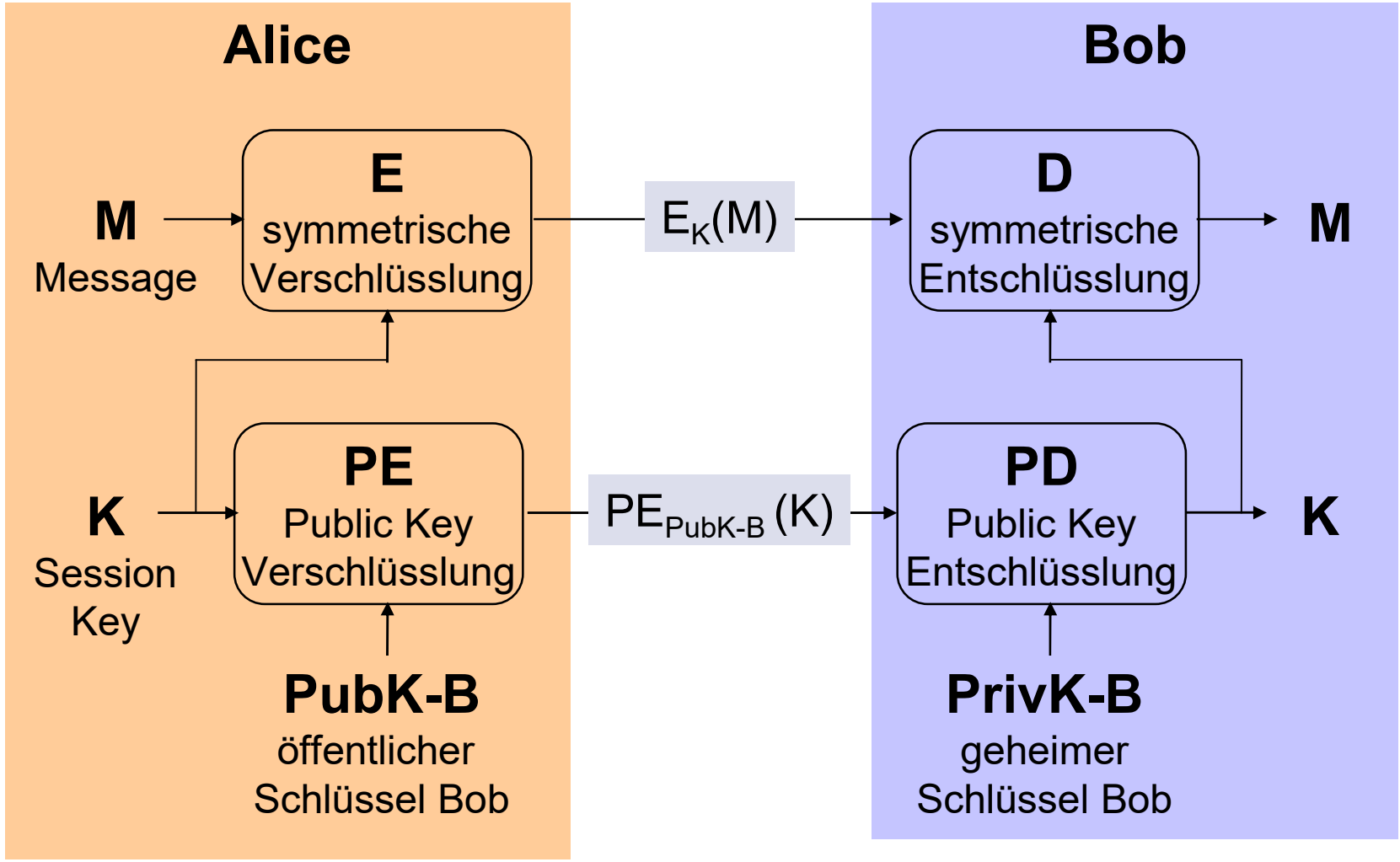
Vergleich	Symmetrische Verschlüsselung	Aymmetrische Verschlüsselung
Vorteil	Schnell	Kein Schlüssel-austausch Problem
Nachteil	Problem des Schlüsselaustauschs	langsam

