# Blinde Signaturen, geheime Abstimmungen und digitale Münzen

Claus Diem

Im Wintersemester 2017 / 18

David Chaum

Department of Computer Science
University of California
Santa Barbara, CA

## INTRODUCTION

Automation of the way we pay for goods and services is already underway, as can be seen by the variety and growth of electronic banking services available to consumers. The ultimate structure of the new electronic payments system may have a substantial impact on personal privacy as well as on the nature and extent of criminal use of payments. Ideally a new payments system should address both of these seemingly conflicting sets of concerns.

On the one hand, knowledge by a third party of the payee, amount, and time of payment for every transaction made by an individual can reveal a great deal about the individual's whereabouts, associations and lifestyle. For example, consider payments for such things as transportation, hotels, restaurants, movies, theater, lectures, food, pharmaceuticals, alcohol, books, periodicals, dues, religious and political contributions.

On the other hand, an anonymous payments systems like bank notes and coins suffers from lack of controls and security. For example, consider problems such as lack of proof of payment, theft of payments media, and black payments for bribes, tax evasion, and black markets.

A fundamentally new kind of cryptography is proposed here, which allows an automated payments system with the following properties:

(1) Inability of third parties to determine payee, time or amount of payments made by an individual.

(2) Ability of individuals to provide proof of payment, or to determine the identity of the payee under exceptional circumstances.



Copyright (c) 1998, Springer-Verlag

# Geheime Abstimmungen

# Eine geheime Abstimmung

**Problem.** Eine Gruppe von Personen will per Brief eine geheime Abstimmung durchführen.

# Eine geheime Abstimmung

**Problem.** Eine Gruppe von Personen will per Brief eine geheime Abstimmung durchführen.

Sie beauftragt einen Treuhänder.

# Eine geheime Abstimmung

**Problem.** Eine Gruppe von Personen will per Brief eine geheime Abstimmung durchführen.

Sie beauftragt einen Treuhänder.

Aber die Teilnehmer vertrauen dem Treuhänder nicht.

# Geheime Abstimmung mit blinder Signatur

1. Jeder Teilnehmer
   erhält einen / macht sich einen Stimmzettel,
   schreibt auf seinen Stimmzettel einen frei gewählten Code,
   füllt den Stimmzettel aus,
   legt den ausgefüllten Stimmzettel und ein Durchschlagpapier in einen Umschlag,
   versiegelt den Umschlag, schreibt seinen Absender drauf,
   schickt diesen ab.

# Geheime Abstimmung mit blinder Signatur

1. Jeder Teilnehmer
   erhält einen / macht sich einen Stimmzettel,
   schreibt auf seinen Stimmzettel einen frei gewählten Code,
   füllt den Stimmzettel aus,
   legt den ausgefüllten Stimmzettel und ein Durchschlagpapier in einen Umschlag,
   versiegelt den Umschlag, schreibt seinen Absender drauf,
   schickt diesen ab.
2. Der Treuhänder unterschreibt die Umschläge,
   schickt diese zurück.

# Geheime Abstimmung mit blinder Signatur

1. Jeder Teilnehmer
   erhält einen / macht sich einen Stimmzettel,
   schreibt auf seinen Stimmzettel einen frei gewählten Code,
   füllt den Stimmzettel aus,
   legt den ausgefüllten Stimmzettel und ein Durchschlagpapier in einen Umschlag,
   versiegelt den Umschlag, schreibt seinen Absender drauf,
   schickt diesen ab.
2. Der Treuhänder unterschreibt die Umschläge,
   schickt diese zurück.
3. Jeder Teilnehmer überprüft das Siegel, öffnet den Umschlag,
   nimmt den unterschrieben Stimmzettel heraus,
   schickt diesen wieder dem Treuhänder.

# Geheime Abstimmung mit blinder Signatur

1. Jeder Teilnehmer
   erhält einen / macht sich einen Stimmzettel,
   schreibt auf seinen Stimmzettel einen frei gewählten Code,
   füllt den Stimmzettel aus,
   legt den ausgefüllten Stimmzettel und ein Durchschlagpapier in einen Umschlag,
   versiegelt den Umschlag, schreibt seinen Absender drauf,
   schickt diesen ab.
2. Der Treuhänder unterschreibt die Umschläge,
   schickt diese zurück.
3. Jeder Teilnehmer überprüft das Siegel, öffnet den Umschlag,
   nimmt den unterschrieben Stimmzettel heraus,
   schickt diesen wieder dem Treuhänder.
4. Der Treuhänder macht alle (ausgefüllten und unterschriebenen) Stimmzettel öffentlich.

