



max planck institut  
informatik

# Ideen und Konzepte der Informatik

## Kryptographie

Wie funktioniert Electronic Banking?

Kurt Mehlhorn



# Übersicht

- Zwecke der Kryptographie
- Techniken
  - Symmetrische Verschlüsselung (Caesar, One-time Pad, moderne Blockchiffres, seit 2000 Jahren).
  - Asymmetrische Verschlüsselung, Public-Key Kryptographie (seit 1978), RSA, ElGamal.
  - Digitale Unterschriften.
- Anwendungen (Sie benutzen Krypto täglich)
  - Electronic Banking, Einkaufen im Internet.
  - Digitale Unterschriften,
  - Geschützte Kommunikation, etwa WhatsApp

# Kryptographie (geheim-schreiben)

## Hauptziele (nach Wolfgang Ertel)

**Vertraulichkeit** / Zugriffsschutz: Nur berechnigte Personen können die Daten / Nachricht lesen.

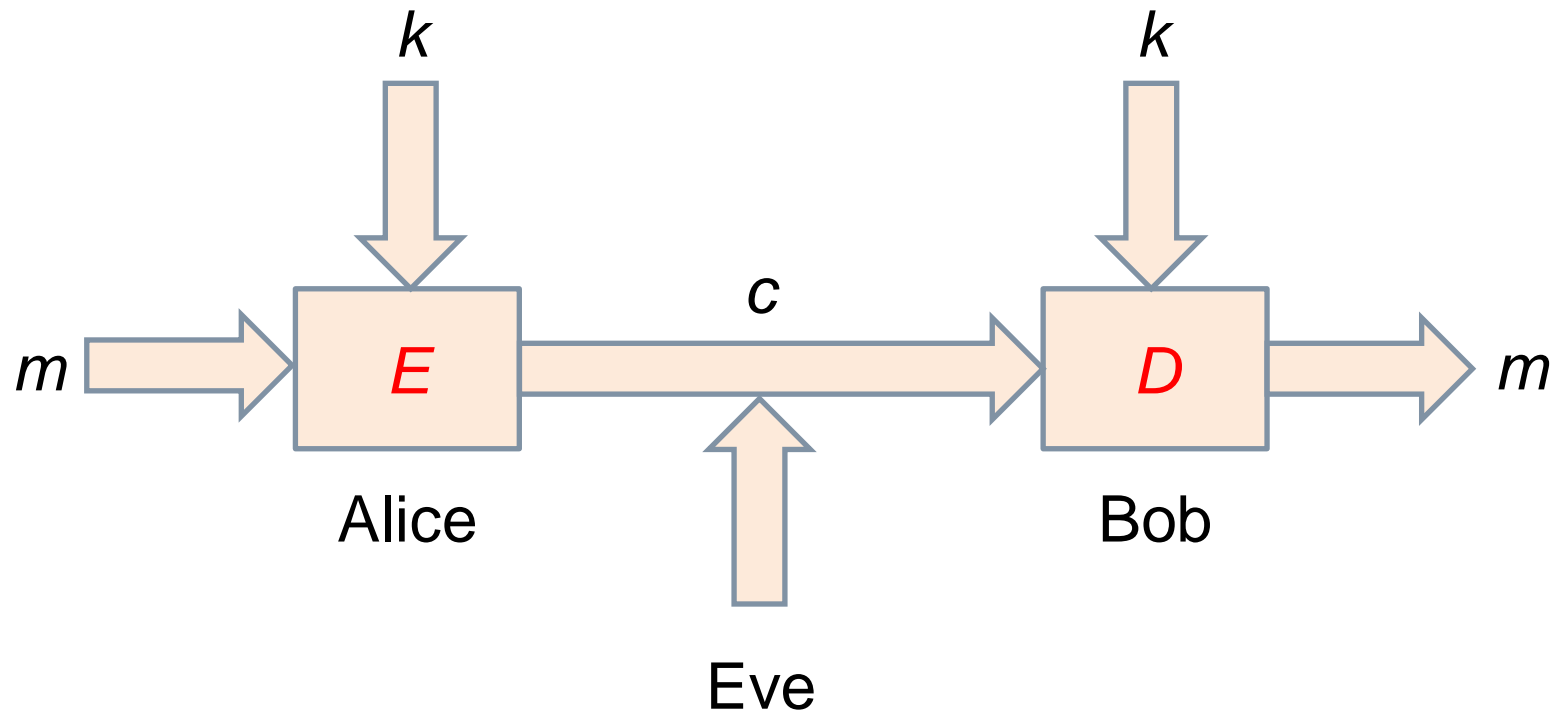
**Integrität** / Änderungsschutz: Daten können nicht unbemerkt verändert werden.

