Die Klausur

Dienstag den 19.2. um 9 Uhr im Felix-Klein-Hörsaal

Aktueller Überblick

§0 Einführende Worte (✓)

§1 Geschichtlicher Überblick (✓)

§2 Zufall (✓)

§3 Perfekte Sicherheit und ihre Grenzen (✓)

§4 Angriffsszenarien (✓)

§5 Der komplexitätstheoretische Ansatz(✓)

§6 Pseudozufallsgeneratoren und Stromchiffren (✓)

§7 Pseudozufallsfunktionen und Blockchiffren (✓)

§8 Message Authentication Codes und Hash-Funktionen (✓)

§9 Kryptographie mit öffentlichen Schlüsseln

§9 Kryptographie mit öffentlichen Schlüsseln

Hintergrund: Elementare Zahlentheorie

Modulorechnen

Die Modulorelation

$$a \equiv b \mod n \quad :\Longleftrightarrow \quad n \mid (a - b)$$

Der Mod-Operator

$$(a \bmod n) \in \{0, 1, \ldots n-1\}$$

mit

$$a \equiv (a \bmod n) \mod n$$

Restklassen

$$\cdots = [a-n]_n = [a]_n = [a+n]_n = [a+2n]_n = \cdots$$

$$[a]_n = [a+kn]_n \quad \text{für alle } k \in \mathbf{Z}$$

Der Restklassenring

$$\mathbf{Z}/n\mathbf{Z} = \{[0]_n, \ldots, [n-1]_n\}$$

mit

$$[a] + [b]_n = [a + b]_n$$

$$-[a]_n = [-a]_n$$

$$[a]_n \cdot [b]_n = [a \cdot b]_n$$

$\mathbf{Z}/n\mathbf{Z}$ ist

- mit der angegeben Addition eine abelsche Gruppe,
- mit der angegebenen Multiplikation ein abelsches Monoid,
- insgesamt ein Ring.

Wie steht's mit Dividieren / Invertieren?

Der Erweiterte Euklidische Algorithmus

Satz. Gegeben $a, b \in \mathbf{Z}$, kann man $\mathrm{ggT}(a, b)$ und $c, d \in \mathbf{Z}$ mit

$$ca + db = \mathrm{ggT}(a, b)$$

effizient berechnen.

Der Erweiterte Euklidische Algorithmus

Beispiel

$$
\begin{aligned}
23 &= 1 \cdot 17 + 6 \\
17 &= 2 \cdot 6 + 5 \\
6 &= 1 \cdot 5 + 1 \\
&\Longrightarrow \mathrm{ggT}(23, 17) = 1
\end{aligned}
$$

$$
\begin{aligned}
1 &= 6 - 1 \cdot 5 \\
&= 6 - (17 - 2 \cdot 6) = -17 + 3 \cdot 6 \\
&= -17 + 3 \cdot (23 - 1 \cdot 17) = 3 \cdot 23 - 4 \cdot 17
\end{aligned}
$$

Invertieren in $\mathbf{Z}/n\mathbf{Z}$

Es sei $[a]_n \in \mathbf{Z}$ gegeben.

Ist $[a]_n$ invertierbar?

Gibt es ein Zahl $c \in \mathbf{Z}$ mit $[c]_n \cdot [a]_n = 1$?

$$[c]_n \cdot [a]_n = 1$$

ist äquivalent zu:

$$c \cdot a \equiv 1 \mod n \,.$$

Dies bedeutet: Es gibt ein $d \in \mathbf{Z}$ mit

$$ca + dn = 1$$

Und das heißt:

$$\mathrm{ggT}(c, n) = 1 \,.$$

Invertieren in $\mathbf{Z}/n\mathbf{Z}$

Beispiel

Es ist

$$3 \cdot 23 - 4 \cdot 17 = 1$$

$$\Longrightarrow 4 \cdot 17 \equiv 1 \mod 23$$

$$\Longrightarrow [4]_{23} \cdot [17]_{23} = [1]_{23}$$

$$\Longrightarrow [17]_{23}^{-1} \text{ existiert und } = [4]_{23}$$

Invertieren in $\mathbf{Z}/n\mathbf{Z}$

Satz

Die Restklasse $[a]_n \in \mathbf{Z}/n\mathbf{Z}$ ist genau dann invertierbar, wenn $\mathrm{ggT}(a, n) = 1$ ist.

Diese Bedingung und ggf. das Inverse sind effizient berechenbar.

Korollar. Für eine Primzahl p ist $\mathbf{Z}/p\mathbf{Z}$ ein Körper.