# Geheime Abstimmung mit blinder Signatur

1. Jeder Teilnehmer
   erhält einen / macht sich einen Stimmzettel,
   schreibt auf seinen Stimmzettel einen frei gewählten Code,
   füllt den Stimmzettel aus,
   legt den ausgefüllten Stimmzettel und ein Durchschlagpapier in einen Umschlag,
   versiegelt den Umschlag, schreibt seinen Absender drauf,
   schickt diesen ab.
2. Der Treuhänder unterschreibt die Umschläge,
   schickt diese zurück.
3. Jeder Teilnehmer überprüft das Siegel, öffnet den Umschlag,
   nimmt den unterschrieben Stimmzettel heraus,
   schickt diesen wieder dem Treuhänder.
4. Der Treuhänder macht alle (ausgefüllten und unterschriebenen) Stimmzettel öffentlich.
5. Jeder Teilnehmer überprüft, ob sein Zettel dabei ist und ob die Anzahl stimmt.

# Blinde Signaturen

# Interaktives Signatur-Verfahren

Ein Interaktives Signaturverfahren besteht aus effizienten Algorithmen:

- einem Schlüsselerzeugungsagorithmus $\mathcal{G}en$:
  Eingabe: $1^n$
  Ausgabe: $\mathcal{G}en(1^n)$, ein Schlüsselpaar $(sk, pk)$

# Interaktives Signatur-Verfahren

Ein Interaktives Signaturverfahren besteht aus effizienten Algorithmen:

- einem Schlüsselerzeugungsagorithmus $\mathcal{G}en$:
  Eingabe: $1^n$
  Ausgabe: $\mathcal{G}en(1^n)$, ein Schlüsselpaar $(sk, pk)$
- interagierenden Algorithmen $\mathcal{S}ign, \mathcal{U}ser$,
  Eingabe an $\mathcal{U}ser$: $(pk, m)$
  Eingabe an $\mathcal{S}ign$: $sk$
  Ausgabe von $\mathcal{U}ser$: $\langle \mathcal{S}ign_{sk}, \mathcal{U}ser_{pk}(m) \rangle$, eine Signatur $\sigma$

# Interaktives Signatur-Verfahren

Ein Interaktives Signaturverfahren besteht aus effizienten Algorithmen:

- einem Schlüsselerzeugungsagorithmus $\mathcal{G}en$:
  Eingabe: $1^n$
  Ausgabe: $\mathcal{G}en(1^n)$, ein Schlüsselpaar $(sk, pk)$
- interagierenden Algorithmen $\mathcal{S}ign, \mathcal{U}ser$,
  Eingabe an $\mathcal{U}ser$: $(pk, m)$
  Eingabe an $\mathcal{S}ign$: $sk$
  Ausgabe von $\mathcal{U}ser$: $\langle \mathcal{S}ign_{sk}, \mathcal{U}ser_{pk}(m) \rangle$, eine Signatur $\sigma$
- einem Verifikationsalgorithmus $\mathcal{V}rfy$.
  Eingabe: $(pk, m, \sigma)$
  Ausgabe: $\mathcal{V}rfy_{pk}(m, \sigma) \in \{0, 1\}, 1 =$ "ja" / $0 =$ "nein"

# Interaktives Signatur-Verfahren

Ein Interaktives Signaturverfahren besteht aus effizienten Algorithmen:

- einem Schlüsselerzeugungsagorithmus $\mathcal{G}en$:
  Eingabe: $1^n$
  Ausgabe: $\mathcal{G}en(1^n)$, ein Schlüsselpaar $(sk, pk)$
- interagierenden Algorithmen $\mathcal{S}ign, \mathcal{U}ser$,
  Eingabe an $\mathcal{U}ser$: $(pk, m)$
  Eingabe an $\mathcal{S}ign$: $sk$
  Ausgabe von $\mathcal{U}ser$: $\langle \mathcal{S}ign_{sk}, \mathcal{U}ser_{pk}(m) \rangle$, eine Signatur $\sigma$
- einem Verifikationsalgorithmus $\mathcal{V}rfy$.
  Eingabe: $(pk, m, \sigma)$
  Ausgabe: $\mathcal{V}rfy_{pk}(m, \sigma) \in \{0, 1\}, 1 =$ "ja" / $0 =$ "nein"

mit: Wenn die Algorithmen hintereinander ausgeführt werden, ist $\mathcal{V}er_{pk}(m, \sigma) = 1$

# Interaktives Signatur-Verfahren

Ein Interaktives Signaturverfahren besteht aus effizienten Algorithmen:

- einem Schlüsselerzeugungsagorithmus $\mathcal{G}en$:
  Eingabe: $1^n$
  Ausgabe: $\mathcal{G}en(1^n)$, ein Schlüsselpaar $(sk, pk)$
- interagierenden Algorithmen $\mathcal{S}ign, \mathcal{U}ser$,
  Eingabe an $\mathcal{U}ser$: $(pk, m)$
  Eingabe an $\mathcal{S}ign$: $sk$
  Ausgabe von $\mathcal{U}ser$: $\langle \mathcal{S}ign_{sk}, \mathcal{U}ser_{pk}(m) \rangle$, eine Signatur $\sigma$
- einem Verifikationsalgorithmus $\mathcal{V}rfy$.
  Eingabe: $(pk, m, \sigma)$
  Ausgabe: $\mathcal{V}rfy_{pk}(m, \sigma) \in \{0, 1\}, 1 =$ "ja" / $0 =$ "nein"

mit: Wenn die Algorithmen hintereinander ausgeführt werden, ist $\mathcal{V}er_{pk}(m, \sigma) = 1$
(oder zumindest mit hoher Wahrscheinlichkeit).

# Blinde Signaturen

**Anforderungen.**

- Unfälschbarkeit
- Blindheit

# Blinde Signaturen

**Anforderungen.**

- Unfälschbarkeit
- Blindheit

Im Folgenden nach

David Pointcheval und Jacques Stern: Security arguments for digital signatures and blind signatures,
Journal of Cryptology, 2000

Dominique Schröder und Dominique Unruh: Security of Blind Signatures Revisited, PKC 2002

(Einige "technische Kleinigkeiten" sind nicht exakt beschrieben.)

# Spiel zur Unfälschbarkeit

Eingabe: Sicherheitsparameter $1^n$

# Spiel zur Unfälschbarkeit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$

# Spiel zur Unfälschbarkeit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Herausforderer erzeugt einen Schlüssel $(sk, pk)$ und gibt $pk$ an den Angreifer.

# Spiel zur Unfälschbarkeit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Herausforderer erzeugt einen Schlüssel $(sk, pk)$ und gibt $pk$ an den Angreifer.
3. Der Angreifer lässt Nachrichten $m_1, \ldots, m_k$ signieren (mittels Interaktion mit $\mathcal{S}ign_{sk}$)

# Spiel zur Unfälschbarkeit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Herausforderer erzeugt einen Schlüssel $(sk, pk)$ und gibt $pk$ an den Angreifer.
3. Der Angreifer lässt Nachrichten $m_1, \ldots, m_k$ signieren (mittels Interaktion mit $\mathcal{S}ign_{sk}$)
4. Der Angreifer berechnet ein Tupel $(m, \sigma)$.

# Spiel zur Unfälschbarkeit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Herausforderer erzeugt einen Schlüssel $(sk, pk)$ und gibt $pk$ an den Angreifer.
3. Der Angreifer lässt Nachrichten $m_1, \ldots, m_k$ signieren (mittels Interaktion mit $\mathcal{S}ign_{sk}$)
4. Der Angreifer berechnet ein Tupel $(m, \sigma)$.
5. Der Angreifer hat gewonnen, wenn $m \notin \{m_1, \ldots, m_k\}$ und $\mathcal{V}er_{pk}(m, \sigma) = 1$ ist.

# Spiel zur Unfälschbarkeit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Herausforderer erzeugt einen Schlüssel $(sk, pk)$ und gibt $pk$ an den Angreifer.
3. Der Angreifer lässt Nachrichten $m_1, \ldots, m_k$ signieren (mittels Interaktion mit $\mathcal{S}ign_{sk}$)
4. Der Angreifer berechnet ein Tupel $(m, \sigma)$.
5. Der Angreifer hat gewonnen, wenn $m \notin \{m_1, \ldots, m_k\}$ und $\mathcal{V}er_{pk}(m, \sigma) = 1$ ist.

Unfälschbarkeit: Jeder PPT-Angreifer gewinnt nur mit vernachlässigbarer Wahrscheinlichkeit.

# Spiel zur Unfälschbarkeit (nicht ganz)

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Herausforderer erzeugt einen Schlüssel $(sk, pk)$ und gibt $pk$ an den Angreifer.
3. Der Angreifer lässt Nachrichten $m_1, \ldots, m_k$ signieren (mittels Interaktion mit $\mathcal{S}ign_{sk}$)
4. Der Angreifer berechnet ein Tupel $(m, \sigma)$.
5. Der Angreifer hat gewonnen, wenn $m \notin \{m_1, \ldots, m_k\}$ und $\mathcal{V}er_{pk}(m, \sigma) = 1$ ist.

