Thorsten Strufe

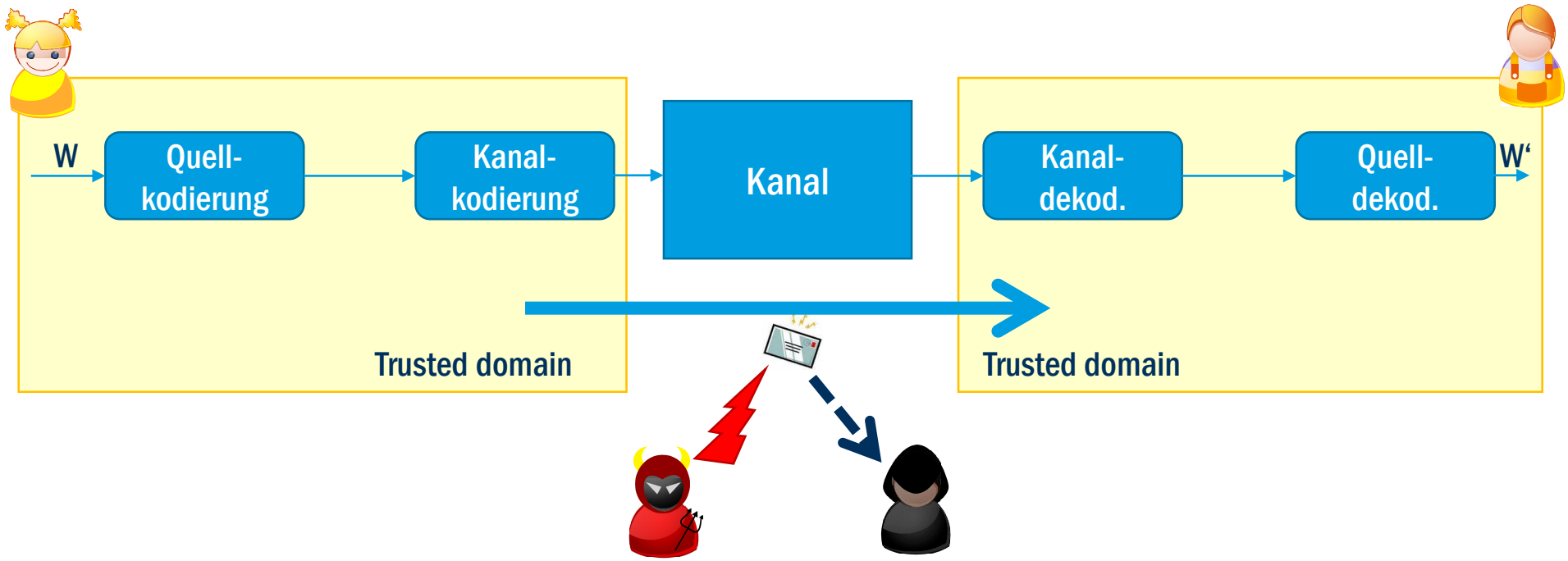
Resilient Networking

Modul 2: Background - Crypto

Disclaimer: Anleihen bei Dan Boneh, Mark Manulis

Dresden, SS 19

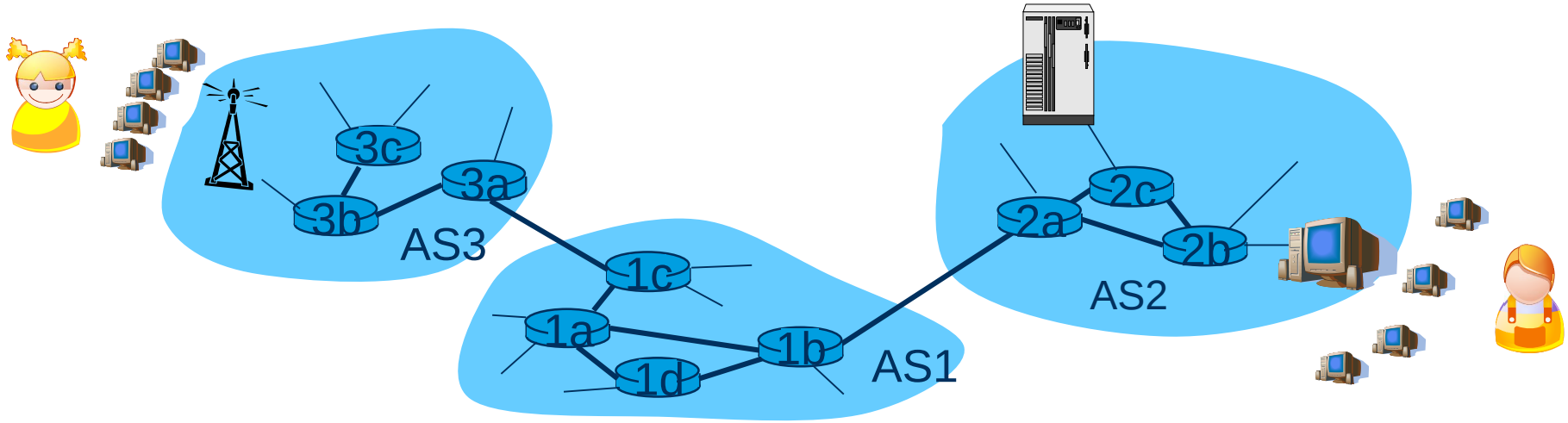
Das Kanalmodell und die Sicherheit



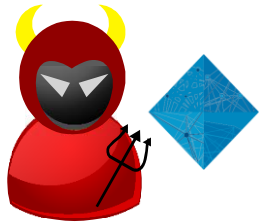
modifizierend
„active“

beobachtend
„passive“

Vernetzte Kommunikation



Eine Konkretisierung: Der Dolev - Yao Angreifer

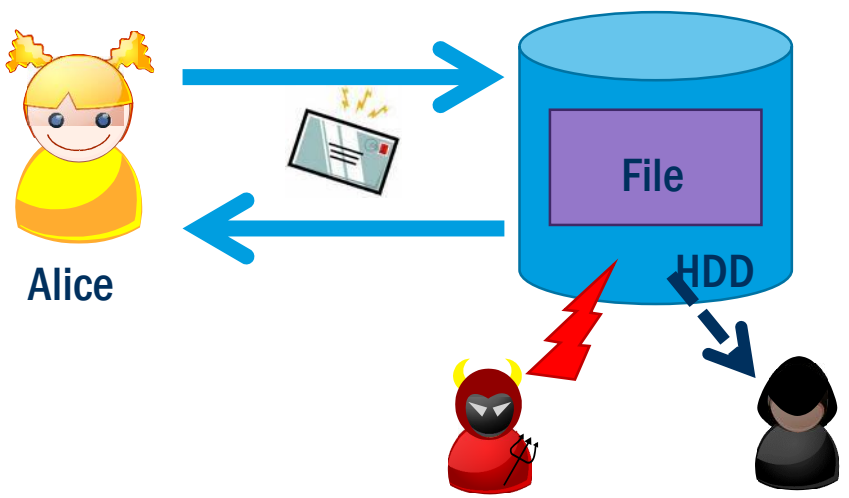


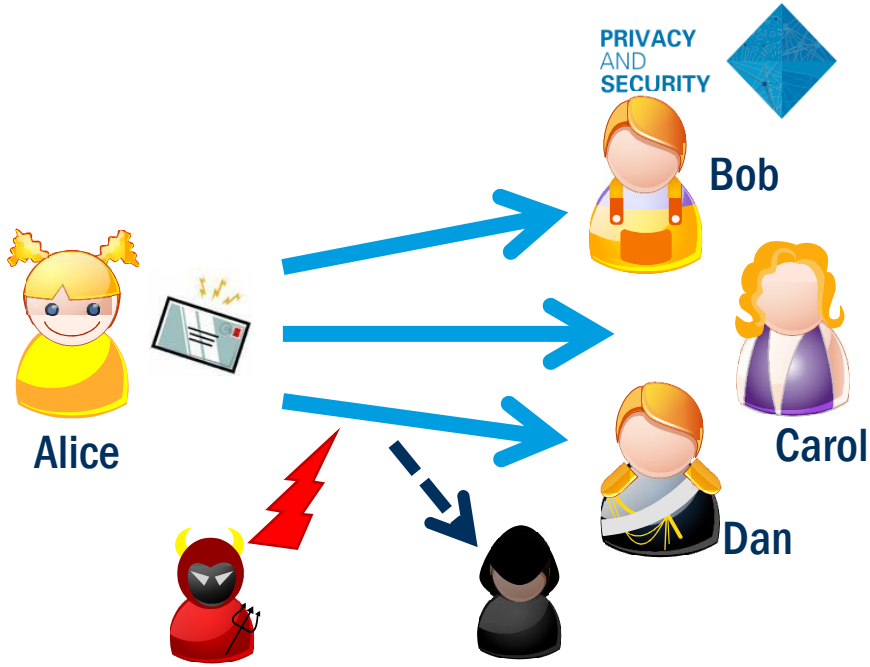
Mallory hat ***volle Kontrolle über den Kommunikationskanal***, kann:

- Nachrichten abhören („*eavesdropping*“, passiv)
- Auslieferung verzögern („delay“)
- Auslieferung unterdrücken („*suppression*“)
- Nachrichten erneut ausliefern („*replay*“)
- Nachrichten verändern („*manipulation*“ / „*tampering*“)
- Nachrichten fälschen („*forgery*“)

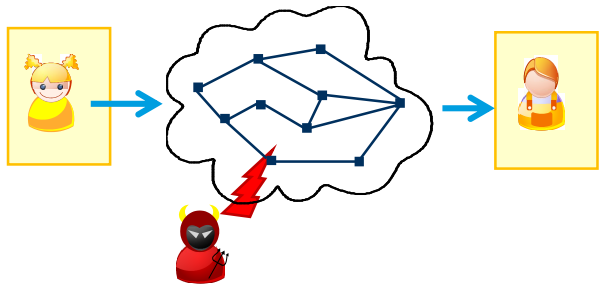
Aber:

- Mallory kann „sichere Kryptographie“ nicht brechen (*PPT Angreifer*)

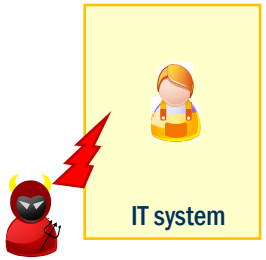




*Was wir **nicht** betrachten:*

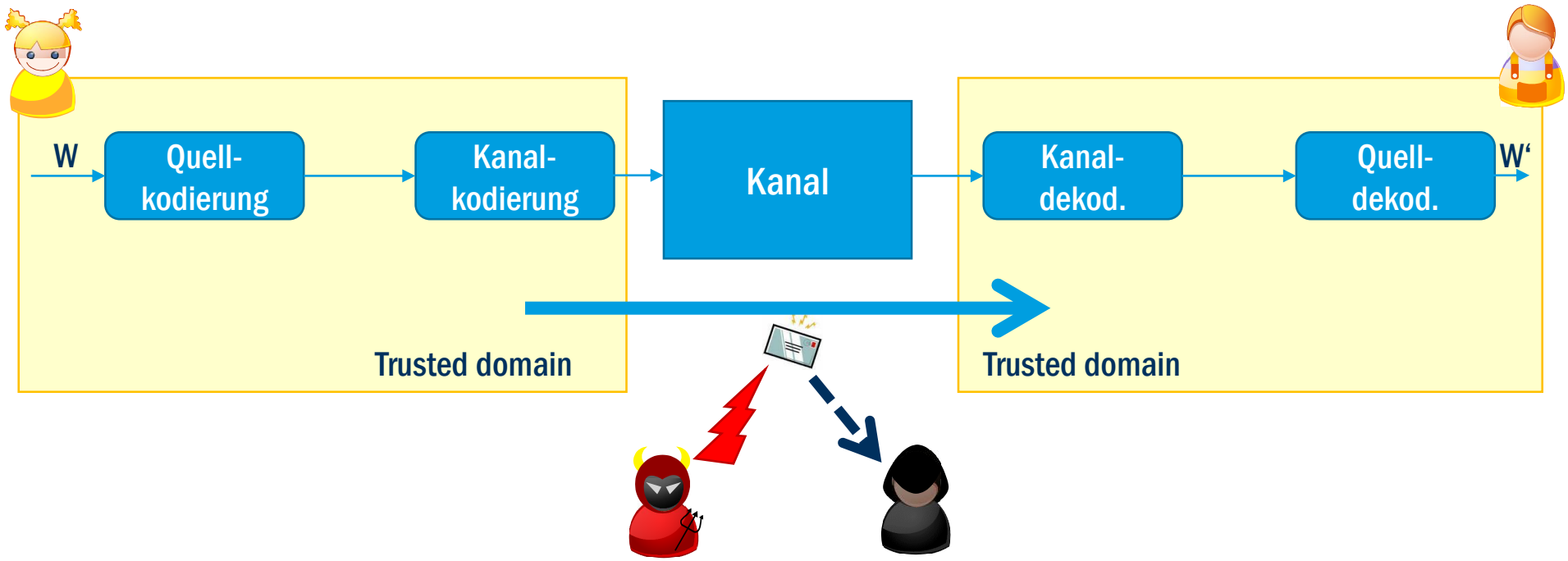


Rest



Syssec Lab

Das Kanalmodell und die Sicherheit



modifizierend
„active“

beobachtend
„passive“

Eve und Mallory: Das Angreifermodell

PRIVACY AND SECURITY

- Intention des Angreifers

  (Stören, verändern, unerlaubt zugreifen)

- Verhalten des Angreifers
  (passiv/aktiv, beobachtend/verändernd)

- Ressourcen (Capabilities)
  - “Rechenkapazität“ (komplexitätstheoretisch (un)beschränkt)
  - Verfügbare Mittel (Zeit, Geld)

- Kontrolle des Angreifers (Area of control)
  - Rollen des Angreifers (Nutzer, Außenstehender, ...)
  - Verbreitung des Angreifers (kontrollierte Subsysteme, Leitungen, ...)