Notation. $\mathbf{F}_p := \mathbf{Z}/p\mathbf{Z}$

Kleiner Satz von Fermat

Satz. Es sei p eine Primzahl und $a \in \mathbf{Z}$ mit $p \nmid a$. Dann gilt

$$a^{p-1} \equiv 1 \mod p$$

Mit anderen Worten:

$$[a]_p^{p-1} = [1]_p$$

Für alle $a \in \mathbf{Z}$ ist

$$a^p \equiv a \mod p$$

$$[a]_p^p = [a]_p$$

Allgemeiner:

Es sei $k \in \mathbf{Z}$. Dann ist

$$a^{k\cdot(p-1)+1} \equiv a \mod p\,.$$

Kleiner Satz von Fermat

$$a^{k\cdot(p-1)+1} \equiv a \mod p\ .$$

Also: Für

$$\ell \equiv 1 \mod (p-1)$$

ist

$$a^{\ell} \equiv a \mod p$$

$$[a]_p^{\ell} = [a]_p$$

Der Chinesische Restsatz

Es sei $N = pq$ für verschiedene Primzahlen p, q.

Wir haben eine strukturerhaltene surjektive Abbildung

$$\mathbf{Z}/N\mathbf{Z} \longrightarrow \mathbf{Z}/p\mathbf{Z}$$

$$[a]_N \longmapsto [a]_p$$

$$d = [\mathit{let}@\mathit{tokeneifnch} \mapsto [m]_p$$

einen Homomorphismus.

Auch für q und insgesamt:

$$\mathbf{Z}/N\mathbf{Z} \longrightarrow \mathbf{Z}/p\mathbf{Z} \times \mathbf{Z}/q\mathbf{Z}$$

$$[a]_N \longmapsto ([a]_p, [a]_q)$$

$$d = [\mathit{let}@\mathit{tokeneifnch} \mapsto ([m]_p, [m]_q)$$

Dies ist eine Bijektion, ein Isomorphismus.

Der "RSA-Satz"

Satz. Es seien p und q verschiedene Primzahlen, $N := pq$. Es sei $\ell \equiv 1 \bmod (p-1)(q-1)$ und $m \in \mathbf{Z}/N\mathbf{Z}$. Dann ist

$$m^\ell = m \in \mathbf{Z}/N\mathbf{Z}$$

Beweis

$$\mathbf{Z}/N\mathbf{Z} \longrightarrow \mathbf{Z}/p\mathbf{Z} \times \mathbf{Z}/q\mathbf{Z}$$

$$m \longmapsto ([m]_p, [m]_q)$$

ist ein Isomorphismus.

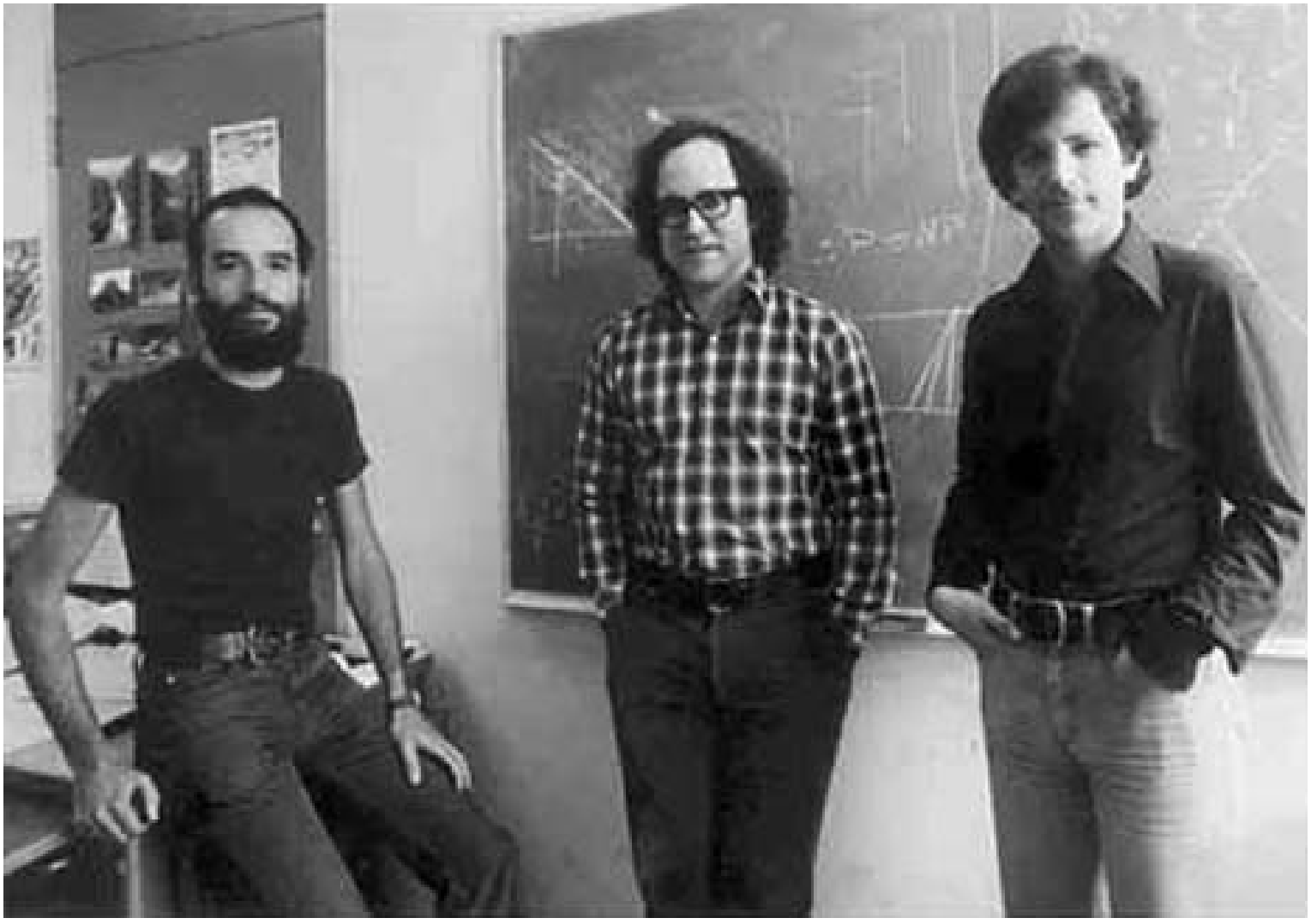
Es ist $\ell \equiv 1 \mod (p-1)$. Somit $[m]_p^\ell = [m]_p$.