**Authentizität** / Fälschungsschutz: Der Urheber der Daten oder der Absender der Nachricht soll eindeutig identifizierbar sein

**Verbindlichkeit** / Nichtabstreitbarkeit: Urheberschaft sollte nachprüfbar und nicht abstreitbar sein.

# Symmetrische Verschlüsselung

Alice und Bob verabreden einen gemeinsamen Schlüssel  $k$



Eve = Eavesdropper (Lauscher)

# Beispiel: Caesar

- *D* und *E* sind ein Gerät.
- Schlüssel *k* ist Drehwinkel, bzw. das Ziel von *A*.
- *E* liest von innen nach außen  
*D* von außen nach innen.
- Einfach, aber sehr unsicher; nur 26 Schlüssel.



# Nomenklatur

- $E$  (Encoder, Verschlüsselung) und  $D$  (Decoder, Entschlüsselung) sind (allgemein bekannte) Geräte (heute meist Programme).
- Diese werden durch den Schlüssel  $k$  personalisiert.
- $m$  = Klartext, Nachricht, message
- $c$  = Geheimtext, cyphertext,  $c = E(k,m)$  und dann  $m = D(k,c)$
- Ohne Kenntnis von  $k$  soll es praktisch unmöglich (10min, 5h, 100 Jahre) sein,  $m$  aus  $c$  zu bestimmen.

# Symmetrische Kryptographie – Eine Analogie

- Alice und Bob kaufen sich eine Kiste und ein Vorhängeschloss mit zwei identischen Schlüsseln. Jeder bekommt einen Schlüssel.
- Nachricht  $m$  kommt in die Kiste und die Kiste wird mit dem Schlüssel verschlossen (Verfahren E). Kiste wird verschickt und mit dem Schlüssel geöffnet (Verfahren D).
- Braucht ein Treffen oder einen vertrauenswürdigen Boten

# One-Time Pad (Rotes Telefon)

- Wie Caesar, aber für jeden Buchstaben des Texts benutzt man einen eigenen Schlüssel, d.h.
- Schlüssel ist ein zufälliger Text (jeder Buchstabe ist gewürfelt) mit der gleichen Länge wie die Nachricht.
- Absolut sicher, aber Schlüssel muss genauso lang wie Nachricht sein
- Schlüsselaustausch ist aufwendig