Bedrohungen - Klassen

Informationsverlust (Abgehört, ausgespäht werden)
- Instanz liest Information, die nicht für sie bestimmt ist

Zerstörung/Modifikation von Information
- Information wird zerstört oder verändert

Fälschung von Information
- Instanz erzeugt Information in der Identität einer anderen Instanz

Maskerade
- Instanz gibt vor die Identität einer anderen Instanz zu haben

Authorisierungsverletzung
- Instanz nutzt Ressourcen ohne dazu autorisiert zu sein

Abstreiten von Ereignissen
- Instanz leugnet fälschlicherweise, an Ereignis beteiligt gewesen zu sein

Sabotage
- Mutwillige/geplante (Zer-)Störung von Diensten oder Systemen

Mechanismen zum Schutz (Security Services)

Authentifizierung (Authentication)

- Nachweis behaupteter Eigenschaft (Identität) einer Instanz

Signieren: Integritätsschutz (Data Integrity)

- Nachweis der Unversehrtheit von Information

Verschlüsseln: Vertraulichkeitsschutz (Confidentiality)

- Verhinderung unauthorisierten Zugriffs zu Information

Verbindlichkeit/Nicht-Abstreitbarkeit (Non repudiation)

- Nachweis der Teilnahme einer Instanz an einem Ereignis

Zugriffskontrolle (Access Control)

- Überwachung der legitimierten Zugriffsweise auf Ressourcen

X.800, ITU, 1991

Ziele der IT Sicherheit

Vertraulichkeit (Confidentiality)
- Übertragene und gespeicherte Daten dürfen nur legitimierten Empfängern zugänglich sein
- Vertraulichkeit der Identität wird als Anonymität bezeichnet

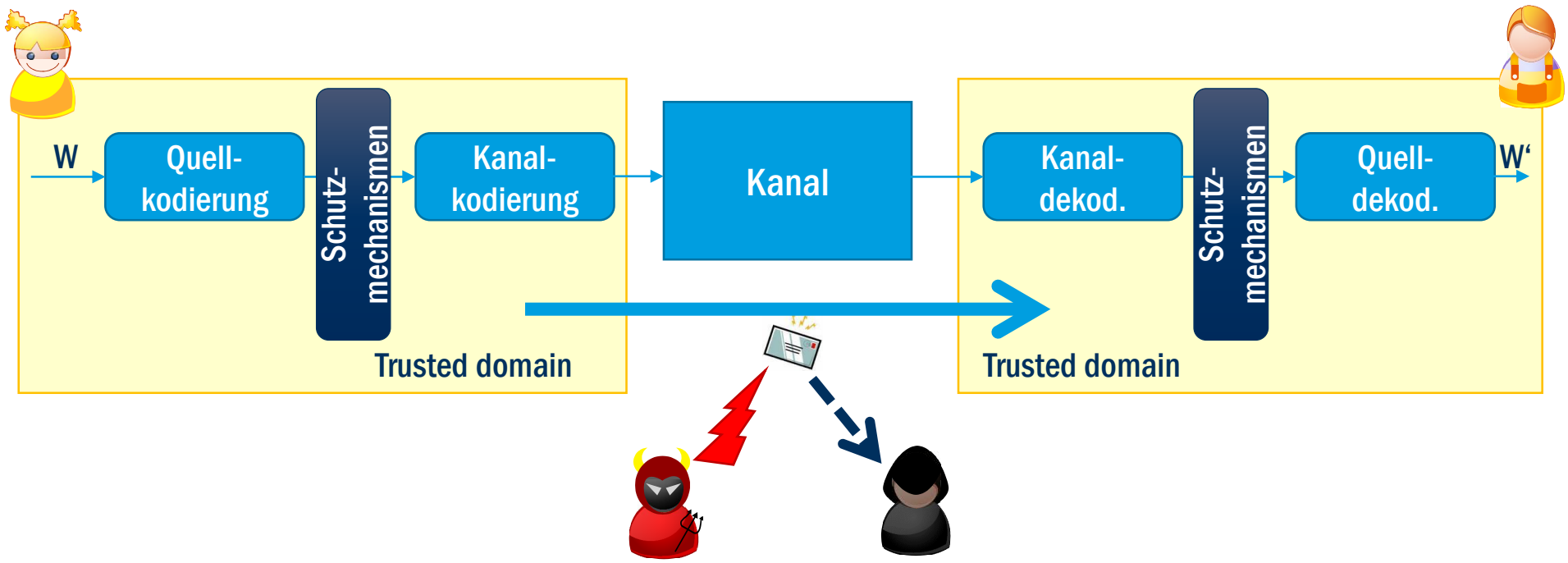
Integrität (Integrity)
- Veränderungen an Daten müssen detektiert werden
- (Bedarf der Identifikation des Absenders!)

Verfügbarkeit (Availability)
- Informationen und Dienste sollen berechtigten Nutzern in angemessener Frist zugänglich sein

Zurechenbarkeit (Accountability)
- Die verantwortliche Partei für eine Operation soll identifizierbar sein

Kontrollierter Zugriff (Controlled Access)
- Nur authorisierte Parteien sollen in der Lage sein, auf Dienste oder Informationen zuzugreifen

Erweiterung des Kanalmodells



Schutzmechanismen Vertraulichkeit/Integrität

Notwendigkeit von Algorithmen (und Protokollen)

Schutz der Vertraulichkeit:

- Klassisch Verschlüsselung

Schutz der Integrität:

- Signieren, „Message Authentication Codes (MAC)“

Beides mittels kryptographischer Algorithmen

Schutzmechanismen Vertraulichkeit

PRIVACY AND SECURITY

Definition von Mengen und Räumen:

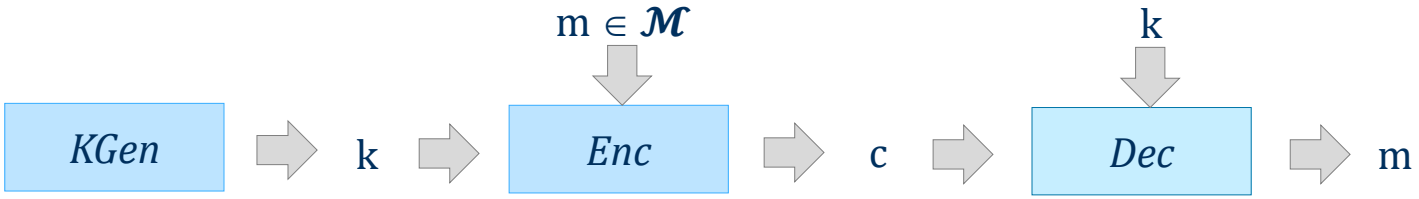
M: Raum potentieller Nachrichten (z.B. Worte über Alphabet)

K: Raum wählbarer Schlüssel (z.B. $\{0,1\}^{128}$)

C: Raum potentieller Chiffrate (z.B. $\{0,1\}^*$)

Algorithmen:

KGen() -> k ∈ K		randomisiert
Enc(k,m) -> c	mit c ∈ C, k ∈ K, m ∈ M	randomisiert
Dec(k,c) -> m		deterministisch



Korrektheit!

Dec(k,(Enc(k,m)) = m für alle k ∈ K und m ∈ M

Schutzmechanismen Integrität

Definition von Mengen und Räumen

M: Raum potentieller Nachrichten $(\{0,1\}^*)$

T: Raum potentieller „Tags“ $(z.B.\{0,1\}^{160})$

K: Raum wählbarer Schlüssel $(z.B.\ \{0,1\}^{128})$

Algorithmen

KeyGen() -> k *randomisiert*

S(k,m) -> t mit $m \in \{0,1\}^n$, $t \in \{0,1\}^t$ $(n>>t)$

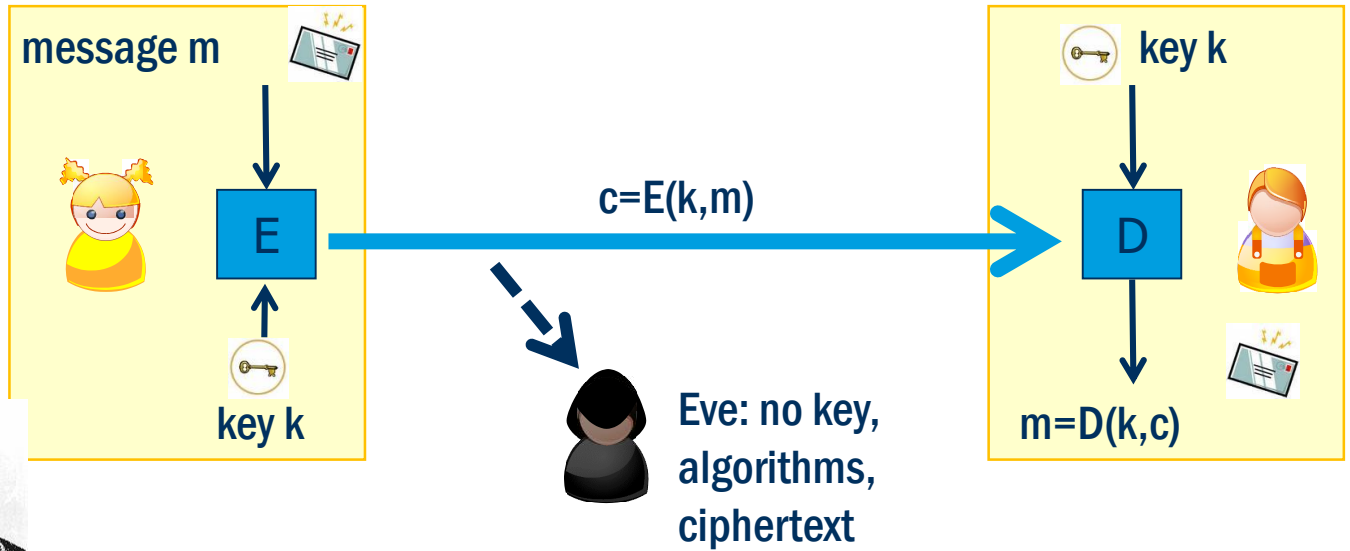
V(k,m,t) -> {0,1} *beide: deterministisch*

Kerckhoffs Prinzip (*wie machen wir's sicher?*)

KGen, Enc, und Dec werden dem Angreifer bekannt

→ Alle Algorithmen sollen öffentlich sein!

→ Sicherheit darf nur am Schlüssel hängen (geheim, unvorhersagbar)



"The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience."

Kerckhoffs: La Cryptographie Militaire, 1883

Was bedeutet „sicher“?

Was beobachtet und weiß der Angreifer?

Was darf er (nicht) dazulernen/erreichen?

Vertraulichkeit:

- Er soll nicht den kompletten Klartext lernen *(??)*
- Er soll nicht den Schlüssel extrahieren können *(??)*
- *→ Darf nichts lernen, was er nicht schon weiß!*

Integrität:

- Er soll keinen lesbaren Text verändern können („0“ einfügen) *(??)*
- *→ Darf kein valides Nachrichten/Tag-Tupel generieren können!*

Formalisierung / Worst-Case

Analyse/Diskussion bedarf Formalisierung von „Sicherheit“ (Ver./Int.)

Für Sicherheit: Analysiere Worst-Case

- Eingaben, bei denen Angreifer am besten lernt/fälschen kann
- → Wenn der Angreifer hier nichts lernt, dann auch sonst nicht!

