Elektronische Bezahlsysteme – Teil II

	Kunde		Bank
Seriennummer:	$x \in \{0,1\}^{160}$		
Blinding Factor:	$k \in \mathbb{Z}_N{}^+,\ \mathrm{ggT}\,(k, N) = 1$		
	$m = x \cdot k^d \bmod N$	$\xrightarrow{m + \mathrm{Sig}_{\mathrm{Kunde}}(m)}$	$s = m^e \bmod N$
	$c = s/k \bmod N$	$\xleftarrow{s}$	Kundenkonto belasten

Korrektheit der Signatur prüfen:

$c^d \bmod N \stackrel{?}{=} x$

Danach Ablage der Münze in der digitalen Brieftasche

eCash

Elektronisches Bargeld – eCash

- Das nachfolgend beschriebene Verfahren für elektronisches Bargeld wurde von David Chaum entwickelt, dessen Firma *DigiCash* das zugehörige Patent innehatte. Durch Insolvenz von *DigiCash* konnte die Verbreitung von eCash jedoch nicht vorangetrieben werden.

- Anders als aktuelle elektronische Währungen basiert es nicht auf verteilten Systemen (Peer-to-Peer).
- Das Verfahren gilt bis heute als sehr sicher, während der gesamten Handelszeit mit eCash gab es keinen (aufgedeckten) Betrugsversuch.

eCash – Überblick

- Elektronische Münzen in eCash können als **signierte Wertmarken** mit **Seriennummern** verstanden werden. Sie werden mittels einer geeigneten Software (*eWallet*) beim Kunden verwaltet.

- Einzelne Münzen werden durch den Kunden selbst erzeugt. Sie erhalten dabei eine eindeutige **Seriennummer**.

- Diese Münzen lässt sich der Kunde durch die Bank signieren, ohne dass diese die Seriennummer erfährt (***blind signature***). Für jede signierte Münze erhält die Bank vom Kunden Echtgeld in Höhe des Nennwertes der Münze.

- Für jeden möglichen **Nennwert** (1$, 2$, 5$, etc.) der Münze nutzt die Bank einen eigenen Schlüssel. So kann bereits an der **Signatur** der Münze ihr Nennwert abgelesen werden.

eCash – Überblick

- Beim **Bezahlvorgang** werden die Münzen durch den Kunden mit dem öffentlichen Schlüssel der Bank verschlüsselt.
- Der **Händler** erhält also nur **verschlüsselte** Münzen, die er durch die signierende Bank online prüfen lässt. Diese kann mit ihrem privaten Schlüssel die Münze entschlüsseln, die Seriennummer entnehmen und diese in einer Datenbank ablegen.
- Wurde die Seriennummer vorher noch nicht verwendet, gibt die Bank die Zahlung frei und der Händler kann liefern.

eCash – Überblick

- Die Münzen können **beliebig** oft **kopiert** (Backups) oder elektronisch versendet werden. Entscheidend ist nur, dass durch die **Onlineprüfung** die Mehrfachverwendung ausgeschlossen ist.
- Solange kein Betrug vorliegt, bleibt der Kunde bei der Bezahlung gegenüber Händler und Bank **anonym**.
- Sobald jedoch eine Münze mehrfach zum Bezahlen verwendet wurde, kann die Bank den Betrug **aufdecken** und unter Zuhilfenahme eines dafür entworfenen Protokolls die **Identität** des Kunden aufdecken.

eCash im Detail: Blind Signatures

- Jede eCash-Münze hat eine **eindeutige Seriennummer**, die vom Kunden erzeugt wird.
- Die Bank soll die Münzen signieren, ohne jedoch die Seriennummern zu kennen. Bei einer späteren Zahlung kann die Bank eine signierte Münze nur auf **Korrektheit** prüfen, nicht aber die Seriennummer einem Kunden zuordnen. So wird die **Anonymität** des Kunden im Zahlungsverkehr gesichert.
- Diese blinde Signatur wird dadurch erreicht, dass der Kunde die zu signierende Nachricht auf eine Art verschleiert, die er später bei der signierten Nachricht wieder rückgängig machen kann, ohne dazu den privaten Schlüssel der Bank kennen zu müssen.

Blind Signatures mit RSA

Voraussetzungen:

- A möchte, dass B eine Nachricht m blind unterschreibt.
- B besitzt den öffentlichen Schlüssel (d, N) und den privaten Schlüssel (e, N).