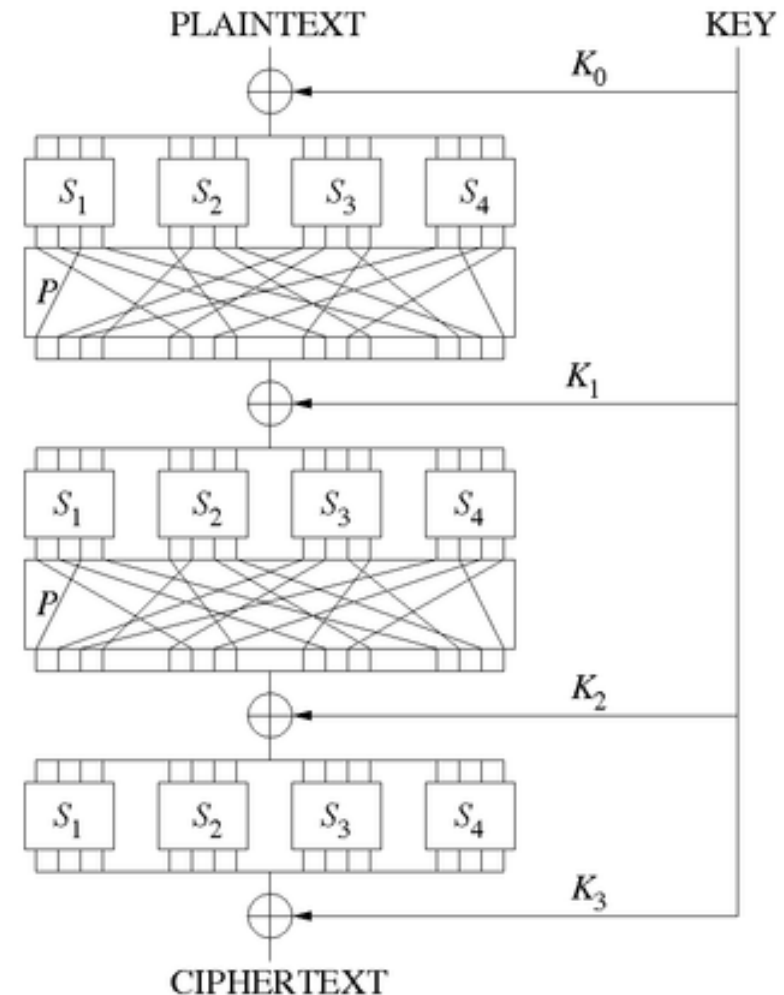


# Blockchiffrierung

- Nachricht wird in Blöcke fester Länge zerlegt. Typische Blocklänge 64, 128, 256 Bits.
- Man hat einen Schlüssel, dessen Länge gleich der Blocklänge ist. Bei 128 Bit,  $2^{128}$  verschiedene Schlüssel.
- Jeder Block wird mit dem Schlüssel verschlüsselt.
- Populäre Verfahren: DES (Data-Encryption-Standard), AES (Nachfolger)
- Sicherheit: 64Bit Versionen unsicher, 128Bit noch sicher.

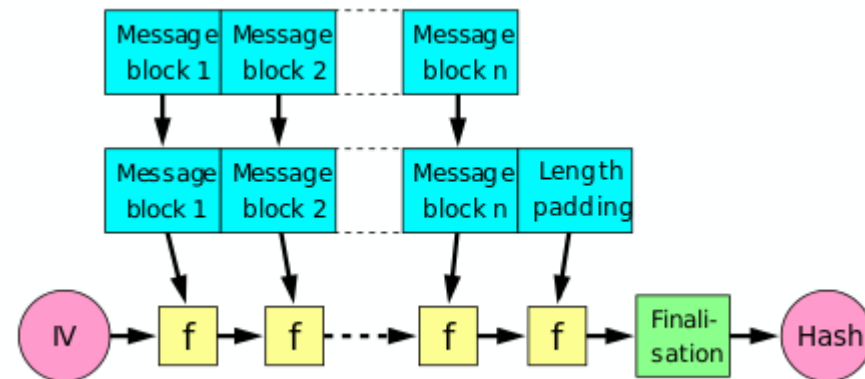
# Blockchiffrierung: Prinzip der Vorgehensweise (schwarze Kunst)

- Kodierung eines Blocks der Länge  $b = 128$
- Verknüpfung mit dem Schlüssel (wie im One-Time Pad)
- Fasse Block als Folge von 16 Miniblocken von je 8 Bit auf.  
8 Bit = 0 .. 255. Substituiere 0 → 132, 1 → 211
- Permutiere die Positionen
- Wiederhole 16 Mal.
- Sicherheit = schwarze Kunst



# Weitere Anwendungen von Kryptographischem Hashing

- **Integrität eines Textes:** Zerlege den Text in Blöcke der Länge 128 Zeichen. Benutze die Hashfunktion wiederholt, um den Text auf 128 Zeichen zu komprimieren (Digest des Textes).
  - Schutz gegen Übertragungsfehler.
  - Commitment (Festlegung auf einen Text): ich veröffentliche das Digest und habe mich damit auf den Text festgelegt.
- **Proof of Work:** Beweise dein Interesse indem du „rmdtswcaesdq“ so zu 128 Zeichen ergänzt, dass der Hash mit 7 Nullen beginnt.