Angenommen, Algorithmen schützen bestimmte Eingaben schlechter

- Angreifer würde diese Eingaben wählen, wenn er könnte

Rational der Analyse

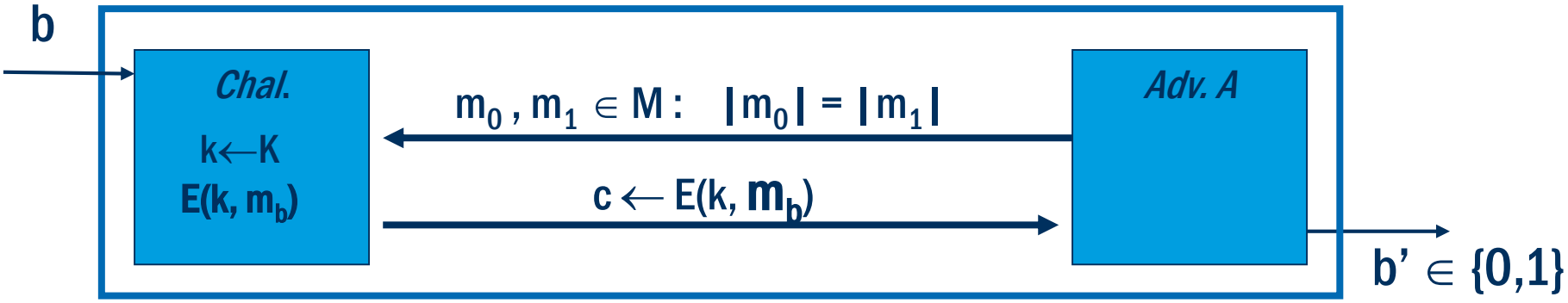
- Um Worst-Case zu berücksichtigen, lassen wir Angreifer alle Eingaben frei wählen
- *Was soll er dann lernen? Welches Tupel generieren?*

Formalisierung von Sicherheit als Spiel

„Indistuingishability“ bzw „Ununterscheidbarkeit“, in Variationen *IND-CPA/IND-CCA*

Das Vertraulichkeits-Spiel mit 2 Parteien:

- Herausforderer (C) wirft Münze
- Angreifer (A) kommuniziert mit (C) und muss Ergebnis erraten:



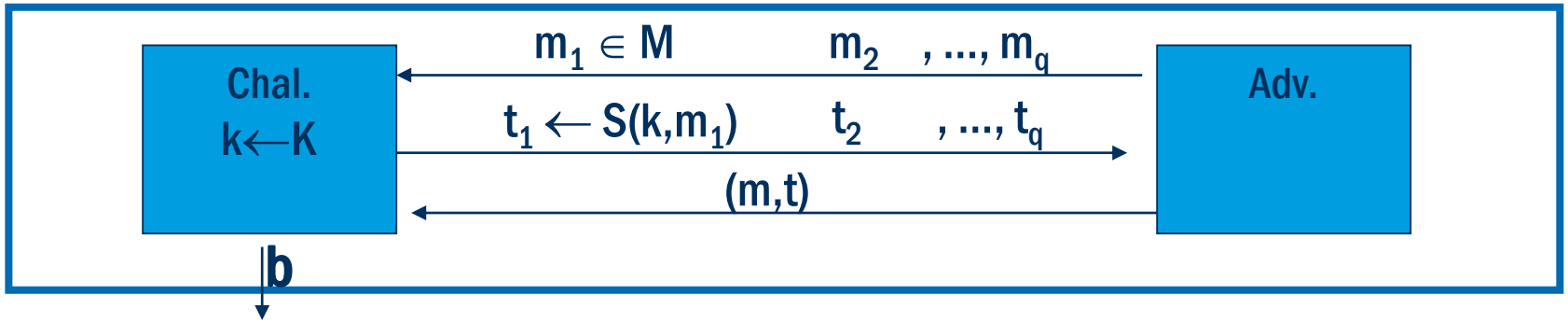
- Mit welcher Häufigkeit gewinnt der Angreifer?

$$Adv_{Conf}[A, E] := |\Pr[b' = b] - \Pr[b' \neq b]| \quad \in [0,1]$$

Formalisierung von Sicherheit als Spiel

Das Integritäts-Spiel mit 2 Parteien, I = (S,V):

- C produziert auf Anfrage valide Tupel (m,t)
- Angreifer kommuniziert mit (C), muss ein valides Tupel generieren:



b=1 wenn V(k,m,t) = `1' und $(m,t) \notin \{(m_1,t_1), \ldots, (m_q,t_q)\}$

b=0 sonst

- Mit welcher Häufigkeit gewinnt der Angreifer?

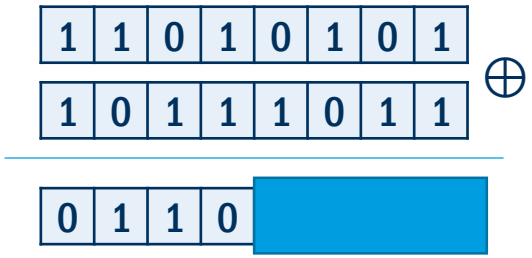
$$\mathrm{Adv}_{\mathrm{MAC}}[A,I] = \Pr[\text{Chal. outputs 1}] \leq \varepsilon$$

Probieren wir es aus!

Intermezzo: XOR

XOR zweier Zeichenketten in $\{0,1\}^n$ ist ihre bitweise Addition mod 2:

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0



Theorem:

Sei Y eine Zufallsvariable über $\{0,1\}^n$

Sei X eine unabhängige, gleichverteilte Zufallsvarible über $\{0,1\}^n$

Dann ist $Z := Y \oplus X$ gleichverteilt über $\{0,1\}^n$:

(Beweis an der Tafel)

The cryptographer‘s workhorse: XOR

PRIVACY AND SECURITY

X	Pr
0	p_0
1	p_1

Y	Pr
0	1/2
1	1/2

x	y	Pr
0	0	$p_0/2$
0	1	$p_0/2$
1	0	$p_1/2$
1	1	$p_1/2$

Pr[Z=0]

= Pr[(x,y)=(0,0) or (x,y)=(1,1)]

= Pr[(x,y)=(0,0)] + Pr[(x,y)=(1,1)]

= $p_0/2 + p_1/2$

= 1/2

Definieren wir zwei Chiffren

Sei $m = m_0 \ldots m_{n-1}$ und

$k = k_0 \ldots k_{l-1}$ eine Zeichenkette aus l gleichverteilt zufälligen Bits, mit l=n

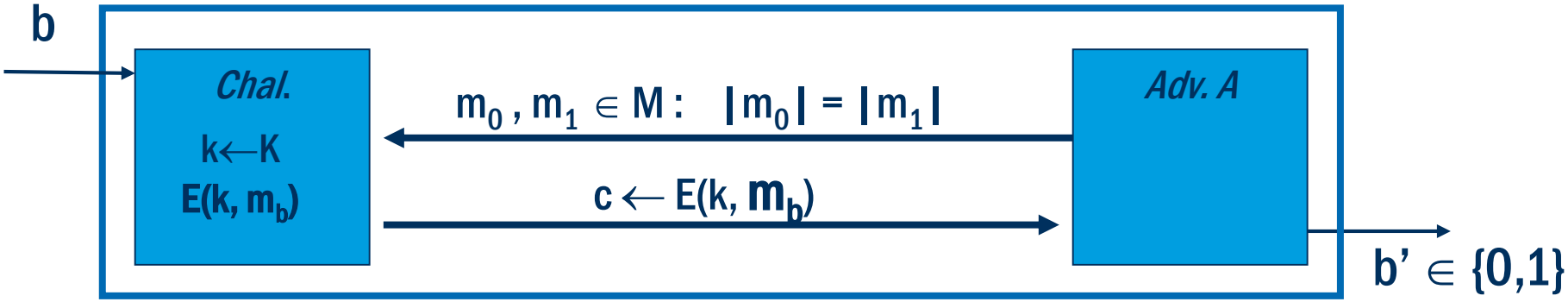
$\mathrm{Enc}_1(k,m) := c_0 \,..\, C_{n-1}$ mit $c_i = m_i \oplus m_{i+1}$ und $m_n = m_0$

$\mathrm{Enc}_2(k,m) := c_0 \,..\, C_{n-1}$ mit $c_i = m_i \oplus k_i$

Analyse der Chiffre Enc_1

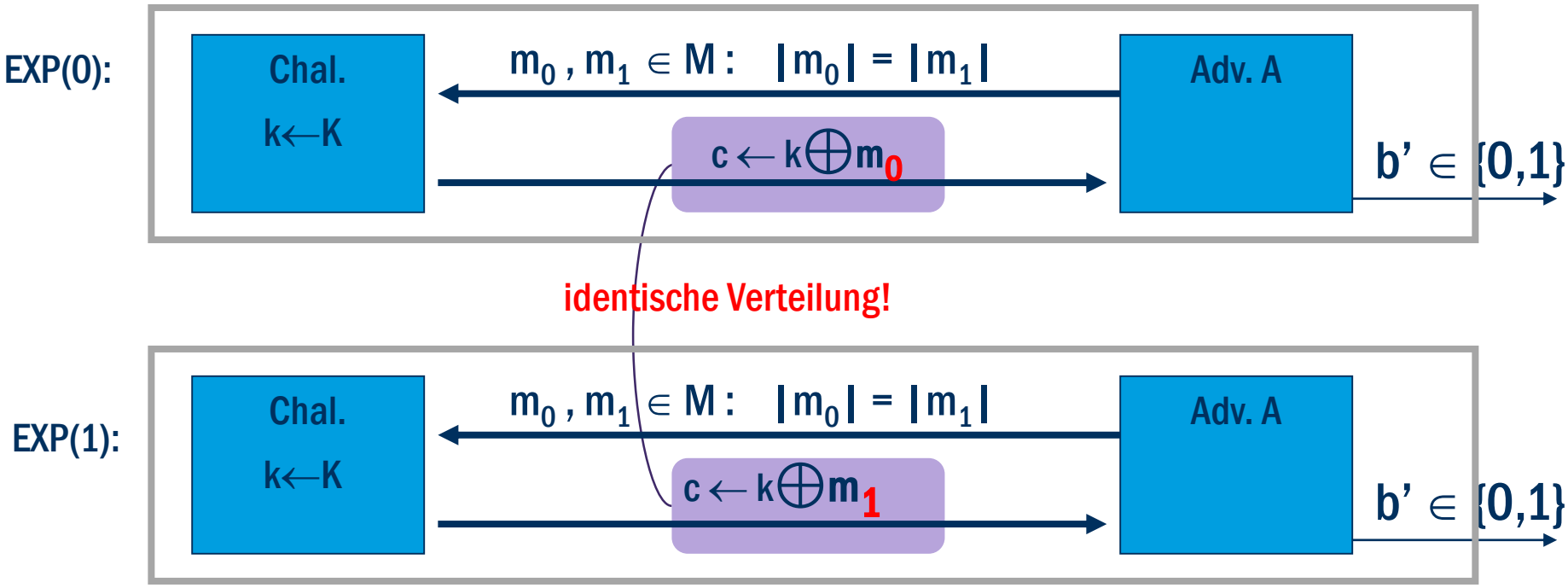
$Enc_1(k,m) := c_i = m_i \oplus m_{i+1}$

welche Nachrichten wählt der Angreifer?



$$Adv_{Conf}[A, \mathrm{Enc}_1] := |\Pr[b' = b] - \Pr[b' \neq b]| \quad \in [0,1]$$

Analyse der Chiffre Enc_2 (One Time Pad)



Für **alle** A: $\mathrm{Adv}_{\mathrm{Conf}}[A, \mathrm{Enc}_2] = \left| \Pr[A(k \oplus m_0) = 1] - \Pr[A(k \oplus m_1) = 1] \right| =$ 

Das One Time Pad (Vernam Cipher)

Gilbert Vernam
(1890-1960)

Grundkonzept:

- Schlüssel: Zeichenkette, so lang wie die Nachricht
- Wähle Schlüsselbits echt zufällig (keine erkennbaren Muster)

$\mathrm{Enc}(k, m) = c_0 \ldots c_{n-1}$ mit $c_i = f(k_i, m_i)$ („+ mod |Alphabet|“)

A	T	T	A	C	K	T	H	E	C	I	T	Y	A	T	T	W	E	L	V	E
P	S	P	I	U	H	G	D	S	P	H	G	D	S	P	I	W	E	E	W	O

(+ mod 26)

P	L	I	I	W	R	Z	K	W	R	P	Z	B	S	I	B	S	I	P	R	S

Y	H	P	R	S	R	G	F	F	D	D	X	N	S	Q	I	S	P	W	N	F
R	E	T	R	E	A	T	F	R	O	M	C	O	A	S	T	A	T	T	E	N

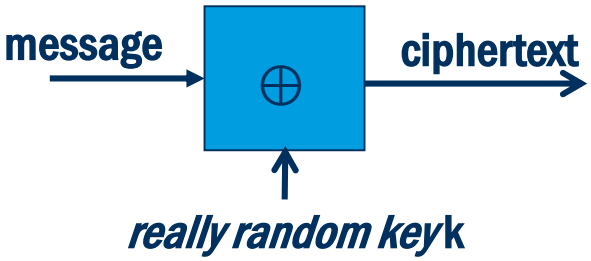
(+ mod 26)

Angreifer kennt Länge, Chiffrat, keinen Klartext – lernt nichts!

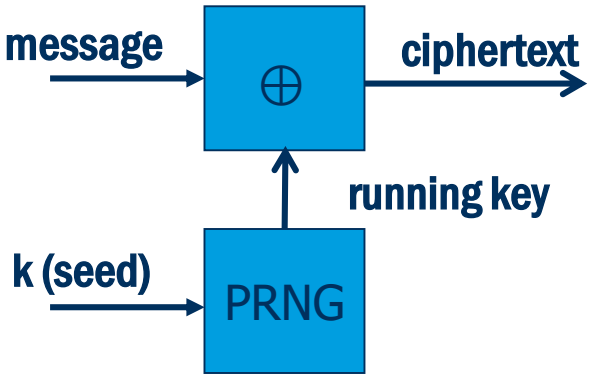
Was ist das praktische Problem?