Ebenso: $[m]_q^\ell = [m]_q$.

$$\Longrightarrow ([m]_p, [m]_q])^\ell = ([m]_p^\ell, [m]_q^\ell) = ([m]_p, [m]_q)$$

$$\Longrightarrow m^\ell = m$$

RSA



Adi Shamir, Ron Rivest, Len Adleman

“Plain” RSA-Verschlüsselung

Setup

Wähle “große” Primzahlen p, q sowie eine Zahl e (z.B. 3).

Berechne d mit $de \equiv 1 \mod (p-1)(q-1)$.

Setze $N := pq$.

Der private Schlüssel ist (N, d).

Der öffentliche Schüssel ist (N, e).

Der Nachrichtenraum ist $\mathbf{Z}/N\mathbf{Z}$.

Es ist $(m^e)^d = m^{de} = m$ für alle $m \in \mathbf{Z}/N\mathbf{Z}$.

Ver- und Entschlüsselung

Die Verschlüsselung zu den gegebenen Schlüsseln ist $m \mapsto c := m^e$.

Die Entschlüsselung zu den gegebenen Schlüsseln ist
$c \mapsto c^d (= m)$.

“Plain” RSA-Signatur

Setup

Wähle “große” Primzahlen p, q sowie eine Zahl e (z.B. 3).

Berechne d mit $de \equiv 1 \mod (p-1)(q-1)$.

Setze $N := pq$.

Der private Schüssel ist (N, d).

Der öffentliche Schlüssel ist (N, e).

Der Nachrichtenraum ist $\mathbf{Z}/N\mathbf{Z}$.

Es ist $(m^d)^e = m^{de} = m$ für alle $m \in \mathbf{Z}/N\mathbf{Z}$.

Signatur und Verifikation

Die Signatur zu den gegebenen Schlüsseln und Nachricht m ist $s := m^d$.

Die Verifikation zu den gegebenen Schlüsseln und signierter Nachricht (m, s) ist $s^e \stackrel{?}{=} m$.

Grundlagen

Verschlüsselungssystem mit öffentlichen Schlüsseln

Ein Verschlüsselungssystem mit öffentlichen Schlüsseln besteht aus drei Algorithmen:

- Einem Schlüsselerzeugungsalgorithmus $\mathcal{G}en$.

  Eingabe: Sicherheitsparameter 1^n

  Ausgabe: Schlüsselpaar $(pk, sk) = \mathcal{G}en(1^n)$
- Einem Verschlüsselungsalgorithmus $\mathcal{E}nc$.

  Eingabe: öffentlicher Schlüssel pk,
  Nachricht $m \in \{0,1\}^*$ oder $m \in \{0,1\}^{\ell(n)}$.