# Angriffe



- Caesar:  
Buchstabenhäufigkeit
- DES 56: brute-force mit  
Spezialhardware
- ENIGMA (Rätsel): Alan  
Turing



# Symmetrische Verfahren – Zusammenfassung

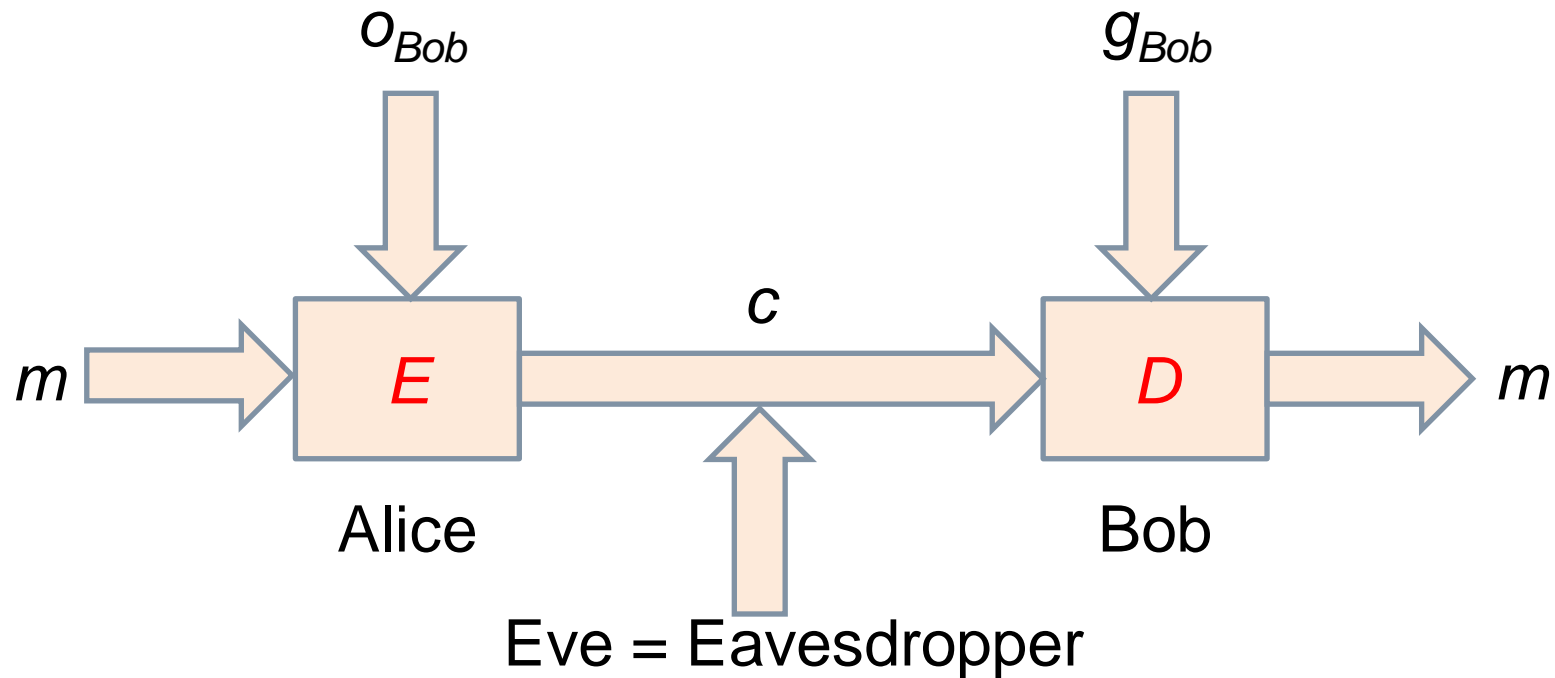
- Sender (Alice) und Empfänger (Bob) verabreden einen gemeinsamen Schlüssel  $k$ .
- Dieser Schlüssel muss geheim bleiben.
- Wie einigt man sich auf einen Schlüssel?
  - Früher: Treffen oder Bote
  - Heute: asymmetrisches Verfahren zum Schlüsselaustausch
- Beispiele: One-Time Pad, Caesar, AES128
- Sehr effiziente Ver- und Entschlüsselung.
- Bei  $n$  Teilnehmern:  $n \times n$  Schlüssel.