Unfälschbarkeit: Jeder PPT-Angreifer gewinnt nur mit vernachlässigbarer Wahrscheinlichkeit.

**Problem.** Die Signaturen werden vom Benutzer berechnet, nicht vom Signierer.

Was bedeutet dann die Aussage in Schritt 3?

# Spiel zur Unfälschbarkeit

Eingabe: Sicherheitsparameter $1^n$

# Spiel zur Unfälschbarkeit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$

# Spiel zur Unfälschbarkeit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Herausforderer erzeugt einen Schlüssel $(sk, pk)$ und gibt $pk$ an den Angreifer.

# Spiel zur Unfälschbarkeit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Herausforderer erzeugt einen Schlüssel $(sk, pk)$ und gibt $pk$ an den Angreifer.
3. Der Angreifer interagiert mit $\mathcal{S}ign_{sk}$.

# Spiel zur Unfälschbarkeit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Herausforderer erzeugt einen Schlüssel $(sk, pk)$ und gibt $pk$ an den Angreifer.
3. Der Angreifer interagiert mit $\mathcal{S}ign_{sk}$.
4. Der Angreifer berechnet Tupel $(m_1, \sigma_1), \ldots, (m_k, \sigma_k)$.

# Spiel zur Unfälschbarkeit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Herausforderer erzeugt einen Schlüssel $(sk, pk)$ und gibt $pk$ an den Angreifer.
3. Der Angreifer interagiert mit $\mathcal{S}ign_{sk}$.
4. Der Angreifer berechnet Tupel $(m_1, \sigma_1), \ldots, (m_k, \sigma_k)$.
5. Der Angreifer hat gewonnen, wenn $\mathcal{V}er_{pk}(m_i, \sigma_i) = 1$ ist und in Schritt 3 $\mathcal{S}ign_{sk}$ höchstens $(k-1)$-mal terminiert hat.

# Spiel zur Unfälschbarkeit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Herausforderer erzeugt einen Schlüssel $(sk, pk)$ und gibt $pk$ an den Angreifer.
3. Der Angreifer interagiert mit $\mathcal{S}ign_{sk}$.
4. Der Angreifer berechnet Tupel $(m_1, \sigma_1), \ldots, (m_k, \sigma_k)$.
5. Der Angreifer hat gewonnen, wenn $\mathcal{V}er_{pk}(m_i, \sigma_i) = 1$ ist und in Schritt 3 $\mathcal{S}ign_{sk}$ höchstens $(k-1)$-mal terminiert hat.

Unfälschbarkeit: Jeder PPT-Angreifer gewinnt nur mit vernachlässigbarer Wahrscheinlichkeit.

# Spiel zur Blindheit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$

# Spiel zur Blindheit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Angreifer erzeugt einen öffentlichen Schlüssel $pk$ und zwei Nachrichten $m_1, m_2$.

# Spiel zur Blindheit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Angreifer erzeugt einen öffentlichen Schlüssel $pk$ und zwei Nachrichten $m_1, m_2$.
3. Der Herausforderer wählt $i = \{0, 1\}$ uniform zufällig.

# Spiel zur Blindheit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Angreifer erzeugt einen öffentlichen Schlüssel $pk$ und zwei Nachrichten $m_1, m_2$.
3. Der Herausforderer wählt $i = \{0, 1\}$ uniform zufällig.
4. Der Angreifer kann mit $\mathcal{U}ser_{pk}(m_i)$ und mit $\mathcal{U}ser_{pk}(m_{1-i})$ interagieren (Reihenfolge beachten!).

# Spiel zur Blindheit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Angreifer erzeugt einen öffentlichen Schlüssel $pk$ und zwei Nachrichten $m_1, m_2$.
3. Der Herausforderer wählt $i = \{0, 1\}$ uniform zufällig.
4. Der Angreifer kann mit $\mathcal{U}ser_{pk}(m_i)$ und mit $\mathcal{U}ser_{pk}(m_{1-i})$ interagieren (Reihenfolge beachten!).
5. Der Angreifer entscheidet sich für $j \in \{0, 1\}$.

# Spiel zur Blindheit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Angreifer erzeugt einen öffentlichen Schlüssel $pk$ und zwei Nachrichten $m_1, m_2$.
3. Der Herausforderer wählt $i = \{0, 1\}$ uniform zufällig.
4. Der Angreifer kann mit $\mathcal{U}ser_{pk}(m_i)$ und mit $\mathcal{U}ser_{pk}(m_{1-i})$ interagieren (Reihenfolge beachten!).
5. Der Angreifer entscheidet sich für $j \in \{0, 1\}$.
6. Der Angreifer hat gewonnen, wenn $i = j$ ist.