Eine praktikablere Chiffre Enc_3

OTP:



Enc_3:



Idee: Benutze „pseudozufällige“ Sequenz als Schlüsselbits

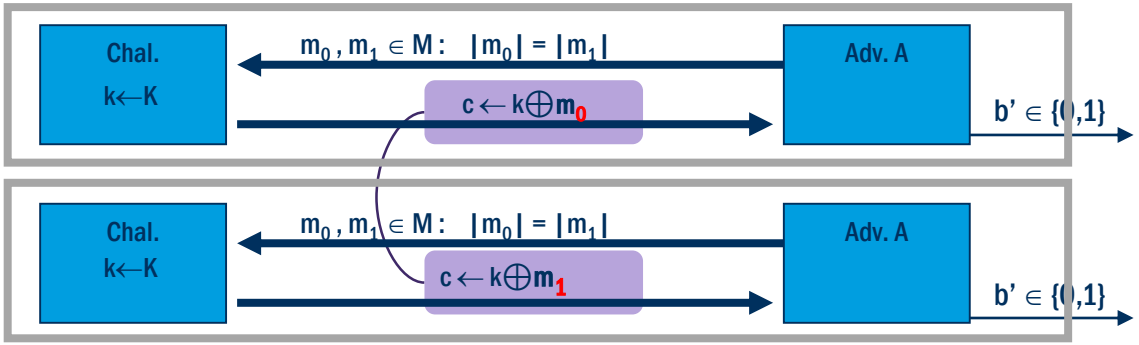
PRNG ist eine Funktion $G: \{0,1\}^s \rightarrow \{0,1\}^l$ $l >> s$

Det. Algorithmus von Seed- in Schlüsselraum (scheinbar zufällig)

$k' = PRNG(k)$

$Enc_3(k,m) := c_0 .. c_{n-1}$ mit $c_i = m_i \oplus k'_i$

Enc_2 vs Enc_3



Sei PRNG ein „sicherer“ (unprädizierbarer) PRNG,

→ Verteilungen in EXP(0) und EXP(1) sind nahezu identisch

Annahme über den Angreifer (Angreifermodell #3, Ressourcen)

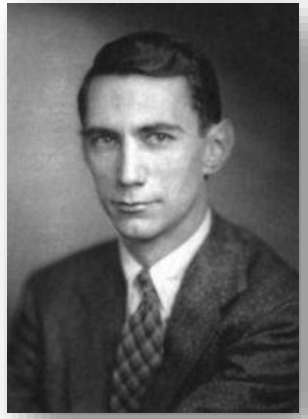
Zeitlich unbeschr. (theor.) Angreifer vs „effizienter“ (PPT) Angreifer

1. Prüft in Enc_3 alle 2^s möglichen Schlüssel und erreicht b‘=b
2. Kann nicht exponentiell viele Schlüssel prüfen (poly. beschränkt)

Wir nennen Enc_2 “***informationstheoretisch sicher***“ (perfect secrecy)

Wir nennen Enc_3 (bestenfalls) “***semantisch sicher***“

Informationstheoretische Sicherheit



Shannon (1949):

„CT darf ***keine*** Information über PT verraten“

Def: Eine Chiffre (E,D) über ($\mathcal{K}$, $\mathcal{M}$, $\mathcal{C}$) hat ***perfekte Sicherheit*** wenn

$\forall\ m_0, m_1 \in \mathcal{M}$ (mit $len(m_0) = len(m_1)$)

$\forall\ c \in \mathcal{C}$ und $k \xleftarrow{R} \mathcal{K}$:

$$Pr[E(k,m_0) = c] = Pr[E(k,m_1) = c]$$

Was lerne ich als Angreifer?

Kein CT-Angriff kann entschieden ob m m_0, m_1 (oder beliebiges m) ist

→ Keine „Ciphertext Only“ (CTO) Angriffe möglich!

Beweis perfekter Sicherheit für das OTP

Wir nehmen an: $\forall$ m, c: $\Pr_k[E(k,m) = c] = \frac{\#keys\ k \in \mathcal{K}\ s.t.\ E(k,m)=c}{|\mathcal{K}|}$

Wenn: $\forall$ m, c $\#\{ k \in \mathcal{K} : E(k,m) = c\}$ konstant ist,

dann ist Chiffre informationstheoretisch sicher

(beobachtetes c, egal welches m: Wahrscheinlichkeit ist identisch!)

Für OTP: c = m $\oplus$ k also k = m $\oplus$ c

$\#\{ k \in \mathcal{K} : E(k,m) = c\}$ = 1

➔ One Time Pad ist informationstheoretisch sicher!

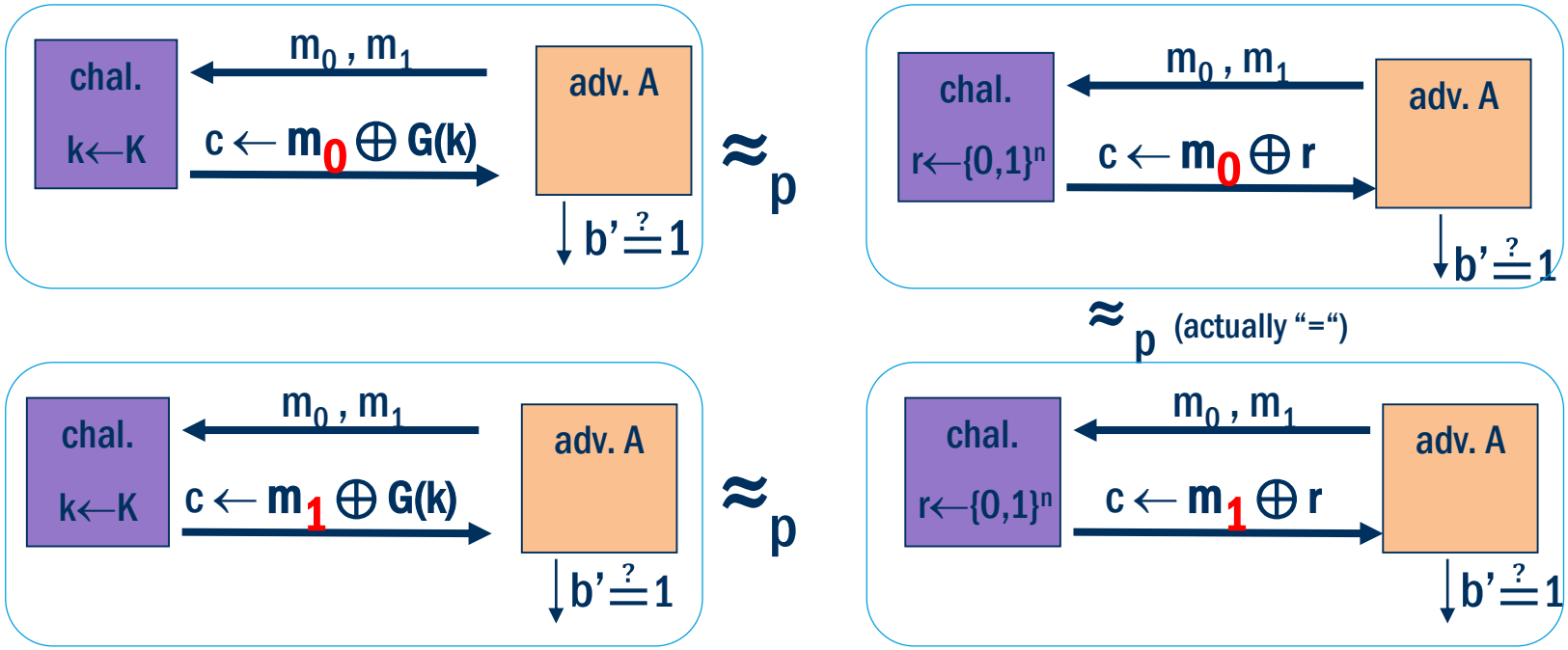
Ist unsere Stromchiffre semantisch sicher?

Nehmen wir an, unser PRNG sei durch effizienten Angreifer nicht von „echtem“ Zufall zu untescheiden.

Wir haben gezeigt:

- OTP (XOR mit echt zufälligem Schlüssel) ist sicher

Intuition für den Beweis:



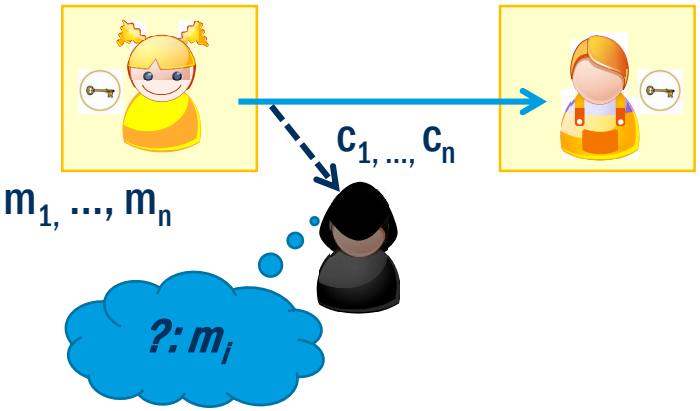
Security Notions, Variationen des Spiels

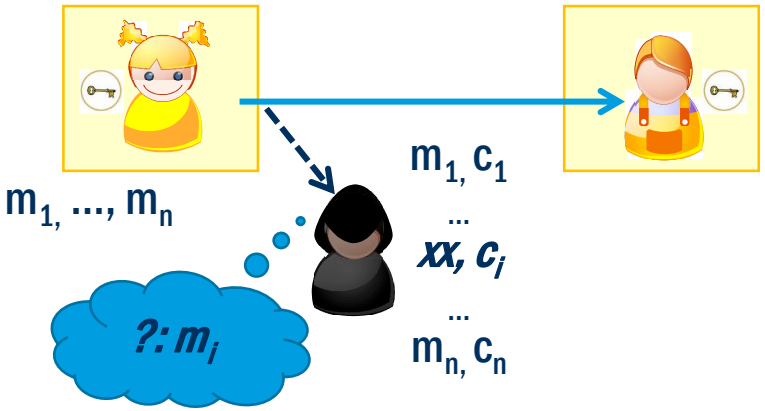
Ciphertext-only attack:

- despite concealed key
- using ciphertext only
- learn about plaintext (or key)

Known-plaintext attack:

- despite concealed key
- Knowing some plaintexts
- Learn about plaintext (or key)





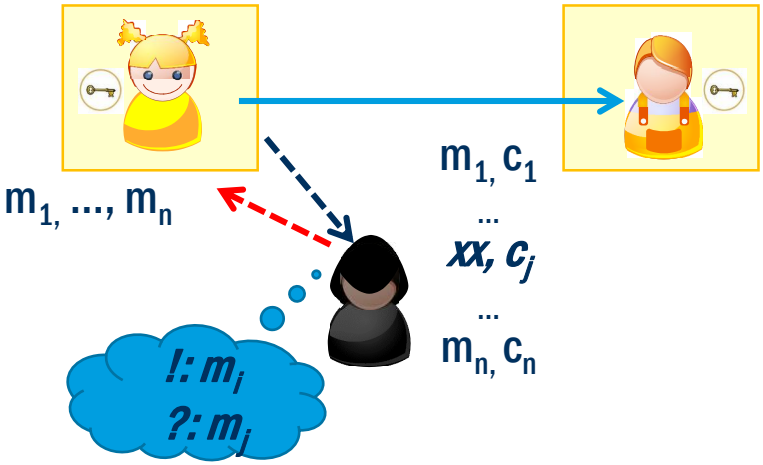
- *Schwächster Angreifer!*

Security Notions ctd.

PRIVACY AND SECURITY

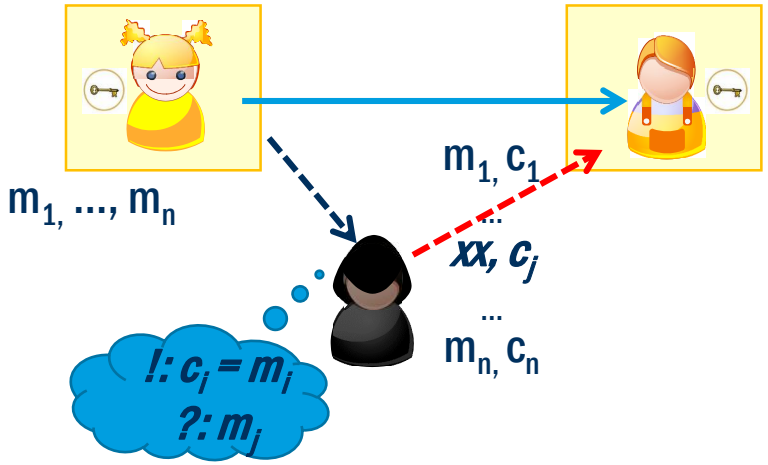
Chosen-plaintext attack:

- despite concealed key
- asking Alice to encrypt m_a
- learn about m_j (or key)



Chosen-ciphertext attack:

- despite concealed key
- asking Bob to decrypt c_a
- learn about m_j (or key)



*Stärkster Angreifer, **realistisch**!*

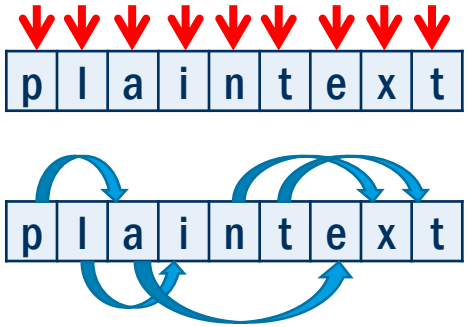
Asymmetric crypto: non-modifying („passive“) attack...