  Ausgabe: Chiffriertext $c = \mathcal{E}nc_{pk}(m)$
- Einem Entschlüsselungsalgorithmus $\mathcal{D}ec$.

  Eingabe: Privater Schlüssel sk, Chiffriertext c

  Ausgabe: $\mathcal{D}ec_{sk}(m)$ deterministisch

mit $\mathcal{D}ec_{sk^{(0)}}(\mathcal{E}nc_{pk^{(0)}}(m^{(0)})) = m^{(0)}$ für (fast) alle relevanten Schlüsselpaare $(pk^{(0)}, sk^{(0)})$ und Nachrichten $m^{(0)}$.

Ein einfaches Ununterscheidbarkeits-Spiel

Gegeben: $\mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec$ mit öffentlichen Schlüsseln.

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
4. Der Herausforderer wählt $i \in \{1, 2\}$ uniform. Er berechnet $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.
5. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.

Der Angreifer hat gewonnen, wenn $i = j$ ist.

Das Ununterscheidbarkeits-Spiel zu EAV und CPA

Gegeben: $\mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec$ mit öffentlichen Schlüsseln.

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
4. Der Herausforderer wählt $i \in \{1, 2\}$ uniform. Er berechnet $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.
5. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.

Der Angreifer hat gewonnen, wenn $i = j$ ist.

IND-EAV = IND-CPA

Das Spiel zum nicht-adaptiven CCA

Gegeben: $\mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec$ mit öffentlichen Schlüsseln.

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt einen Schlüssel $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer kann beliebige Chiffriertext entschlüsseln lassen.
4. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
5. Der Herausforderer wählt $i \in \{1, 2\}$ uniform. Er berechnet $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.
6. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.

Der Angreifer hat gewonnen, wenn $i = j$ ist.

Das Spiel zum adaptiven CCA (CCA2)

Gegeben: $\mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec$ mit öffentlichen Schlüsseln.

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer kann beliebige Texte entschlüsseln lassen.
4. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
5. Der Herausforderer wählt $i \in \{1, 2\}$ uniform. Er berechnet $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.
6. Der Angreifer kann beliebige Chiffriertexte verschieden von c entschlüsseln lassen.
7. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.

Der Angreifer hat gewonnen, wenn $i = j$ ist.

Signaturverfahren

Ein Signaturverfahren besteht aus:

- Einem Schlüsselerzeugungsalgorithmus $\mathcal{G}en$.

  Eingabe: 1^n mit $n =$ Sicherheitsparameter

  Ausgabe: Schlüsselpaar $(sk, pk) = \mathcal{G}en(1^n)$
- Einem Signieragorithmus $\mathcal{S}ign$
  Eingabe: Privater Schlüssel sk,
  Nachricht $m \in \{0,1\}^*$ oder $m \in \{0,1\}^\ell \quad (\ell = \ell(n))$

  Ausgabe: Signatur $s = \mathcal{S}ign_{sk}(m)$
- Einem deterministischen Verifikationsalgorithmus $\mathcal{V}rfy$.
  Eingabe: Öffentlicher Schlüssel pk, Nachricht m, Signatur s

  Ausgabe: $\mathcal{V}rfy_{pk}(m, s) \in \{0 =$ "ungültig", $1 =$ "gültig"$\}$

mit: $\mathcal{V}rfy_{pk^{(0)}}(m^{(0)}, \mathcal{S}ign_{sk^{(0)}}(m^{(0)})) = 1$ (mit Wahrscheinlichkeit 1)
für (fast) alle Schlüsselpaare $(sk^{(0)}, pk^{(0)})$ und Nachrichten $m^{(0)}$.

Spiel für die Sicherheit von Signaturverfahren

Gegeben: $\mathcal{G}en, \mathcal{S}ign, \mathcal{V}rfy$.

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer kann beliebige Signaturen erzeugen lassen, d.h. für Nachricht m nach Signatur $\mathcal{S}ign_{sk}(m)$ fragen.
4. Der Angreifer gibt ein Tupel (m, s) aus.

Der Angreifer hat gewonnen, wenn dies gültig ist $(\mathcal{V}rfy_{pk}(m, s) = 1)$ und er nicht nach m gefragt hat.

Def. Das Verfahren ist sicher, wenn jeder PPT-Angreifer nur mit vernachlässigbarem Erfolg hat.

Probleme mit “plain” RSA

Probleme

RSA-Verschlüsselung und -Signatur

Wie genau sollen die Primzahlen p, q gewählt werden?