# Spiel zur Blindheit

Eingabe: Sicherheitsparameter $1^n$

1. Herausforderer und Angreifer erhalten $1^n$
2. Der Angreifer erzeugt einen öffentlichen Schlüssel $pk$ und zwei Nachrichten $m_1, m_2$.
3. Der Herausforderer wählt $i = \{0, 1\}$ uniform zufällig.
4. Der Angreifer kann mit $\mathcal{U}ser_{pk}(m_i)$ und mit $\mathcal{U}ser_{pk}(m_{1-i})$ interagieren (Reihenfolge beachten!).
5. Der Angreifer entscheidet sich für $j \in \{0, 1\}$.
6. Der Angreifer hat gewonnen, wenn $i = j$ ist.

Blindheit: Für jeden PPT-Angreifer ist der Vorteil vernachlässigbar.

# Ein einfaches RSA-Schema

**Schlüsselerzeugung.**

Wähle Primzahlen $p, q$,

berechne $N := pq$, $\phi(N) = (p-1)(q-1)$,

wähle $e, d$ mit $ed \equiv 1 \bmod \phi(N)$,

gib aus:

- privater Schlüssel: $N, d$.
- öffentlicher Schlüssel $N, e$.

# Ein einfaches RSA-Schema

**Interaktive (blinde) Signatur**

Sei $m$ eine Nachricht, $m \in \mathbf{Z}/N\mathbf{Z}$.

# Ein einfaches RSA-Schema

**Interaktive (blinde) Signatur**

Sei $m$ eine Nachricht, $m \in \mathbf{Z}/N\mathbf{Z}$.

Der Benutzer wählt $r \in \mathbf{Z}/N\mathbf{Z}$ und berechnet $c := r^e \cdot m \in \mathbf{Z}/N\mathbf{Z}$.

# Ein einfaches RSA-Schema

**Interaktive (blinde) Signatur**

Sei $m$ eine Nachricht, $m \in \mathbf{Z}/N\mathbf{Z}$.

Der Benutzer wählt $r \in \mathbf{Z}/N\mathbf{Z}$ und berechnet $c := r^e \cdot m \in \mathbf{Z}/N\mathbf{Z}$.

Der Signierer berechnet $s := c^d \in \mathbf{Z}/N\mathbf{Z}$.
[Es ist nun $s = (r^{ed} \cdot m^d) = r \cdot m^d$.]

# Ein einfaches RSA-Schema

**Interaktive (blinde) Signatur**

Sei $m$ eine Nachricht, $m \in \mathbf{Z}/N\mathbf{Z}$.

Der Benutzer wählt $r \in \mathbf{Z}/N\mathbf{Z}$ und berechnet
$c := r^e \cdot m \in \mathbf{Z}/N\mathbf{Z}$.

Der Signierer berechnet $s := c^d \in \mathbf{Z}/N\mathbf{Z}$.
[Es ist nun $s = (r^{ed} \cdot m^d) = r \cdot m^d$.]

Der Benutzer berechnet $\sigma := s \cdot r^{-1} \in \mathbf{Z}/N\mathbf{Z}$, gibt $\sigma$ aus.
[Es ist nun $\sigma = m^d$.]

# Ein einfaches RSA-Schema

**Interaktive (blinde) Signatur**

Sei $m$ eine Nachricht, $m \in \mathbf{Z}/N\mathbf{Z}$.

Der Benutzer wählt $r \in \mathbf{Z}/N\mathbf{Z}$ und berechnet
$c := r^e \cdot m \in \mathbf{Z}/N\mathbf{Z}$.

Der Signierer berechnet $s := c^d \in \mathbf{Z}/N\mathbf{Z}$.
[Es ist nun $s = (r^{ed} \cdot m^d) = r \cdot m^d$.]

Der Benutzer berechnet $\sigma := s \cdot r^{-1} \in \mathbf{Z}/N\mathbf{Z}$, gibt $\sigma$ aus.
[Es ist nun $\sigma = m^d$.]

**Verifikation**

$\sigma^e \stackrel{?}{=} m \in \mathbf{Z}/N\mathbf{Z}$.

# Ein einfaches RSA-Schema

**Interaktive (blinde) Signatur**

Sei $m$ eine Nachricht, $m \in \mathbf{Z}/N\mathbf{Z}$.