# Asymmetrische Verfahren (seit 78)

- (Empfänger) Bob erzeugt Schlüsselpaar  $g_{Bob}$  und  $o_{Bob}$ , hält  $g_{Bob}$  geheim, veröffentlicht  $o_{Bob}$ .
- Jeder, der Bob eine Nachricht schicken will, benutzt  $o_{Bob}$  zum Verschlüsseln.
- $g_{Bob}$  kann aus  $o_{Bob}$  nach heutiger Kenntnis nicht in wenigen Jahren berechnen. Ohne Kenntnis von  $g_{Bob}$  kann man nicht entschlüsseln.

# Ver- und Entschlüsselung

$$m = D(g_{Bob}, E(o_{Bob}, m))$$



# Asymmetrisches Verfahren – Eine Analogie

- Bob möchte, dass man ihm geheime Nachrichten schicken kann.
- Er kauft viele identische Bügelschlösser und hinterlegt die offenen Schlösser an öffentlichen Orten. Er behält den einzigen Schlüssel.
- Alice tut ihre Nachricht in eine Kiste, verschließt die Kiste mit dem Bügelschloss und schickt die Kiste an Bob.
- Nur Bob kann die Kiste öffnen.
- Vorteil: kein Treffen nötig.
- Problem: aufwendiger, Authentifizierung, woher weiß Alice, dass das Schloss zu Bob gehört.

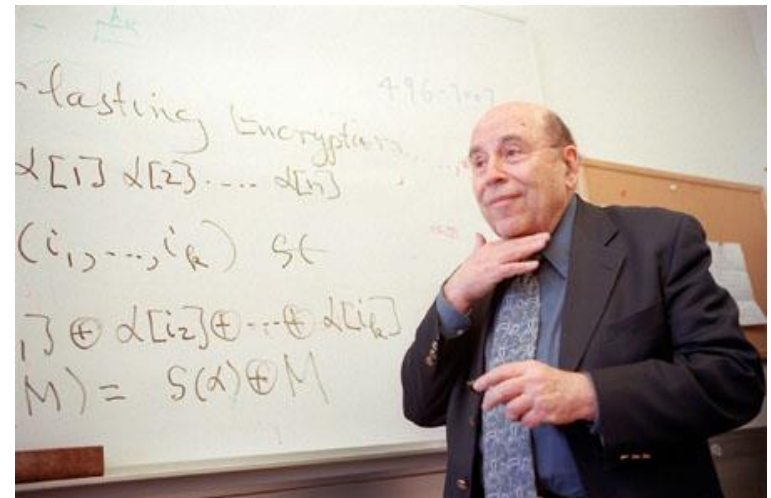
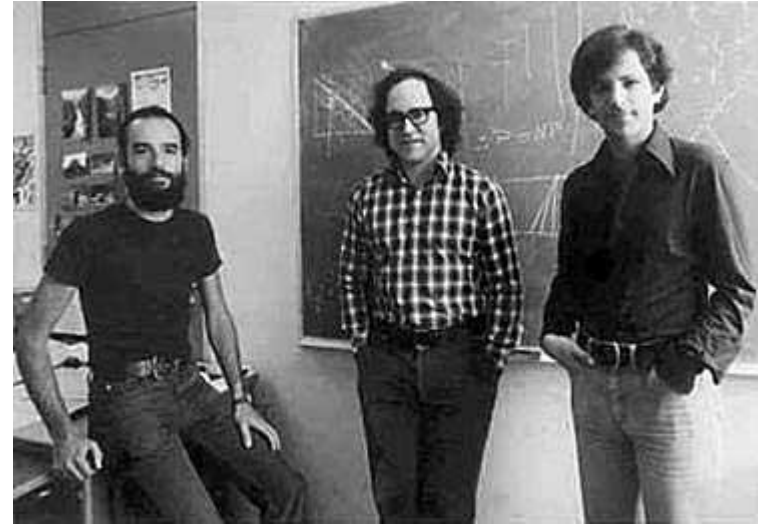


# Nomenklatur

- $E$  und  $D$  sind allgemein bekannte Geräte (heute meist Programme).
- Werden durch die Schlüssel personalisiert, also
  - $E_{Bob} = E$  mit Schlüssel  $o_{Bob}$  und
  - $D_{Bob} = D$  mit Schlüssel  $g_{Bob}$ .
- $E_{Bob}$  ist öffentlich; also kann jeder damit eine Nachricht für Bob verschlüsseln.
- Nur Bob kann  $D_{Bob}$  ausführen, da nur Bob seinen geheimen Schlüssel kennt.