Der Nachrichtenraum sollte $\{0,1\}^\ell$ sein, nicht $\mathbf{Z}/N\mathbf{Z}$.

RSA-Verschlüsselung

Das Verfahren ist nicht ranomisiert und somit unsicher nach jedem sinnvollen Kriterium.

Für Nachrichten m_1, m_2 gilt

$$\mathcal{D}ec_{sk}(m_1)\,\mathcal{D}ec_{sk}(m_2) = \mathcal{D}ec_{sk}(m_1 m_2)$$

RSA-Signatur

Für Nachrichten m_1, m_2 gilt

$$\mathcal{S}ign_{sk}(m_1)\,\mathcal{S}ign_{sk}(m_2) = \mathcal{S}ign_{sk}(m_1 m_2)$$

Somit ist das Verfahren auch vollkommen unsicher.

Lösungsansätze

Verschlüsselung

Für Sicherheit gegenüber Lauschern: Randomisieren (wie?)

Problem: Mangelnde Authentisierung

Signatur

Zuerst haschen, dann signieren

Ein partielles Resultat

Neues Signaturverfahren

Zuerst haschen, dann RSA-signieren

RSA-Probleme

Wir fixieren einen Schlüsselerzeugungsalgorithmus $\mathcal{G}en$ mit Ausgabe (pk, sk) mit $pk = (N, e), sk = (N, d)$.

Wir erhalten das zugehörige RSA-Problem:

Gegeben $pk = (N, e)$, berechne d.

$[N = pq\ ,\ de \equiv 1 \mod (p-1)(q-1)]$

Satz. Es sei ein Schlüsselerzeugungsalgorithmus fixiert.

Wenn jeder PPT-Algorithmus für das entsprechende RSA-Problem nur vernachlässigbaren Erfolg hat, dann ist "zuerst-hashen-dann-RSA-signieren" sicher im Zufallsorakelmodell.

Hybride Systeme

Schlüsselvereinbarung

In der Praxis:

Verfahren mit öffentlichen Schlüsseln (wie RSA) sind wesentlich langsamer als “normale” Verfahren wie AES.

Deshalb: Verschlüsseln von Daten erfolgt mit einem “normalen” Verfahren.

Verfahren mit öffentlichen Schlüsseln nur für Kommunikationsaufbau.

Schlüsselvereinbarung

Für Schlüsselvereinbarung:

"Aushandeln" (z.B. mit Diffie-Hellman)

oder

KEM = Key Encapsulation Mechanism (z.B. mit RSA)

Danach:

DEM = Data Encapsulation Mechanism

KEM und DEM zusammen: KEM-DEM

Das Diffie-Hellman-Protokoll

Alice und Bob einigen sich (in der Öffentlichkeit) auf eine große Primzahl p und ein Element $a \in \mathbf{F}_p^*$.

Alice		**Bob**
Wählt $x < \mathrm{ord}(a)$.		Wählt $y < \mathrm{ord}(a)$.
	$X := a^x$ →	
	← $Y := a^y$	

$$\boxed{Y^x = a^{xy} = X^y}$$

Zu KEM-DEM

Satz. Wir betrachten ein Verfahren beruhend auf dem KEM-DEM-Paradigma.

a) Wenn der KEM IND-CPA-sicher ist und der DEM IND-EAV-sicher ist, dann ist das Verfahren IND-CPA-sicher.

b) Wenn der KEM IND-CCA2-sicher ist und der DEM IND-CCA2-sicher ist, dann ist das Verfahren IND-CCA2-sicher.

Authentisierung

Das Problem und eine Lösung

Problem. Bis jetzt gibt es keine Authentisierung.

Alice weiß gar nicht, ob sie mit Bob kommuniziert ...

Lösungsansatz. Eine vertrauensvolle Instanz muss gegenüber Alice garantieren, dass sich "Bob" Nennende tatsächlich Bob ist.

Das geht mit Zertifikaten.

Publik-Key-Infrastruktur

Zentralistischer Ansatz

Es gibt eine Certificate Authority (CA).

Deren öffentlicher Schlüssel ist unumstritten.

Bob muss sich gegenüber der CA ausweisen, erhält dann ein von der CA unterschriebenes Schlüsselpaar.

In der Praxis:

- SSL-Zertifikate
- Trust-Center

Dezentraler Ansatz

Web-of-Trust

Unterstützende Hardware

Smartcards

Mehr Mathe ...

Das Faktorisierungsproblem und das diskrete Logarithmusproblem bieten “relativ viel” Angriffsmöglichkeiten.

Viel besser als RSA und Diffie-Hellman “mod p” ist ...

Elliptische-Kurven-Kryptographie