Der Benutzer wählt $r \in \mathbf{Z}/N\mathbf{Z}$ und berechnet
$c := r^e \cdot m \in \mathbf{Z}/N\mathbf{Z}$.

Der Signierer berechnet $s := c^d \in \mathbf{Z}/N\mathbf{Z}$.
[Es ist nun $s = (r^{ed} \cdot m^d) = r \cdot m^d$.]

Der Benutzer berechnet $\sigma := s \cdot r^{-1} \in \mathbf{Z}/N\mathbf{Z}$, gibt $\sigma$ aus.
[Es ist nun $\sigma = m^d$.]

**Verifikation**

$\sigma^e \stackrel{?}{=} m \in \mathbf{Z}/N\mathbf{Z}$.

**Sicherheit**

Blindheit: ✓

# Ein einfaches RSA-Schema

**Interaktive (blinde) Signatur**

Sei $m$ eine Nachricht, $m \in \mathbf{Z}/N\mathbf{Z}$.

Der Benutzer wählt $r \in \mathbf{Z}/N\mathbf{Z}$ und berechnet $c := r^e \cdot m \in \mathbf{Z}/N\mathbf{Z}$.

Der Signierer berechnet $s := c^d \in \mathbf{Z}/N\mathbf{Z}$.
[Es ist nun $s = (r^{ed} \cdot m^d) = r \cdot m^d$.]

Der Benutzer berechnet $\sigma := s \cdot r^{-1} \in \mathbf{Z}/N\mathbf{Z}$, gibt $\sigma$ aus.
[Es ist nun $\sigma = m^d$.]

**Verifikation**

$\sigma^e \stackrel{?}{=} m \in \mathbf{Z}/N\mathbf{Z}$.

**Sicherheit**

Blindheit: ✓

Unfälschbarkeit: nicht erfüllt, da $(m, \sigma)$ und $(c, s)$ signierte Nachrichten sind.

# Ein einfaches RSA-Schema

**Interaktive (blinde) Signatur**

Sei $H$ eine Hash-Funktion.
Sei $m$ eine Nachricht, $m \in \mathbf{Z}/N\mathbf{Z}$.

# Ein einfaches RSA-Schema

**Interaktive (blinde) Signatur**

Sei $H$ eine Hash-Funktion.
Sei $m$ eine Nachricht, $m \in \mathbf{Z}/N\mathbf{Z}$.

Der Benutzer wählt $r \in \mathbf{Z}/N\mathbf{Z}$ und berechnet
$c := r^e \cdot H(m) \in \mathbf{Z}/N\mathbf{Z}$.

# Ein einfaches RSA-Schema

**Interaktive (blinde) Signatur**

Sei $H$ eine Hash-Funktion.
Sei $m$ eine Nachricht, $m \in \mathbf{Z}/N\mathbf{Z}$.

Der Benutzer wählt $r \in \mathbf{Z}/N\mathbf{Z}$ und berechnet
$c := r^e \cdot H(m) \in \mathbf{Z}/N\mathbf{Z}$.

Der Signierer berechnet $s := c^d \in \mathbf{Z}/N\mathbf{Z}$.
[Es ist nun $s = r \cdot H(m)^d$.]

# Ein einfaches RSA-Schema

**Interaktive (blinde) Signatur**

Sei $H$ eine Hash-Funktion.
Sei $m$ eine Nachricht, $m \in \mathbf{Z}/N\mathbf{Z}$.

Der Benutzer wählt $r \in \mathbf{Z}/N\mathbf{Z}$ und berechnet
$c := r^e \cdot H(m) \in \mathbf{Z}/N\mathbf{Z}$.

Der Signierer berechnet $s := c^d \in \mathbf{Z}/N\mathbf{Z}$.
[Es ist nun $s = r \cdot H(m)^d$.]

Der Benutzer berechnet $\sigma := s \cdot r^{-1} \in \mathbf{Z}/N\mathbf{Z}$, gibt $\sigma$ aus.
[Es ist nun $\sigma = H(m)^d$.]

# Ein einfaches RSA-Schema

**Interaktive (blinde) Signatur**

Sei $H$ eine Hash-Funktion.
Sei $m$ eine Nachricht, $m \in \mathbf{Z}/N\mathbf{Z}$.

Der Benutzer wählt $r \in \mathbf{Z}/N\mathbf{Z}$ und berechnet
$c := r^e \cdot H(m) \in \mathbf{Z}/N\mathbf{Z}$.

Der Signierer berechnet $s := c^d \in \mathbf{Z}/N\mathbf{Z}$.
[Es ist nun $s = r \cdot H(m)^d$.]