Konstruktion von Chiffren

PRIVACY AND SECURITY

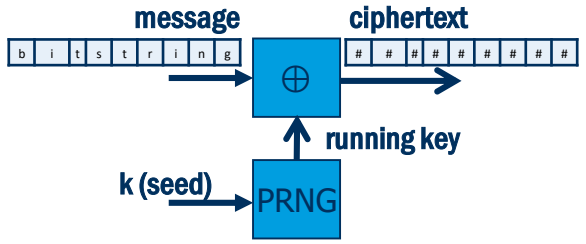
Operationen zur Verschlüsselung

- *Substitution* ersetze Symbole durch andere
- *Transposition* permutiere Symbole nach Schema

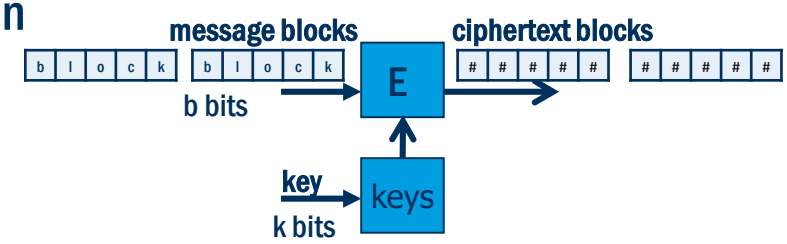


Verarbeitung der Klartexte

- *Stromchiffren* Generiere Schlüsselbits und subst.



- *Blockchiffren* Pseudozufallspermutationen



Konstruktion von Pseudozufallspermutation

PRIVACY AND SECURITY

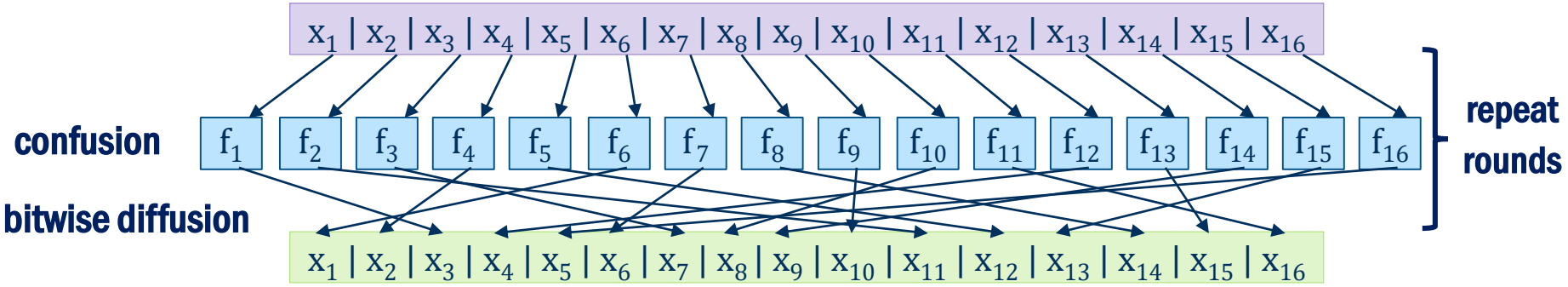
Shannon postuliert ***Confusion-Diffusion Paradigma***

- Verhältnis zwischen Schlüssel und Chiffrat muss versteckt werden
- Strukturen im Klartext müssen versteckt werden

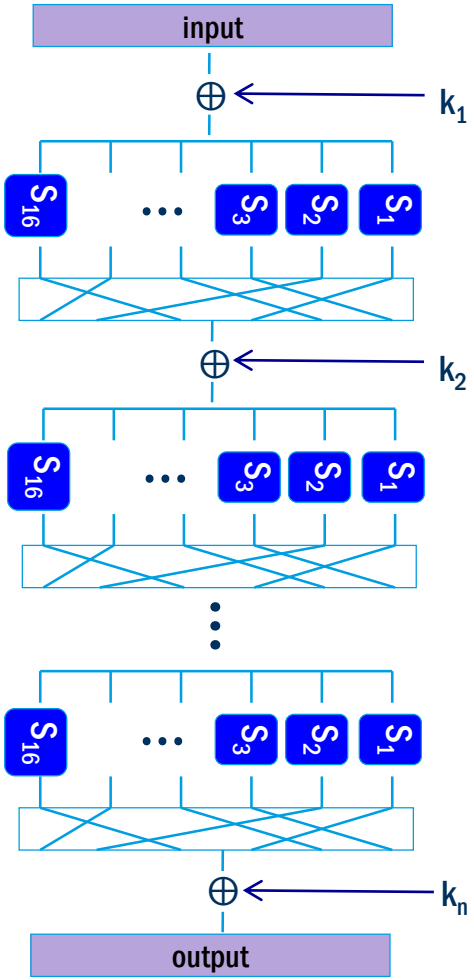
Wunsch: ***Lawinen- (Avalanche) Effekt***: Bitkippen im Input führt zu Änderung von *jedem* Output-Bit mit jeweils Wahrscheinlichkeit .5

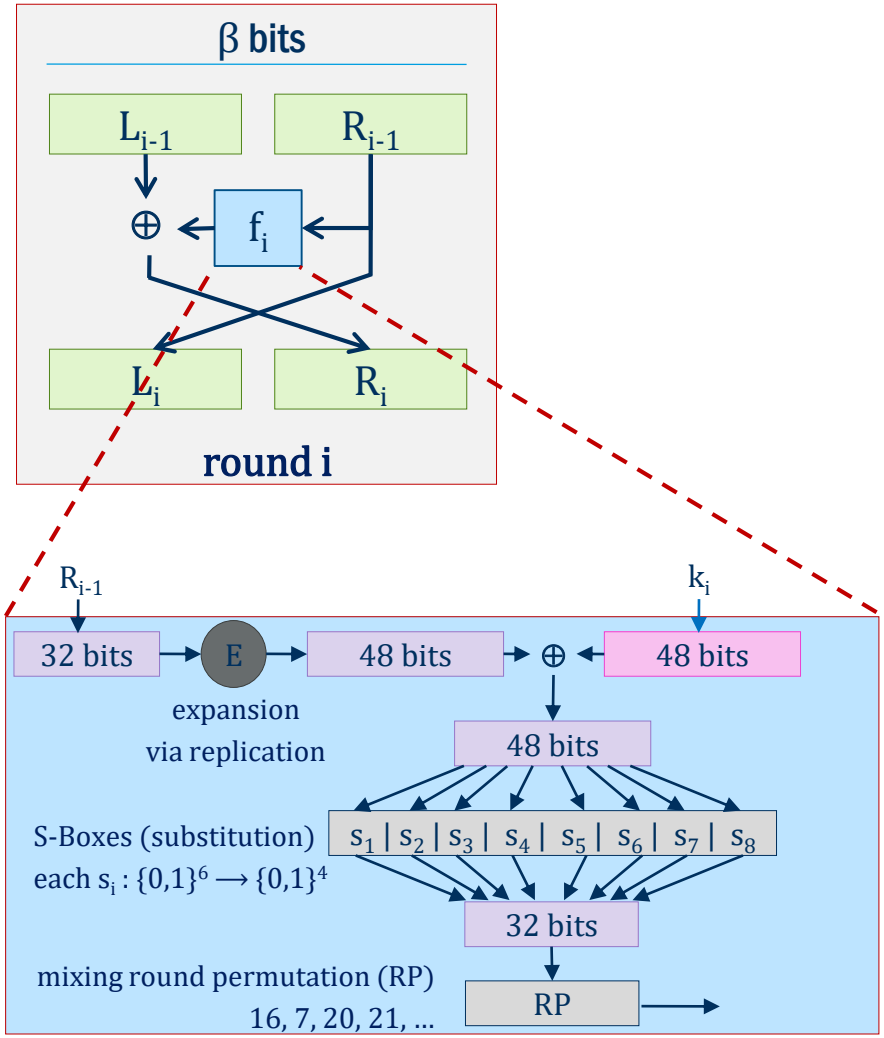
Konstruktion von ***Substitutions-Permutations-Netzen***:

- Kombination „kleiner“ Zufallspermutationen („S-Boxen“)
- Anschließend bitweise Tansposition
- Rinse and Repeat (generiere Rundenschlüssel, $\oplus$ mit Input)



Beispiel PRPs: AES und Feistel-Netze (DES)

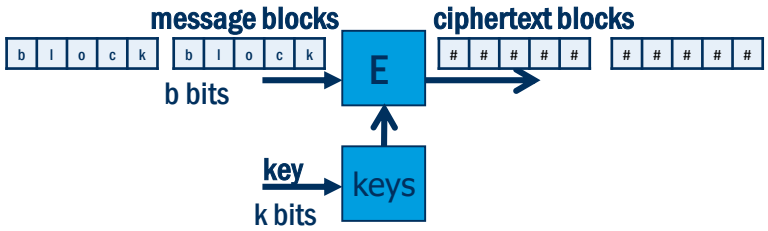




Von PRPs zu Block-Chiffren (Operationsmodi)

Soweit haben wir PRPs mit fester Eingabegröße (64 oder 128 bits)

Die meisten Nachrichten im richtigen Leben sind länger…

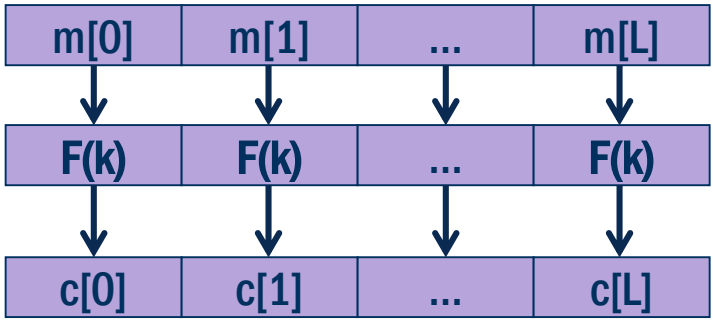


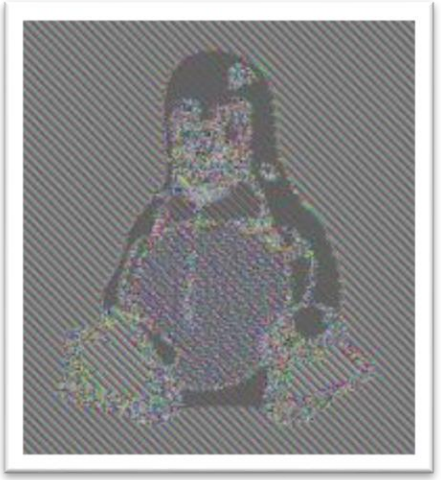
Wie verschlüsseln wir längere Nachrichten sicher mit PRPs?

Electronic Code Book Mode

PRIVACY AND SECURITY

Verschlüssle jeden Block unabhängig mit PRP:





ECB- Verschlüsselung ist *deterministisch*

⇒ gleiche PT-Blöcke → gleiche CT-Blöcke

Ist das "sicher" (wie)?

Sichere Operationsmodi

ECB einer deterministischen PRP ist unsicher:

- Zwei identische PT mit gleichem Schlüssel ergeben gleichen CT
- Zwei identische PT-Blöcke ergeben identische CT-Blöcke

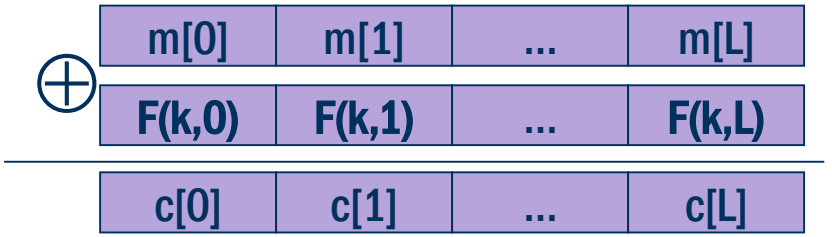
Was können wir hier tun?

- One-time key (internal): verschlüssele jeden Block unterschiedlich
- Many-time key (external): Füge Zufall hinzu

Counter Modes und Chaining

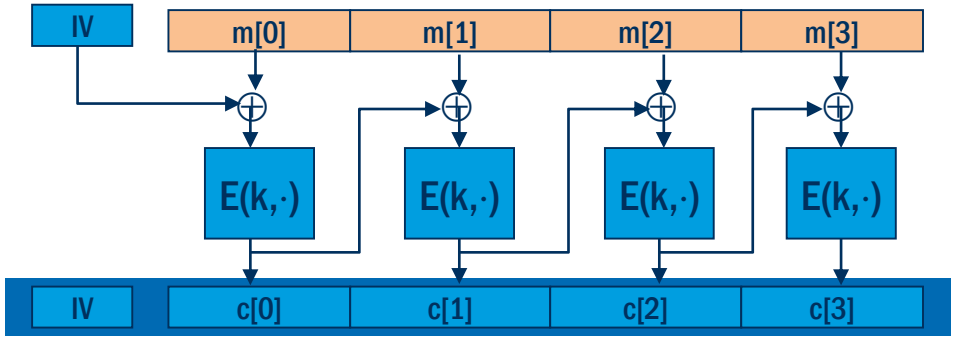
(1) Verschlüssle jeden Block unterschiedlich:

- Integriere (variierenden) Wert in die Verschlüsselung der Blöcke
  - Nonces: $c_i = E(k,n_i,m_i) = E(k, (n_i,m_i))$ or $E((k,n_i),m_i)$?
  - ...und Übertragung aller n_i?
  - Zähler to the rescue!