# Erfinder

- RSA (Rivest-Shamir-Adleman, Turing Award), Rabin (Turing Award),
- Später dann Verfahren von ElGamal und Elliptische Kurven



# Sicherheit

- Sicherheit von RSA
  - Multiplizieren von 1000-stelligen Zahlen ist einfach, aber
  - sie aus ihrem Produkt zu berechnen, dauert nach heutigem Wissen 100 Jahre. Faktorisieren ist schwer
- ElGamal: das Gleiche gilt für den diskreten Logarithmus bezüglich 2000-stelliger Primzahl. Potenzieren ist einfach, Logarithmus ist schwer.
- 1000-stellige Primzahlen findet man leicht

# Baby-Version von ElGamal

- Darstellung folgt Bongartz / Unger (Alg. der Woche)
- Annahme: Wir können multiplizieren und addieren / subtrahieren, aber dividieren ist sehr sehr schwer, also
- Aus  $p$  und  $f$  kann man  $P = p \cdot f$  berechnen, aber niemand kann aus  $f$  und  $P (= p \cdot f)$  das  $p$  berechnen.

# Baby-Version von ElGamal

- **Empfänger** wählt  $p$  und  $f$ ; veröffentlicht  $f$  und  $P = p \cdot f$ ; hält  $p$  geheim.
- **Sender** möchte  $m$  schicken,  $m < P$
- Wählt eine zufällige Zahl  $s$  und schickt öffentlich ( $s$  bleibt geheim)

$$s \cdot f \text{ und } N = m + s \cdot P.$$

- **Empfänger** berechnet

$$p \cdot (s \cdot f) = s \cdot P$$

und dann 
$$m = N - s \cdot P.$$

# Baby-Version von ElGamal

- Empfänger wählt  $p$  und  $f$  und veröffentlicht  $f$  und  $P = p \cdot f$ .
- Sender möchte  $m$  schicken,  $m < P$ .
- Wählt eine Zahl  $s$  und schickt öffentlich  
 $s \cdot f$  und  $N = m + s \cdot P$ .
- **Eve** kennt  $f$ ,  $s \cdot f$ ,  $P = p \cdot f$  und weiß  
 $m$  liegt in  $\{N, N - P, N - 2P, N - 3P, \dots\}$   
**Eve** braucht  $s \cdot P$ . Da kommt man aber ohne Kenntnis von  
 $p$  nicht dran.

# Die Details von ElGamal

- Die Details von ElGamal werde ich in der Vorlesung nicht behandeln, die Folien sind zum Nachlesen.
- Im wesentlichen ersetzt man Addieren durch Multiplizieren und Multiplizieren durch Potenzieren. Dann spielt der Logarithmus die Rolle der Division. Ferner rechnet man Modulo einer Primzahl.

# Rechnen mod $n$

- Grundmenge =  $\{0, 1, \dots, n - 1\}$ , etwa  $n = 7$ .
- Addition, Subtraktion, Multiplikation mod  $n$

Bringe Ergebnis durch Restbildung wieder in die Grundmenge

$$4 \cdot 6 = 24 \equiv 3 \pmod{7}$$

$$3 + 4 \cdot 2 = 11 \equiv 4 \pmod{7}$$

- $n$  prim, dann gibt es zu jedem  $a \neq 0$  ein  $b$  so dass  $a \cdot b \equiv 1 \pmod{n}$  und es gibt ein  $g$  so dass  $\{g, g^2, \dots, g^{n-1}\} = \{1, \dots, n-1\}$ .