- Hybride Verschlüsselung kombiniert Vorteile beider Verfahren
  - Nutzung asymmetrischer Verfahren für den Schlüsselaustausch
  - Nutzung symmetrischer Verfahren zur Klartextverschlüsselung

- ECIES: Elliptic Curve Integrated Encryption Scheme

Hybride Verschlüsselung: Funktion

Hochschule Osnabrück
University of Applied Sciences



Forward Secrecy

Hochschule Osnabrück
University of Applied Sciences

- Teilnehmer wechseln Ihr Schlüsselpaar selten.

=> Bei der hybriden Verschlüsselung mit Bob ist der SessionKey immer mit Bobs öff. Schlüssel verschlüsselt

=> wenn Bobs privater Schlüssel geknackt wird, können *alle bisher ausgetauschten SessionKeys* nachträglich entschlüsselt werden!

- Schlüsselaustausch per Diffie-Hellman:

=> Für jede Session wird erneut ein zufälliger Schlüssel generiert.

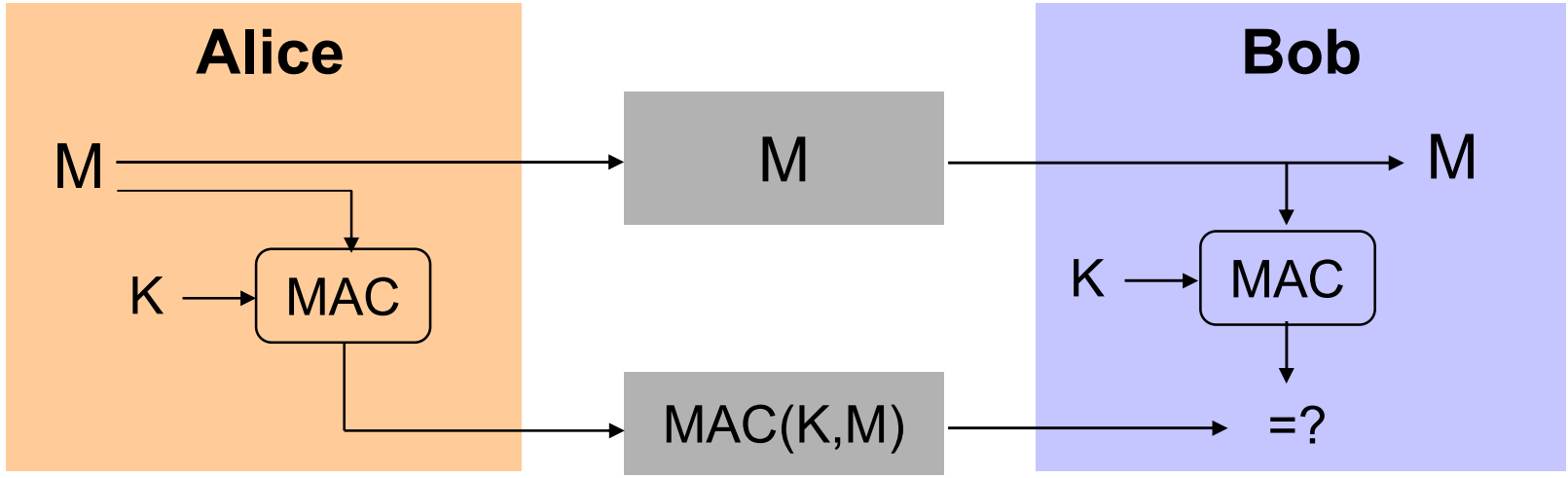
=> Wenn der Schlüssel geknackt wird, ist *nur diese eine Session* betroffen. Die Schlüssel andere Sessions müssten separat geknackt werden.