(2) Many-time key (Initialisiere mit Zufall):

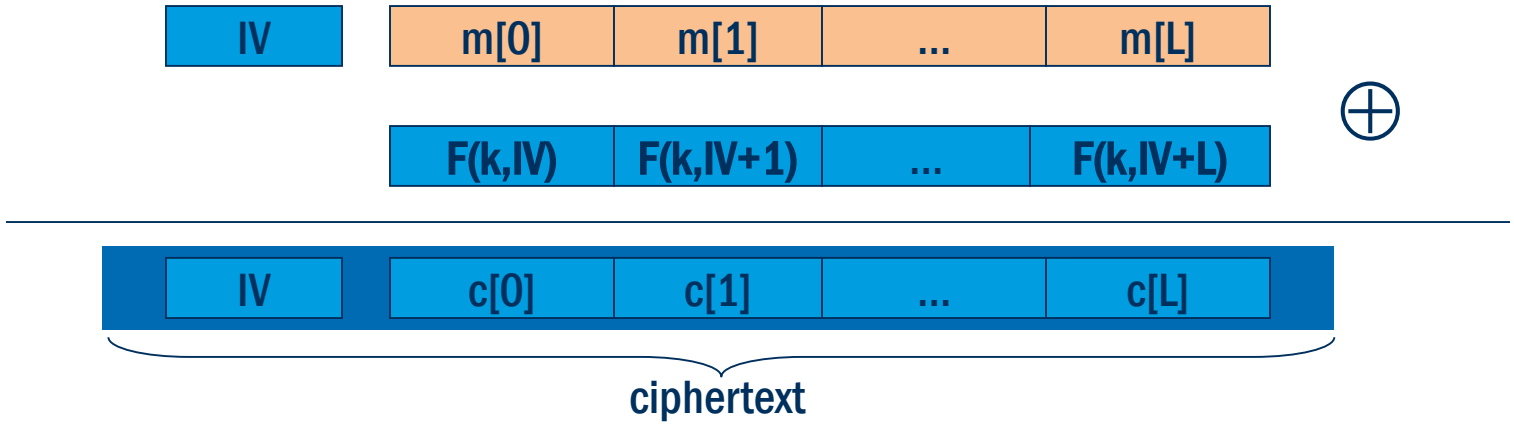
- Integriere (variierenden) Wert in Verschlüsselung jeder Nachricht!
  - Unabhängiger Zufall für *jeden Block* (wirklich nötig)?
  - Wähle zufälligen IV
  - Verkette in Verschlüsselung



Randomized Counter Mode R-CTR

Let $F: K \times \{0,1\}^n \longrightarrow \{0,1\}^n$ be a secure PRF.

E(k,m): choose a random $IV \in \{0,1\}^n$ and do:



Hinweise:

- E, D können parall. werden und F(k,IV+i) vorberechnet
- R-CTR erlaubt random access, jeder Block kann einzeln ver- und entschlüsselt werden
- F kann eine PRF sein, muss selbst nicht invertierbar sein

Zwischenergebnis

Angreifer / Angreifermodell

Bedrohungen

Sicherheitsziele (CIA)

Sicherheitsdienste

Definitionen und Formalisierung als Spiel

Perfekte Sicherheit / semantische Sicherheit

One-Time-Pad, Stream-Cipher

Block-Cipher und deren Operationsmodi

Konstruktion eines Integritäts-Dienstes

Ziel: Detektion unautorisierten Modifikation (*nichts zu verstecken*)

- Abbildung bel. langer Nachrichten auf Tag fixer Länge
- Tag authentisiert Absender und weist Manipulationsfreiheit nach
- → „Message Authentication Code“ (MAC)

Algorithmen

$$\text{KeyGen}() \rightarrow k \in K$$

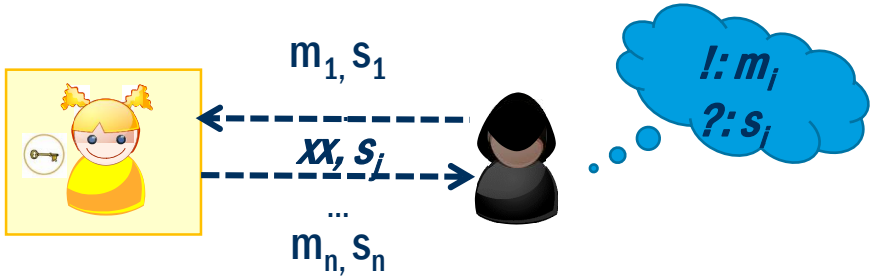
$$S(k,m) \rightarrow t \qquad \text{mit} \quad m \in \{0,1\}^n,\ t \in \{0,1\}^t \qquad (n >> t)$$

$$V(k,m,t) \rightarrow \{0,1\}$$

Security Notions: Existential Forgery

Chosen Message Attack:

- gegeben $s_1, s_2, \ldots s_n$ für gew. m_i



(Variationen: known verification key / known signature attacks)

Existential Forgery:

Produziere bel. neues valides Tupel (m,t) (bel. Nachricht, ggf Nonsense)

⇒ Angreifer kann kein valides Tag für neue Nachricht generieren

⇒ Angreifer kann nicht mal (m,t'), (m',t) für (m,t) und t' ≠ t oder m ≠ m'

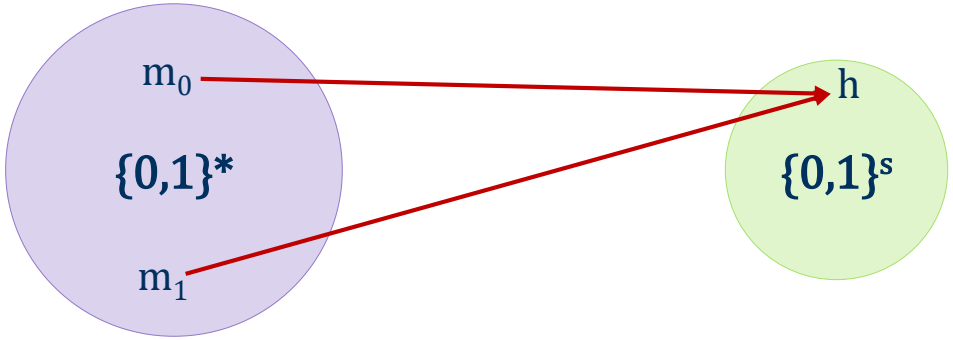
Generelle Notions:

Exist. forgery < selective forgery < universal forgery < total break

Konstruktion von MACs

Ideen:

- Authentisierung des Absenders → Geheimnis
- Abbildung auf Fingerabdruck fixer Länge → Hashfunktionen

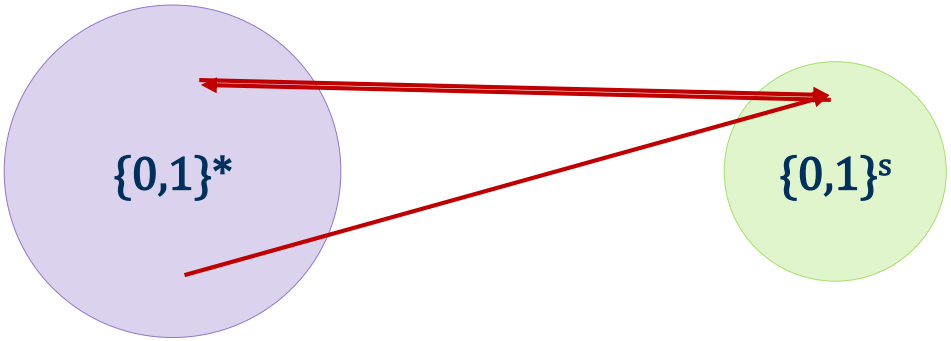


Geheimnis bleibt geheim (keine Verschlüsselung) → Einwegfunktionen

Effekt kleinster Änderungen an m nicht prädizierbar in t → „Chaos“

Anforderungen an *sichere* Hash/MACs

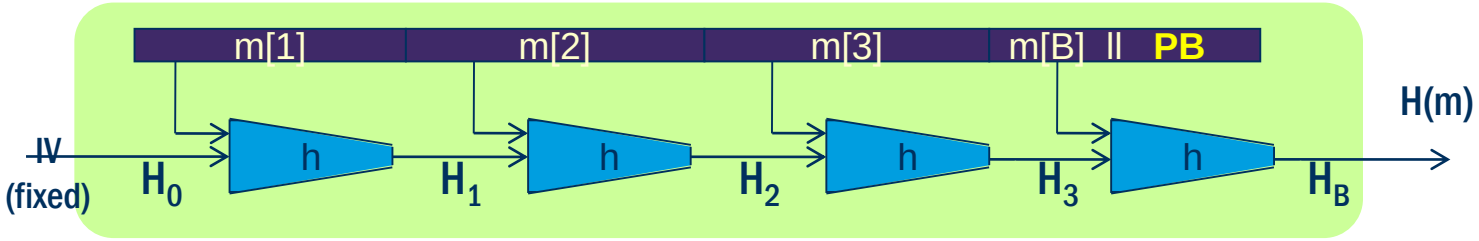
1. Einwegfunktion (preimage resistance)
2. Starke Kollisionsresistenz (collision resistance)
3. Schwache Kollisionsresistenz (2nd preimage resistance)



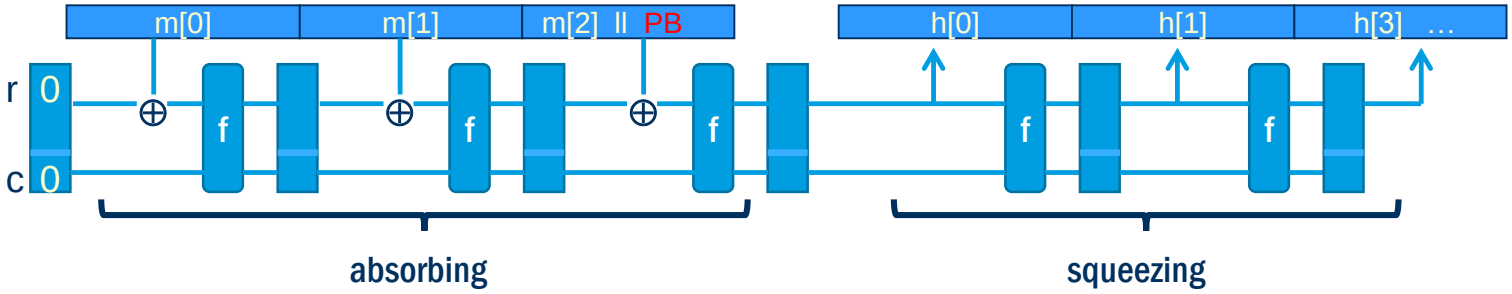
*Warum nicht kollisions**freie** Hash-Funktionen?*

Zwei sichere Hash-Funktionen

Merkle-Damgård: Kette von PRP ohne Verschlüsselung (MD5/SHA1..)



SHA-3: Verrechnen der Nachricht zu pseudozufälligem Zustand



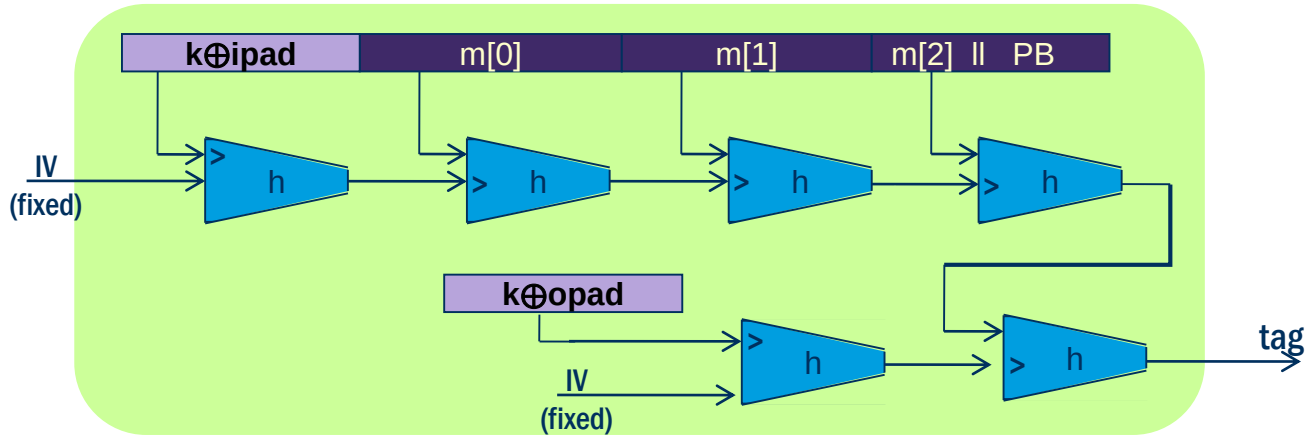
Konstruktion von MACs aus Hash-Funktionen

Sicherheitsziele von Hash-Funktionen (Einweg/keine Kollisionen)

Sicherheitsziele von MACs (kein Existential Forgery)

MAC:= Enc(k,h(m))?