# ElGamal

- Empfänger wählt Primzahl  $p$ , Erzeuger  $g$  und ein  $x$  mit  $2 \leq x \leq p - 1$  und veröffentlicht  $(p, g, y)$  wobei  $y = g^x \text{ mod } p$ .
- Berechnung von  $y$  aus  $x$  ist leicht, aber Berechnung von  $x$  aus  $y$  ist nach heutiger Kenntnis praktisch unmöglich.
- Sender möchte  $m$  schicken, wählt  $s$  und schickt  
$$(z = g^s \text{ mod } p, N = m \cdot y^s \text{ mod } p)$$

# ElGamal

- Empfänger wählt Primzahl  $p$ , Erzeuger  $g$  und ein  $x$  mit  $2 \leq x \leq p - 1$  und veröffentlicht  $(p, g, y)$  wobei  $y = g^x \bmod p$ .
- Sender möchte  $m$  schicken, wählt  $s$  und schickt  
$$(z = g^s \bmod p, N = m \cdot y^s \bmod p)$$

Eve kennt  $y$  und weiß nur  $m$  liegt in  $\{N, N/y, N/y^2, \dots\}$ .

- Empfänger berechnet  $z^x = g^{sx} = y^s$  und dann  
 $m = N / y^s \bmod p$ .

# Electronic Banking

- Kunde kennt öffentlichen Schlüssel  $o_B$  der Bank.
- Kunde erfindet geheimen Schlüssel  $k$  (256 Bit Zufallszahl) für symmetrisches Verfahren.
- Kunde verschlüsselt  $k$  mit  $o_B$  und schickt den verschlüsselten Schlüssel an die Bank.
- Bank entschlüsselt mit Hilfe ihres privaten Schlüssels  $g_B$ .
- Nun symmetrisches Verfahren mit  $k$ .
  
- **Problem:** Woher kenne ich den öffentlichen Schlüssel meiner Bank? Wir kommen darauf zurück.

# Unterschriften

- Eigenschaft: Unterschreiber kann sie nicht abstreiten.
- Zweck: Verbindlichkeit.
- Was kann als Unterschrift dienen? Alles was nur der Unterschreiber kann:
  - Traditionell: handschriftliche Unterschrift, Fingerabdruck
  - Nun: Die Funktion  $D_x$  kann nur die Person X ausführen, weil nur sie ihren geheimen Schlüssel kennt.