Diese Sicherheitseigenschaft heißt (Perfect) Forward Secrecy

Erinnerung: Message Authentication Codes

Hochschule Osnabrück
University of Applied Sciences

- Kryptographische Prüfsummen (symmetrisches Verfahren)
  - Funktionsweise s. Abbildung
- Sicherheitsanforderungen
  - Ohne K soll es einem Angreifer praktisch unmöglich sein, für eine Nachricht M eine gültige Prüfsumme zu berechnen.
  - Bei gegebener Nachricht M und Prüfsumme MAC(K,M) darf es einem Angreifer praktisch weder möglich sein, K zu berechnen, noch einen zweite Nachricht M' mit identischer Prüfsumme zu finden.



Digitale Signaturen - Motivation

Hochschule Osnabrück
University of Applied Sciences

MACs: Schwachstellen

- Während des Datenaustausches ist durch Einsatz von MACs eine Authentisierung der Daten und der Kommunikationspartner gegeben.
- Nach dem Datenaustausch ist gegenüber Dritten nicht beweisbar, von wem die Daten gesendet wurden
  - Alice und Bob kennen beide den zugehörigen geheimen Schlüssel
  - Beide sind in der Lage gültige MACs zu berechnen
  - Ein Dritter kann nicht entscheiden, ob ein gegebenen MAC von Alice oder Bob stammt

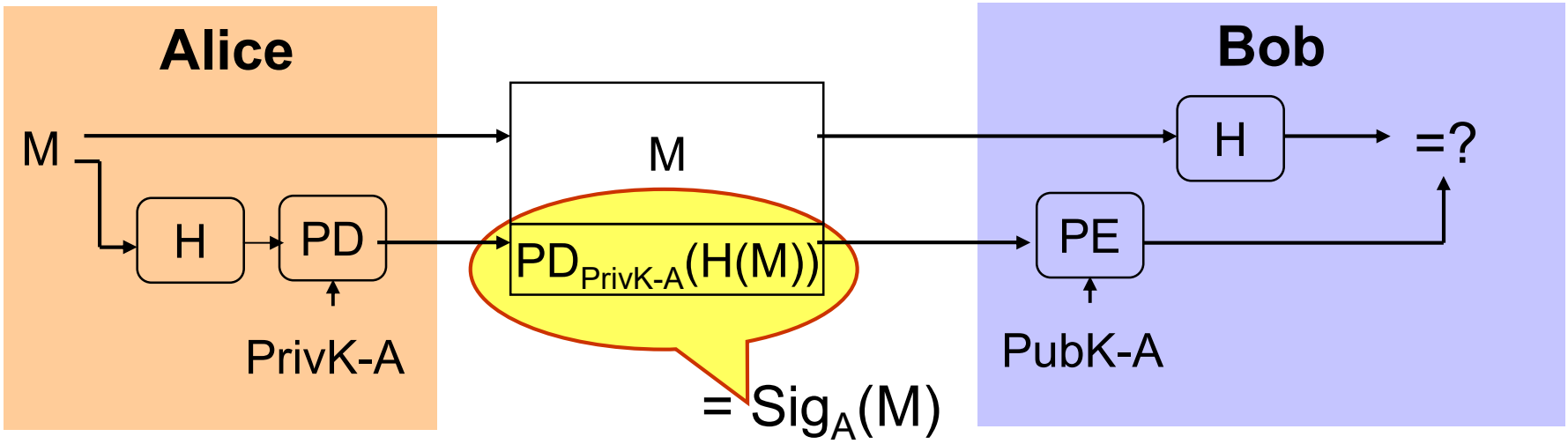
Beweisbarkeit / Nichtabstreitbarkeit => Digitale Signaturen

- Bob darf keine Signaturen von Alice fälschen können
- Signatur muss Alice eindeutig ausweisen

Digitale Signaturen mit asymmetischem Kryptosystem

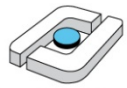
Hochschule Osnabrück
University of Applied Sciences

- PD = asym. Entschlüsselung, PE = asym. Verschlüsselung
- Alice erzeugt ihre Digitale Signatur durch Anwendung ihres privaten Schlüssels PrivK-A auf den Hashwert der Nachricht M
- Bob (und jeder andere) kann die Signatur mit dem öff. Schlüssel von Alice PubK-A prüfen