HMAC:



Zusammenfassung

Bedrohungen für Kommunikation und Angreifermodelle

Sicherheitsziele

- **C**onfidentiality (Vertraulichkeit)
- **I**ntegrity (Integrität)
- **A**vailability (Verfügbarkeit)

Definition und Design von Sicherheitsmechanismen und Algorithmen

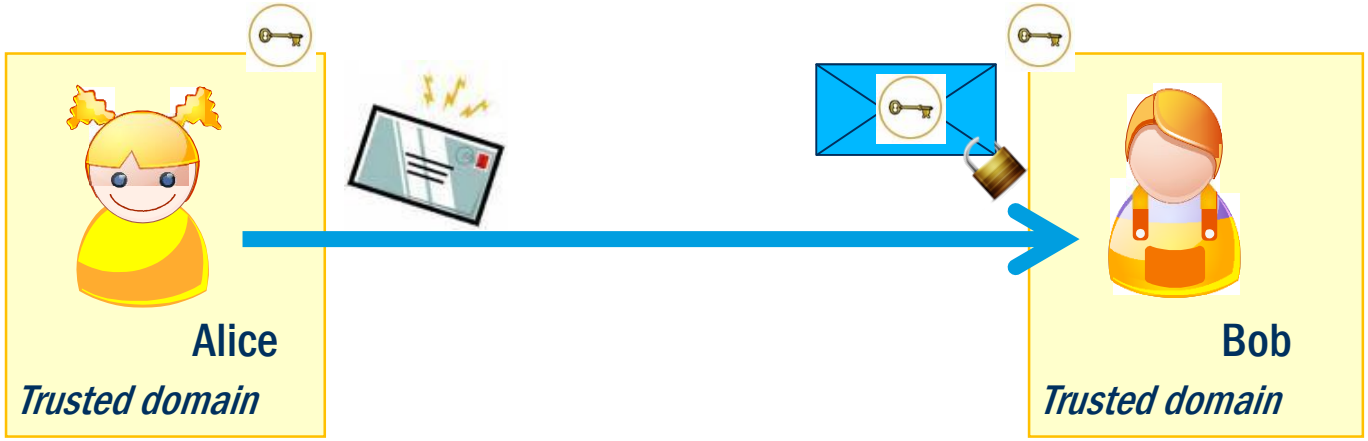
Spiel-basierte Analyse von Verschlüsselung und Signierung

Perfekte Sicherheit und semantische Sicherheit

Design von Stromchiffren (OTP) und Block-Chiffren/Operationsmodi

Design von Hash-Funktionen und Message Authentication Codes (MACs)

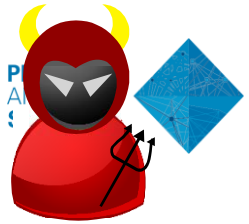
Schlüsselverteilungsproblem



Ansätze:

- Alice/Bob einigen sich direkt
  - Im Vorwege
  - Zum Verbindungsaufbau
- Alice und Bob delegieren Vertrauen an dritte Instanz (*„TTP“, „KDC“*)

Unsere Annahme: Der Dolev - Yao Angreifer



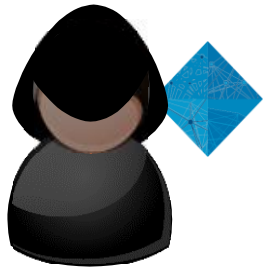
Mallory hat volle Kontrolle über den Kommunikationskanal, er kann:

- Nachrichten abhören („*eavesdropping*“, passiv)
- Auslieferung unterdrücken („*suppression*“)
- Nachrichten erneut ausliefern („*replay*“)
- Nachrichten verändern („*manipulation*“ / „*tampering*“)
- Nachrichten fälschen („*forgery*“)
- Nachrichten umleiten („*relay*“)
- Nachrichten vertauschen

Aber:

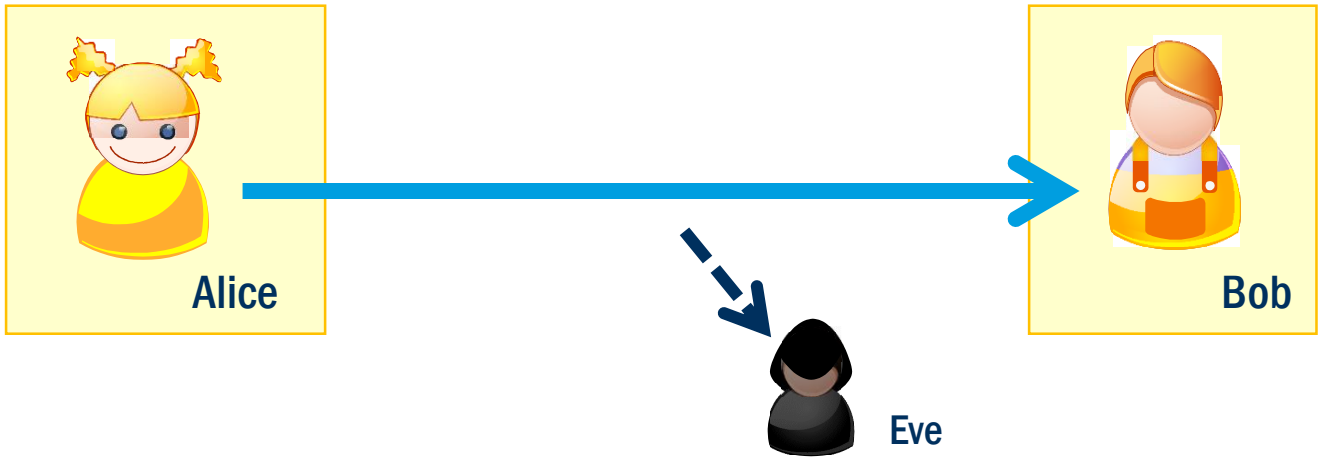
- Mallory kann „Krypto“ nicht brechen (*PPT Angreifer*)

Erster Versuch: Passive Angreifer



Eve kann:

- Nachrichten abhören („*eavesdropping*“, passiv)



- Können Alice und Bob sich sicher auf einen Schlüssel einigen?

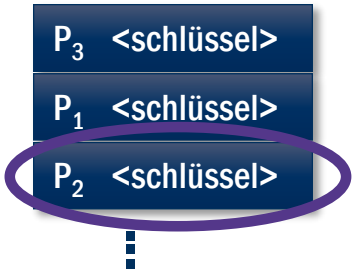
Schlüssel-Aushandlung

Ralph Merkle, Martin Hellman, Whitfield Diffie

Ziel:

Nachrichtenaustausch öffentlich

Im Ergebnis: geheimer Schlüssel



Ursprüngliche Idee (Merkle, '74):

- Alice erstellt 2^{32} Puzzles (mit Index P_i und Schlüssel)
- Alice mischt die Puzzle und sendet sie Bob
- Bob wählt und löst ein Puzzle zufällig
- Bob informiert Alice über den Index P_j, beide kennen den Schlüssel.

Wie schwer ist das Brechen für Eve? Quadratisch

Polynomieller Vorteil: Diffie-Hellman(-Merkle)

Können wir das Brechen erschweren?

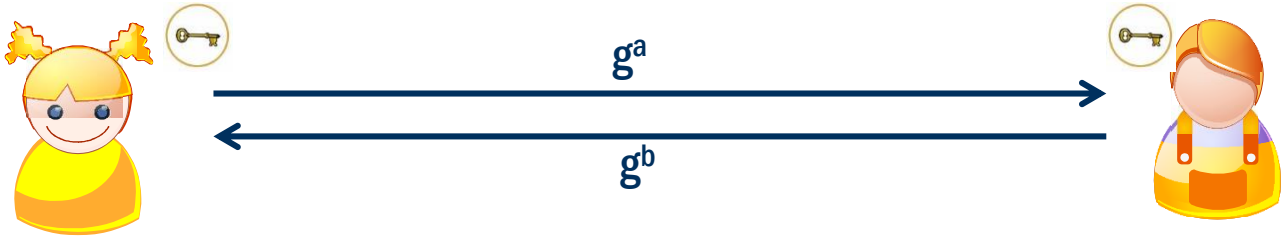
Beobachtungen:

1. Diskreter Logarithmus (einige Umkehrfunktionen) schwer zu lösen
2. Potenzgesetze: $(g^x)^y = g^{xy} = g^{yx} = (g^y)^x$

Idee:

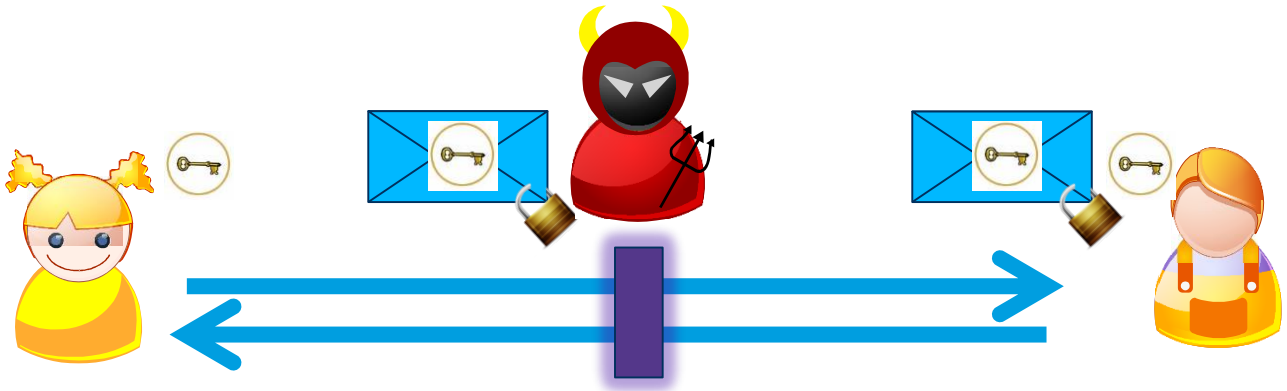
Wahl zyklischer Gruppe $\mathbb{Z}_p^*$, generiert durch g, und $\varphi(p) = p-1$

Alice wählt $a \xleftarrow{R} \{1,\ldots,(p-1)\}$, Bob wählt $b \xleftarrow{R} \{1,\ldots,(p-1)\}$



Alice berechnet: $(g^b)^a \bmod p = g^{ab}$ net: $(g^a)^b \bmod p$

Zwei Konkrete Angriffe



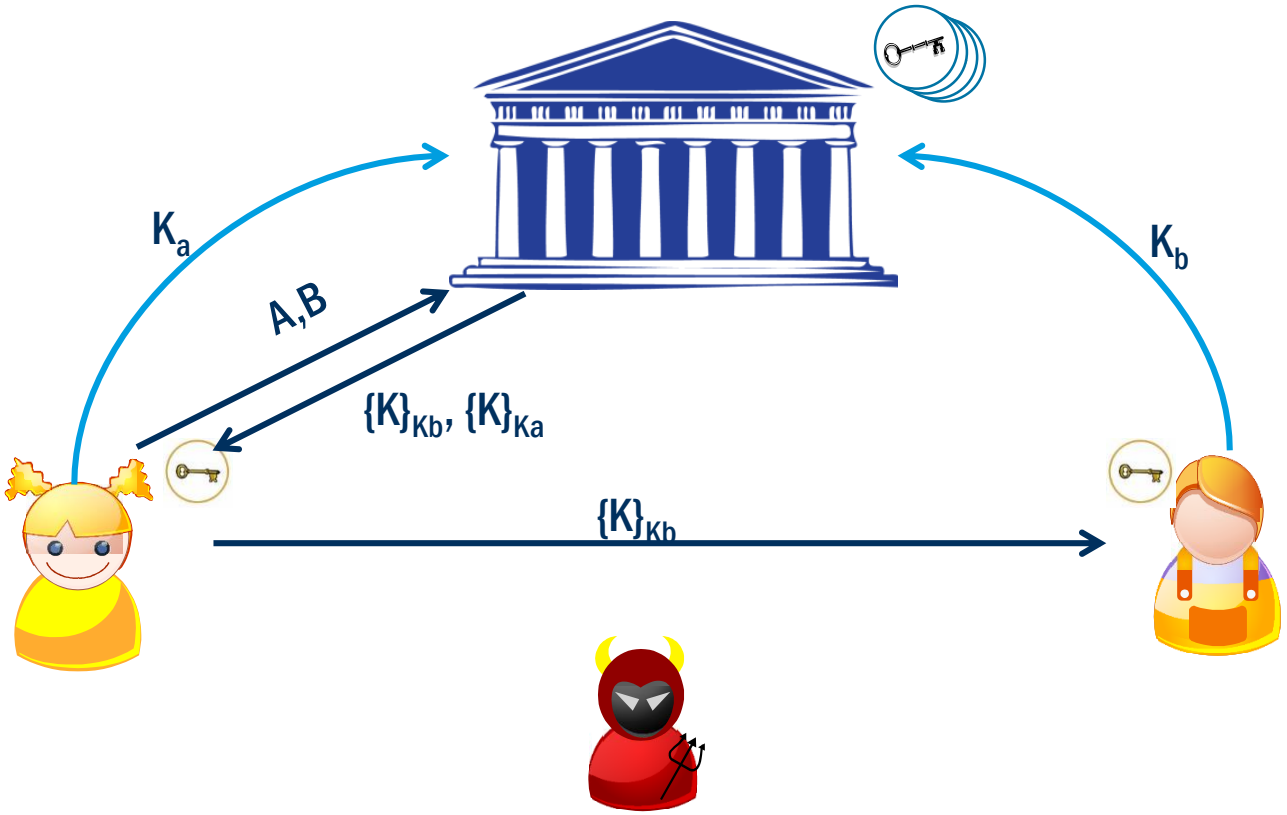
- Man-in-the-middle Angriff

Parteien proaktiv/extern identifizieren/ authentisieren!