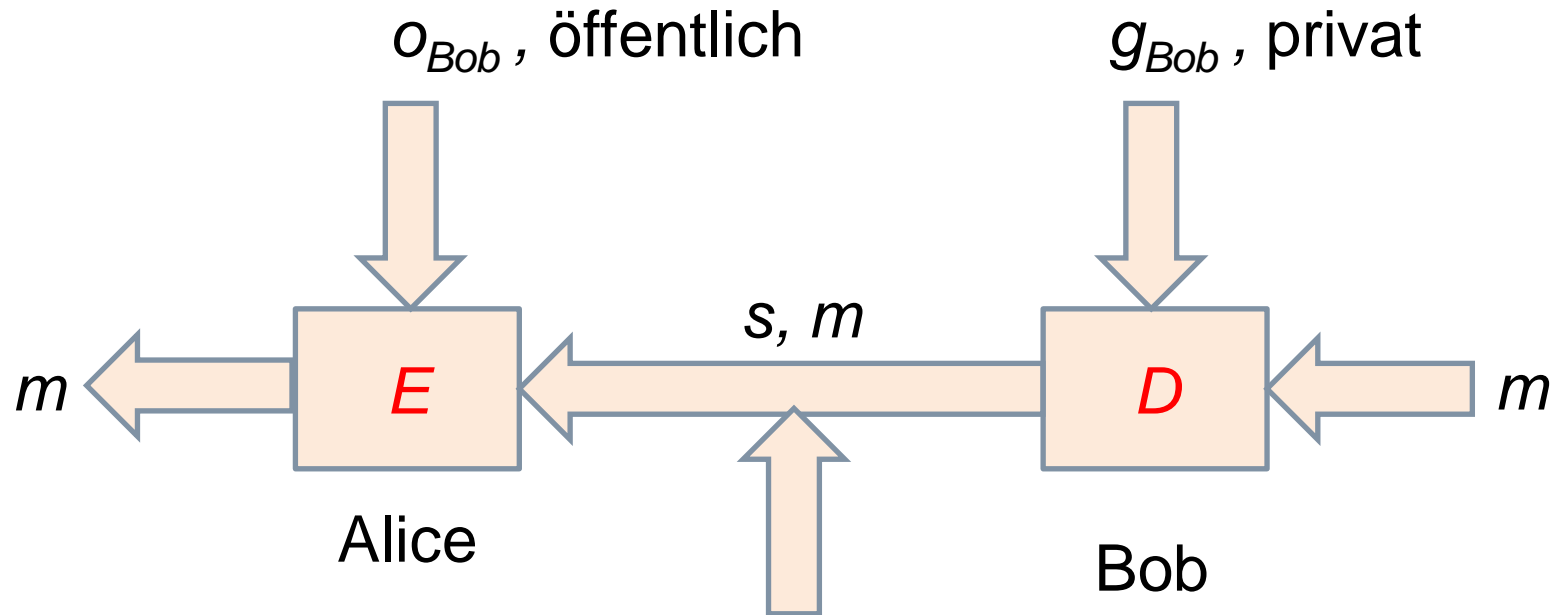
# Digitale Signaturen

- Seien  $E_x$  und  $D_x$  die Funktionen von  $X$  und gelte auch  $(E_x(D_x(x))) = x$  für alle  $x$ .
- Um  $m$  zu signieren, berechnet  $X$  den String  $s = D_x(m)$ .
- Das Paar  $(m,s)$  ist das unterschriebene  $m$ .
- Vertragspartner überprüft, dass  $E_x(s) = m$  gilt.
- **Nur  $X$  kann  $s$  aus  $m$  erzeugen. Also kann  $X$  die Unterschrift nicht abstreiten.**

# Digitale Signaturen

## Signatur = etwas, das nur ich kann

$$m = E(o_{Bob}, D(g_{Bob}, m))$$



$$s = D(g_{Bob}, m) = \text{Bobs Signatur von } m$$

# Bob möchte $m$ signieren und dann verschlüsselt an Alice schicken

---



# Electronic Banking, Schritt 1

- Bank hinterlegt ihren öffentlichen Schlüssel  $o_B$  bei einem Trustcenter (TC).
- Kunde kennt (fest eingebaut im Browser) den öffentlichen Schlüssel des TC und fragt nach Schlüssel der Bank.
- TC signiert  $o_B$  und schickt an Kunden.
- Kunde verifiziert die Unterschrift und benutzt dann  $o_B$  wie oben beschrieben.

High Security



Signaturgesetz

regtp Z 0 0 0 2



# Zusammenfassung

- Electronic Banking, Einkaufen im Netz nutzt symmetrische und asymmetrische Kryptographie.
- Kommunikation mit der Bank ist damit geschützt.  
*<https://my.hypovereinsbank.de/>*
- Aber Vorsicht: Verschlüsselte Übertragung garantiert noch nicht Gesamtsicherheit, z.B. unsicheres Passwort.
- Siehe erst Vorlesung „Sicherheit und Privatheit“.

# Kryptographie (geheim-schreiben)

## Hauptziele (nach Wolfgang Ertel)

**Vertraulichkeit** / Zugriffsschutz: Nur dazu berechnigte Personen sollen in der Lage sein, die Daten oder die Nachricht zu lesen (auch teilweise).

**Nachricht / Daten verschlüsseln**

**Integrität** / Änderungsschutz: Die Daten müssen nachweislich vollständig und unverändert sein.

**Nachricht / Daten verschlüsseln oder signieren**

**Authentizität**, Verbindlichkeit / Fälschungsschutz, Nichtabstreitbarkeit: Der Urheber der Daten oder der Absender der Nachricht soll eindeutig identifizierbar sein, und seine Urheberschaft sollte nachprüfbar und nicht abstreitbar sein.

**Nachricht / Daten signieren**

# Speicherung von Passwörtern

- $h$  = One-Way Funktion, z.B. Blockcypher.
- Sei  $c = h(\text{Passwort von KM})$ .
- Speichere ungeschützt das Paar  $(\text{KM}, c)$ .
- **KM beweist seine Authentizität durch die Fähigkeit  $c$  erzeugen zu können.**
- Angriffe: brute-force, da Passworte oft kurz. Findet Passwort eines Nutzers, nicht eines bestimmten Nutzers.
- Abhilfe
  - Maschine für  $h$  geht nach 3 inkorrekten Auswertungen kaputt.
  - Oder automatische Verlängerung durch Zufallsstring speichere  $(\text{KM}, \text{zufälliges } s, h(\text{Passwort von KM}, s))$ .