Der Benutzer berechnet $\sigma := s \cdot r^{-1} \in \mathbf{Z}/N\mathbf{Z}$, gibt $\sigma$ aus.
[Es ist nun $\sigma = H(m)^d$.]

**Verifikation**

$\sigma^e \stackrel{?}{=} H(m) \in \mathbf{Z}/N\mathbf{Z}$.

# Ein einfaches RSA-Schema

**Interaktive (blinde) Signatur**

Sei $H$ eine Hash-Funktion.
Sei $m$ eine Nachricht, $m \in \mathbf{Z}/N\mathbf{Z}$.

Der Benutzer wählt $r \in \mathbf{Z}/N\mathbf{Z}$ und berechnet
$c := r^e \cdot H(m) \in \mathbf{Z}/N\mathbf{Z}$.

Der Signierer berechnet $s := c^d \in \mathbf{Z}/N\mathbf{Z}$.
[Es ist nun $s = r \cdot H(m)^d$.]

Der Benutzer berechnet $\sigma := s \cdot r^{-1} \in \mathbf{Z}/N\mathbf{Z}$, gibt $\sigma$ aus.
[Es ist nun $\sigma = H(m)^d$.]

**Verifikation**

$\sigma^e \stackrel{?}{=} H(m) \in \mathbf{Z}/N\mathbf{Z}$.

**Sicherheit**

Blindheit: ✓

Unfälschbarkeit: OK im Random Oracle Modell

# Digitale Münzen

# Digitale Münzen

**Setup.** Alice und Bob haben Konten mit echtem Geld bei einer Bank.

# Digitale Münzen

**Setup.** Alice und Bob haben Konten mit echtem Geld bei einer Bank.

**Ziel.**

Die Bank gibt Alice “elektronische Münzen” gegen echtes Geld, die diese z.B. an Bob weiterreichen kann.

Bob kann diese einreichen und erhält dafür dann wieder echtes Geld.

# Digitale Münzen

**... mit blinden Signaturen**

# Digitale Münzen

## ... mit blinden Signaturen

- Alice wählt eine (lange) “Seriennummer” und “geht mit dieser zur Bank”

# Digitale Münzen

## ... mit blinden Signaturen

- Alice wählt eine (lange) “Seriennummer” und “geht mit dieser zur Bank”
- Die Bank unterschreibt die Seriennummer blind.

# Digitale Münzen

**... mit blinden Signaturen**

- Alice wählt eine (lange) “Seriennummer” und “geht mit dieser zur Bank”
- Die Bank unterschreibt die Seriennummer blind.
- $\longrightarrow$ Alice hat eine unterschriebene Seriennummer, eine Münze.

# Digitale Münzen

## ... mit blinden Signaturen

- Alice wählt eine (lange) "Seriennummer" und "geht mit dieser zur Bank"
- Die Bank unterschreibt die Seriennummer blind.
- $\longrightarrow$ Alice hat eine unterschriebene Seriennummer, eine Münze.
- Die Bank bucht dafür echtes Geld von Alice' Konto ab.

# Digitale Münzen

## ... mit blinden Signaturen

- Alice wählt eine (lange) "Seriennummer" und "geht mit dieser zur Bank"
- Die Bank unterschreibt die Seriennummer blind.
- $\longrightarrow$ Alice hat eine unterschriebene Seriennummer, eine Münze.
- Die Bank bucht dafür echtes Geld von Alice' Konto ab.
- Alice sendet die (Daten der) Münze an Bob.

# Digitale Münzen

## ... mit blinden Signaturen

- Alice wählt eine (lange) "Seriennummer" und "geht mit dieser zur Bank"
- Die Bank unterschreibt die Seriennummer blind.
- $\longrightarrow$ Alice hat eine unterschriebene Seriennummer, eine Münze.
- Die Bank bucht dafür echtes Geld von Alice' Konto ab.
- Alice sendet die (Daten der) Münze an Bob.
- Bob "geht mit der Münze zur Bank".

# Digitale Münzen

## ... mit blinden Signaturen

- Alice wählt eine (lange) "Seriennummer" und "geht mit dieser zur Bank"
- Die Bank unterschreibt die Seriennummer blind.
- $\longrightarrow$ Alice hat eine unterschriebene Seriennummer, eine Münze.
- Die Bank bucht dafür echtes Geld von Alice' Konto ab.
- Alice sendet die (Daten der) Münze an Bob.
- Bob "geht mit der Münze zur Bank".
- Die Bank speichert die nun verbrauchte Seriennummer.