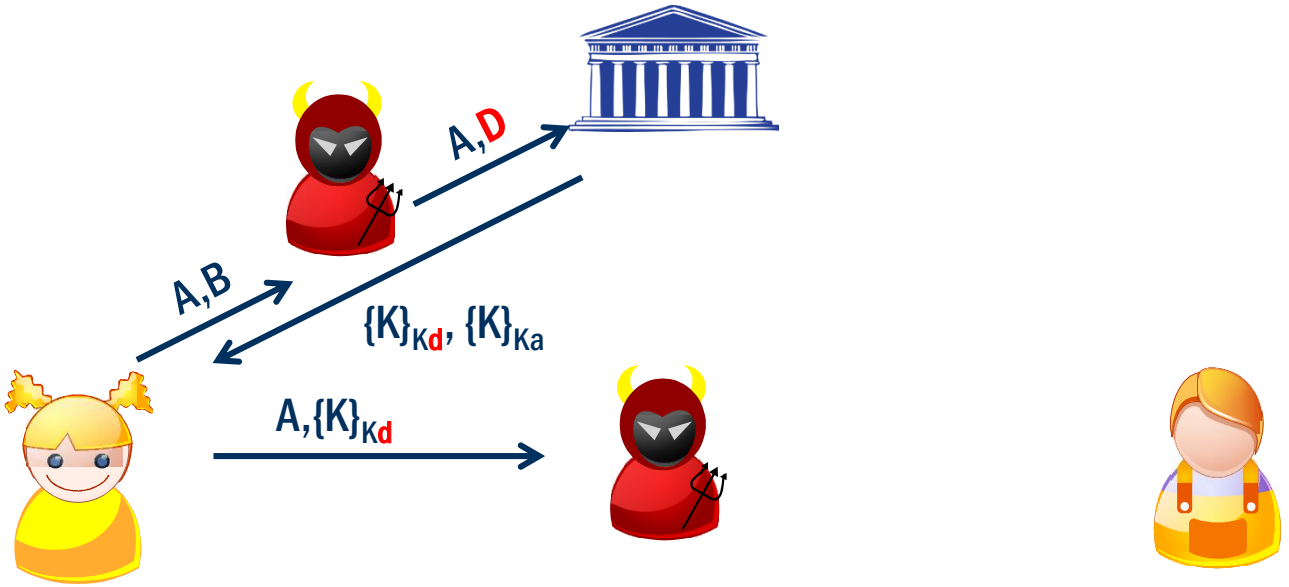
- Replay Angriff

Aktualität sicherstellen!

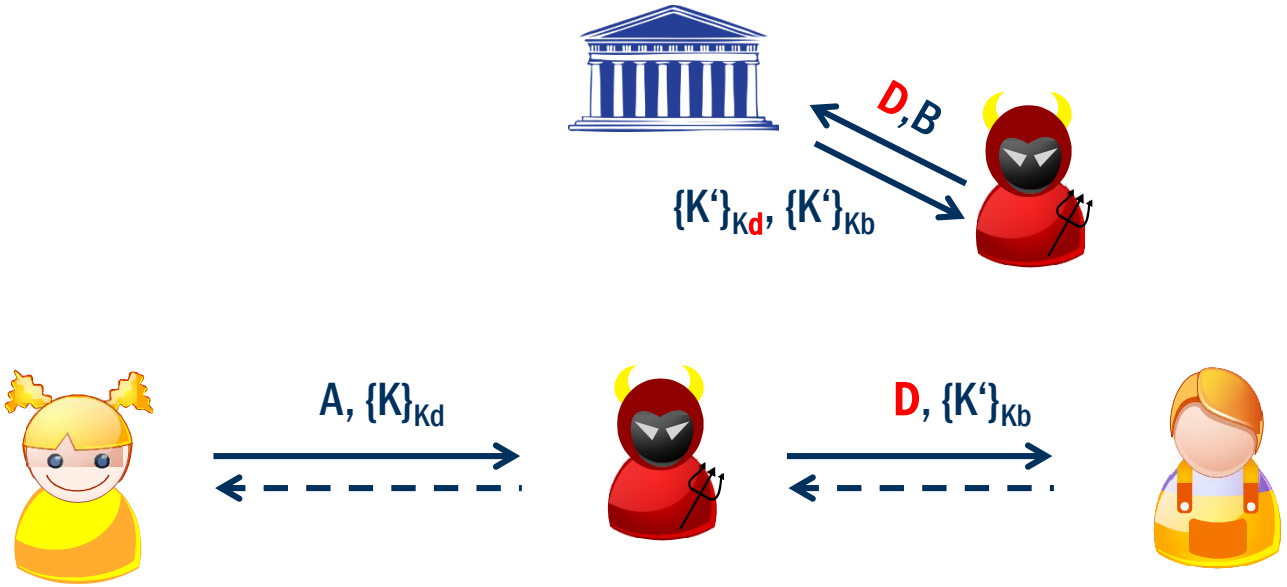
Ein einfaches (unsicheres) Protokoll



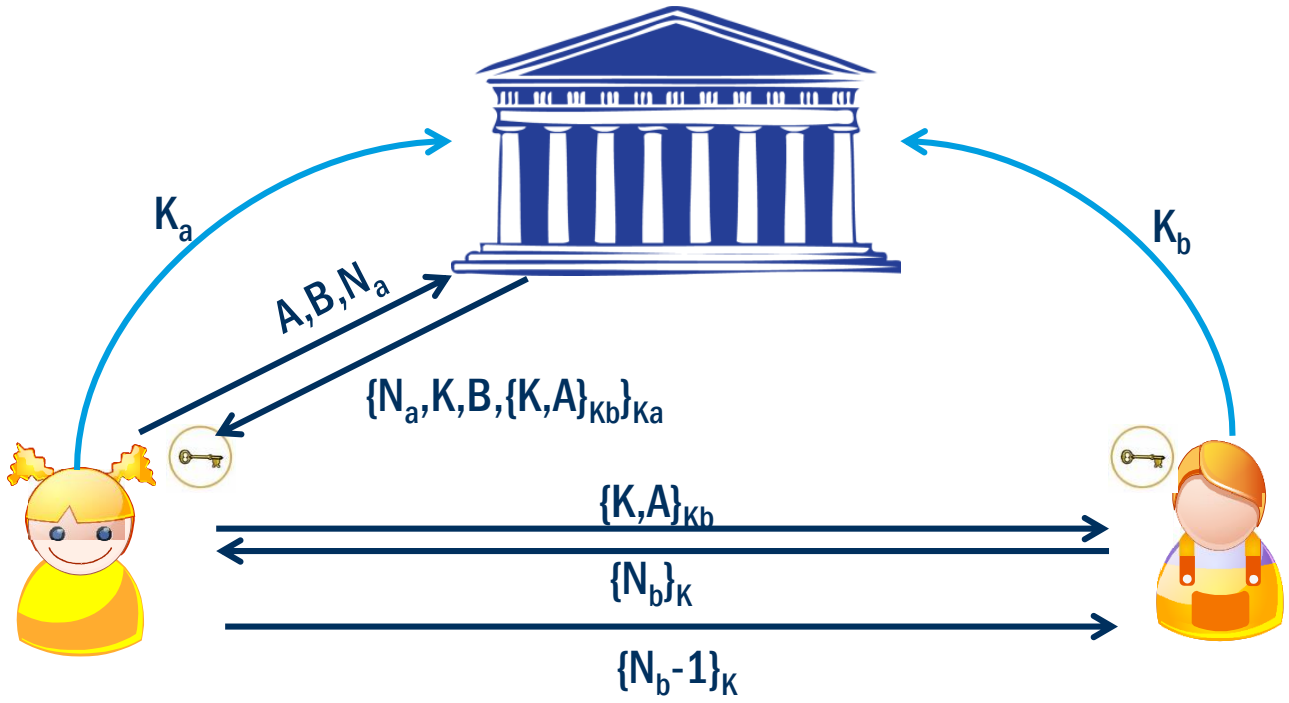
Man in the Middle - 1



Man in the Middle - 2



Schroeder-Needham Schlüsselaustausch



Erkenntnis:

- *MitM* durch *Identifikation* verhindert
- Alice und Bob durch Kenntnis von Schlüssel K authentisiert (brauchen K_a oder K_b)
- *Replay* durch „*Nonce*“ verhindert
- *Erweiterung: Zeitstempel für Aktualität, falls Mallory alten Schlüssel errät*

Asymmetrie (RSA) Mathematische Grundlagen

Bei freier Wahl großer Primzahlen p und q:

- Die Berechnung von $n = p \cdot q$ leicht
- Faktorisierung von n zu p und q schwierig

In mult. zyklischen Gruppen: Multiplikation trivial, Division hingegen…

Mit Wissen der Primfaktoren und erweitertem euklidischen Algorithmus ist einfach zu berechnen:

Für multiplikative zykl. Gruppe Z_n^* und e (teilerfremd zu n), gilt

e^{-1} : $ggT(e,n) = 1 : d \cdot e + k \cdot n$

$k \cdot n \equiv 0 \bmod n$

$e \cdot e^{-1} \equiv 1 \bmod n$

=> $e^{-1} = d$

RSA - Schlüsselgenerierung

Jeder Teilnehmer

- wählt zufällig und unabhängig 2 verschiedene Primzahlen p, q ungefähr gleicher Länge
- berechnet $n = p \cdot q$ und $\varphi(n) = n\text{-}p\text{-}q+1 = (p\text{-}1)(q\text{-}1)$
- wählt zufällige Zahlen e,d mit $2 < e < \varphi(n)$, $ggT(e, \varphi(n)) = 1$
- Und $e \cdot d = 1 \bmod (\varphi(n))$ (mit erweitertem euklidischen Algorithmus)

Öffentlicher Schlüssel: (n, e)

Geheimer Schlüssel: (p, q, d)

Ver- und Entschlüsselung

Verschlüsselung:

Gegeben (N,e), RSA (m):

$= m^e \mod N$

Entschlüsselung:

Gegeben (p,q,d), RSA^{-1} (c):

$= „c^{1/e} \mod N“$ $= c^d \mod N$

$= „m^{e\,1/e} \mod N“$ $= m$

Bonus, Signieren einer Nachricht:

Gegeben sk (p,q,d):

$tag = RSA^{-1}(pk, h(m)) = RSA(d,h(m))$

Factoring (prime decomposition)

PRIVACY AND SECURITY

Theorem: all integers > 1 are either prime or a product of primes.

Factoring:

Consider set of integers $\mathbb{Z}_{(2)}(n) = \{ N=pq,$ where p,q are n-bit primes$\}$

Task: Find the prime factors (p *and* q) of a random N in $\mathbb{Z}_{(2)}(n)$

Best known algorithm (NFS): $\exp(\tilde{O}(\sqrt[3]{n}))$ for n-bit integers

Current world record: RSA-768 (232 digits) (200 machine years)

Consumed enough energy to heat to boiling point 2 olympic pools…

(Breaking RSA-2380 equivalent to evaporating all water on earth)

Lenstra, Kleinjung, Thomé

Hybride Verschlüsselung

PRIVACY AND SECURITY

secret key (session key) $k_{A,B}$

Public key

$k_{e,B}$

Private key

$k_{d,B}$

E

c_1

D

$k_{A,B} = D(k_{d,B}, c_1)$

Message

m

E

c_2

D

Message

$m = D(k_{A,B}, c_2)$

c_1, c_2

$c_1 = E(k_{e,B}, k_{A,B}),\ c_2 = E(k_{A,B}, m)$

Zusammenfassung

Angreifer / Angreifermodell und Bedrohungen

Sicherheitsziele (CIA) und Sicherheitsdienste

Definitionen und Formalisierung als Spiel

Perfekte Sicherheit / semantische Sicherheit

One-Time-Pad, Stream-Cipher

Block-Cipher und deren Operationsmodi

Hash-Funktionen und MACs

Schlüssel-Vereinbarung, direkt und indirekt

Asymmetrische Verschlüsselung und signieren