- Beispielverfahren:
  - RSA with SHA-256
  - DSA, ECDSA (DSA = Digital Signature Algorithm)

Aufgaben: bitte zuordnen



Hochschule Osnabrück
University of Applied Sciences

Aktuell sichere Schlüssellängen zuordnen

128 Bit,

256 Bit,

2048 Bit

(eine doppelt)

Schlüssellänge	Verfahren
	Symmetrische Verschlüsselung
	Hashwertlänge einer Hashfunktion
	RSA
	Elliptische Kurven Kryptoverfahren

Hugo schickt eine verschlüsselte und signierte Nachricht an Dörte.
Bitte Einsatzzweck zu Schlüsseln zuordnen:

Einsatzzweck

1. Verschlüsselung
2. Entschlüsselung
3. Signierung
4. Signaturprüfung

Nr.	Schlüssel
	Öffentlicher Schlüssel Hugo
	Privater Schlüssel Hugo
	Öffentlicher Schlüssel Dörte
	Privater Schlüssel Dörte

Erzeugung sicherer pseudo-zufälliger Zahlen

Hochschule Osnabrück
University of Applied Sciences

- Benötigt z.B. zur Schlüsselerzeugung
- Erzeugung von Bitfolgen ist ausreichend
  - Erzeugung von Zahlen im Intervall [0,n]: Wandle jeweils $\log_2(n)+1$ Bits in eine natürliche Zahl x um. Falls x >n herauskommt, x verwerfen und neuer Versuch
- Zufallsbitgenerator (Random Bit Generator)
  - erzeugt eine Folge *statistisch unabhängiger* und *gleich verteilter* Bits
- Pseudozufallsbitgenerator (Pseudo Random Bit Generator, PRBG)
  - deterministischer Algorithmus, der auf Eingabe einer zufälligen Bitfolge der Länge *s* (Seed) eine Bitfolge erzeugt, die deutlich länger ist als *s* (Multiplikatoreffekt) und dieselben statistischen Eigenschaften wie eine echte Zufallsfolge aufweist
  - z.B. Häufigkeiten (Bit, Teilfolgen), Verteilung von Runs (Folgenabschnitte mit nur „1“ oder nur „0“) , Autokorrelation, etc.

Kryptographisch sichere PRBG

Hochschule Osnabrück
University of Applied Sciences

- Ein PRBG heißt kryptographisch sicher, wenn es *praktisch nicht* möglich ist, aus Kenntnis ersten *n* Folgenglieder, das *n+1* Folgenglied mit einer Wahrscheinlichkeit von größer als 0.5 vorherzusagen
- Die von einem kryptographisch sicheren PRBG erzeugte Pseudozufallsfolge besteht sämtliche statistischen Tests (mit polynomialer Laufzeit)

LCG und Schieberegister

Hochschule Osnabrück
University of Applied Sciences

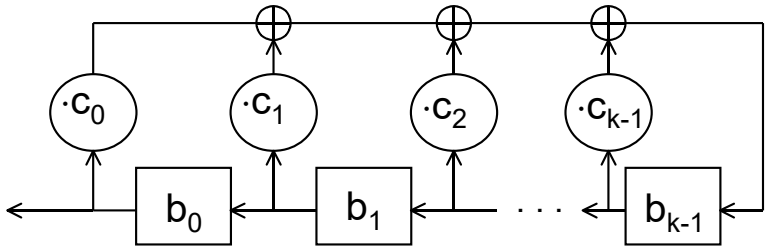
Linear Congruential Generator (LCG)

- Für einen Modul $m > 0$, einen Faktor a, $0 \leq a < m$, einen Summanden c, $0 \leq c < m$, und einen Startwert x_0, $0 \leq x_0 < m$, berechnet der Linear Congruential Generator Zahlen x_i, $i > 0$, durch

$$x_{i+1} := (a \cdot x_i + c) \bmod m$$

Folgen linear rückgekoppelter Schieberegister (LRS)