# Digitale Münzen

## ... mit blinden Signaturen

- Alice wählt eine (lange) "Seriennummer" und "geht mit dieser zur Bank"
- Die Bank unterschreibt die Seriennummer blind.
- $\longrightarrow$ Alice hat eine unterschriebene Seriennummer, eine Münze.
- Die Bank bucht dafür echtes Geld von Alice' Konto ab.
- Alice sendet die (Daten der) Münze an Bob.
- Bob "geht mit der Münze zur Bank".
- Die Bank speichert die nun verbrauchte Seriennummer.
- Bob wird echtes Geld gutgeschrieben.

# Digitale Münzen

**... mit blinden Signaturen**

- Alice wählt eine (lange) "Seriennummer" und "geht mit dieser zur Bank"
- Die Bank unterschreibt die Seriennummer blind.
- $\longrightarrow$ Alice hat eine unterschriebene Seriennummer, eine Münze.
- Die Bank bucht dafür echtes Geld von Alice' Konto ab.
- Alice sendet die (Daten der) Münze an Bob.
- Bob "geht mit der Münze zur Bank".
- Die Bank speichert die nun verbrauchte Seriennummer.
- Bob wird echtes Geld gutgeschrieben.

Mit verschiedenen Bankschlüsseln kann man auch Münzen mit unterschiedlichem Wert erzeugen.

# Digitale Münzen

- Alice ist anonym.

# Digitale Münzen

- Alice ist anonym.
- Wenn Alice ihre Münze zweimal weitergibt, fällt das auf, aber ein Händler kann nur sicher sein, wenn er die Münze sofort einlöst ...

# Digitale Münzen

- Alice ist anonym.
- Wenn Alice ihre Münze zweimal weitergibt, fällt das auf, aber ein Händler kann nur sicher sein, wenn er die Münze sofort einlöst ...

Mit einer (nicht so leichten) Variante des RSA-Schemas:

- Wenn Alice beim Schummeln erwischt wird, ist sie (mit großer Wahrscheinlichkeit) nicht mehr anonym.

David Chaum, Amos Fiat und Moni Naor: Untraceable Electronic Cash (Crypto 1988)

# Kurze Geschichte

1982: David Chaum: “Blind signatures for untraceable payments”

1988: David Chaum, Amos Fiat und Moni Naor:
“Untraceable Electronic Cash”

# Kurze Geschichte

1982: David Chaum: “Blind signatures for untraceable payments”

1988: David Chaum, Amos Fiat und Moni Naor:
“Untraceable Electronic Cash”

1990: DigiCash, gegründet von David Chaum mit Produkt ecash

# Kurze Geschichte

1982: David Chaum: “Blind signatures for untraceable payments”

1988: David Chaum, Amos Fiat und Moni Naor:
“Untraceable Electronic Cash”

1990: DigiCash, gegründet von David Chaum mit Produkt ecash

1998: ecash von Deutscher Bank und anderen Banken

# Kurze Geschichte

1982: David Chaum: “Blind signatures for untraceable payments”

1988: David Chaum, Amos Fiat und Moni Naor:
“Untraceable Electronic Cash”

1990: DigiCash, gegründet von David Chaum mit Produkt ecash

1998: ecash von Deutscher Bank und anderen Banken

Insolvenz von DigiCash

# Kurze Geschichte

1982: David Chaum: “Blind signatures for untraceable payments”

1988: David Chaum, Amos Fiat und Moni Naor:
“Untraceable Electronic Cash”

1990: DigiCash, gegründet von David Chaum mit Produkt ecash

1998: ecash von Deutscher Bank und anderen Banken

Insolvenz von DigiCash

2014: gnu Taler, initiiert von Christian Grothoff am Inria in Rennes

Ziel: ein transparentes System mit Anonymität für Kundern aber Möglichkeit der Besteuerung von Händlern

(Homepage mit Demo, ausprobieren!)

# Kurze Geschichte

1982: David Chaum: “Blind signatures for untraceable payments”

1988: David Chaum, Amos Fiat und Moni Naor:
“Untraceable Electronic Cash”

1990: DigiCash, gegründet von David Chaum mit Produkt ecash

1998: ecash von Deutscher Bank und anderen Banken

Insolvenz von DigiCash

2014: gnu Taler, initiiert von Christian Grothoff am Inria in Rennes

Ziel: ein transparentes System mit Anonymität für Kundern
aber Möglichkeit der Besteuerung von Händlern

(Homepage mit Demo, ausprobieren!)

Informationen teilweise nach wikipedia Artikel “ecash”