- Rückkopplung $c=(c_0,...,c_{k-1})$
  Startvektor $b=(b_0,...,b_{k-1})$
- i.d.R. werden binäre LRS betrachtet



Eigenschaften

- Mit geeigneten Parametern liefern beide Generatoren Folgen mit sehr guten statistischen Eigenschaften
- Folgen sind kryptographisch unsicher!
  LCG: 3-4 Folgenglieder reichen, um Folge vorherzusagen
  LRS: 2k Folgenglieder reichen, um Folge vorherzusagen

PRBG mit Blockchiffren

Hochschule Osnabrück
University of Applied Sciences

- Zur Erzeugung einer Pseudozufallsfolge kann eine Blockchiffre E im Output-Feedback-Mode (OFB) oder Counter-Mode (CTR) betrieben werden
  - vgl. Folie „Betriebsarten von Blockchiffren (2)“
- Diese Generatoren werden häufig in der Praxis eingesetzt. Es ist jedoch **nicht** nachgewiesen, dass diese Generatoren kryptographisch sicher sind!

Nachweisbar sichere PRBG

Hochschule Osnabrück
University of Applied Sciences

RSA Generator (iterierte RSA-Verschlüsselung)

- Sei *m* ein RSA-Modul, *e* ein zugehöriger öffentlicher Exponent und C_0, $1 < C_0 < m-1$, ein zufällig gewählter Startwert. Der RSA-Generator erzeugt die Folge b_i, i>0, von Bits durch iterierte RSA-Verschlüsselung:

$$C_i := (C_{i-1})^e \bmod m, \; b_i \text{ ist dann das niederwertigste Bit von } C_i$$

Blum-Blum-Shub-Generator (iterierte modulare Quadrierung)

- Sei $m=pq$ das Produkt Primzahlen *p* und *q* die beide modulo 4 den Rest 3 ergeben. Sei C_0, $1 < C_0 < m-1$ ein zufällig gewählter Startwert. Der Blum-Blum-Shub-Generator erzeugt die Folge b_i, i>0, von Bits durch:

$$C_i := (C_{i-1})^2 \bmod m, \; b_i \text{ ist dann das niederwertigste Bit von } C_i$$

Eigenschaften:

- Mit einem Algorithmus, der Bits besser als mit einer Wahrscheinlichkeit von 0.5 vorhersagt, könnte beim RSA-Generator das RSA Verfahren gebrochen werden bzw. beim BBS-Generator der Modul faktorisiert werden
- Sehr schlechte Performance, da pro RSA-Verschlüsselung bzw. modularer Quadrierung lediglich ein Bit erzeugt wird.

Erzeugung echter Zufallszahlen

Hochschule Osnabrück
University of Applied Sciences

- Hardware Zufallsgeneratoren: Basis: physikalische Phänomene
  - Z.B.: Zeit zwischen Partikelemissionen beim radioaktiven Verfall
  - Z.B.: Messung thermischen Rauschens
- Zufallsquellen für softwarebasierte Zufallsgeneratoren
  - Systemzeit, Zeit zwischen Tastaturanschlägen
  - Mouseposition / Mousebewegung
  - Inhalte von Ein- und Ausgabepuffern
  - Betriebssystemkennwerte: CPU-Last, Logdaten

- **Wichtig!** Hinreichend viele Zufallsquellen kombinieren, um genügend Redundanz / Zufälligkeit zu erhalten
  - Mindestens ca. 128 zufällige Bits

Aufgabe: bitte zuordnen

Hochschule Osnabrück
University of Applied Sciences

1. OFB
2. DH
3. CTR
4. AES
5. GCM
6. ECDSA
7. HMAC
8. DES
9. SHA-256
10. ChaCha20
11. POLY1305

	Zweck
	Blockchiffre
	Stromchiffre
	Nachrichtenauthentisierung
	Verbindlichkeit
	Hashfunktion
	Gleichzeitige Authentisierung und Verschlüsselung
	Schlüsselaustausch
	Generierung von Pseudozufallsfolgen