Verfahren:

- A wählt einen *Blinding Factor* $k \in \mathbb{N}$ mit $1 < k < N$ und $ggT(k, N) = 1$.
- Er verschleiert m mit Hilfe des öffentlichen Schlüssels von B: $m' = m \cdot k^d \bmod N$.

Blind Signatures mit RSA

- B signiert m‘: $s' = (m')^e \bmod N = (m \cdot k^d)^e \bmod N$
  $= (m^e \cdot k) \bmod N$

- A entschleiert: $s = s'/k = \frac{(m^e \cdot k) \bmod N}{k}$
  $= (m^e \cdot k)/k \bmod N = m^e \bmod N$ *)

- A erhält also für die Nachricht m von B eine Signatur, ohne dass B die Nachricht selbst hätte ermitteln können.

*) Dies gilt, weil k und N teilerfremd sind!

Blind Singatures bei eCash

- Der Kunde erzeugt „Roh“-Münzen (sog. Kandidaten) auf seinem Rechner, die er von seiner Bank signieren lassen will.
- Dazu verschleiert der Kunde die Rohmünzen mit einem *Blinding Factor* und signiert diese verschleierten Münzen mit seinem privaten Schlüssel.
- Die Nachricht aus verschleierten Münzen und deren Signatur wird an die Bank übertragen.
- Durch die Signatur des Kunden weiß die Bank, dass die Münzen auf dem Übertragungsweg nicht verändert wurden und tatsächlich dem Kunden zuzuordnen sind. Sie kann nun die verschleierten Münzen mit ihrem privaten Schlüssel signieren und an den Kunden zurückleiten.

Blind Singatures bei eCash

- Der Kunde kann die empfangenen signierten verschleierten Münzen mit dem nur ihm bekannten *Blinding Factor* entschleiern.
- Parallel wird der Nennbetrag der Münzen durch die Bank vom Kundenkonto abgebucht.

Double-Spending

Problem: Münzen könnten prinzipiell mehrfach zur Bezahlung eingesetzt werden (*double spending*).

- In einem (verzögerungsfreien) **Onlinesystem** kannn *double spending* leicht erkannt und vermieden werden, da die Münzen durch eine eindeutige Seriennummer gekennzeichnet sind und jede Transaktion durch die Bank bestätigt wird.

- Wenn aber der Bezahlvorgang **offline** erfolgt, liegt zwischen Bezahlung beim Händler und Einlösen bei der Bank eine größere Zeitspanne, *double spending* wird also möglich.

- In diesem Fall muss dafür Sorge getragen werden, dass ebendieses *double-spending* **im Nachhinein aufgedeckt** wird und der betrügende Kunde zur Rechenschaft gezogen werden kann.

Double-Spending

Offlinemethode 1: Es wird eine fälschungs- und veränderungssichere elektronische Brieftasche (z.B. mit Smartcard) verwendet. Diese allerdings

- sind ggf. anfällig für Angriffe
- werden erfahrungsgemäß nicht von den Nutzern angenommen.

Offlinemethode 2: Ein Protokoll das den Nutzer bei *double spending* identifiziert und sonst dessen Anonymität wahrt.

Double Spending Protocol – Schema

Das ***Chaum Double Spending Protocol*** bietet genau diese Funktionalität:

- Teile der **Identität** des Kunden werden in seinen Münzen gespeichert.
- Die **Bank** stellt bei der Signatur der Münzen sicher, dass der Kunde bei der Identitätsangabe **ehrlich** ist.
- Jedes Mal, wenn eine Münze ausgegeben wird, fordert der Händler einen **Teil** der **Kundenidentität** an.
- Löst der Händler die Münze bei der Bank ein und existiert bereits eine andere Einlösung **derselben** Münze, so kann die Identität des Kunden durch die Bank **aufgedeckt** werden.

Voraussetzung 1: Cut-and-Choose

Problem: Die Bank muss sicherstellen, dass der Kunde in all seine Münzen Teile seiner Identität einprägt, bevor die Bank die Münzen signiert. Aber: Die Münzen sind verschleiert, wenn die Bank sie signiert, d.h. die Bank kann dies nicht direkt prüfen.

Lösung: Der Kunde erzeugt die doppelte Anzahl an Münzen. Die Bank wählt zufällig die Hälfte der Münzen zur Bezahlung aus und signiert diese. Für die **andere Hälfte** fordert die Bank die *Blinding-Factors* vom Kunden ab und entschleiert die Münzen. So kann sie prüfen, ob der Kunde dort die Identitätsinformationen eingeprägt hat.

Da die Bank die Auswahl trifft, würde ein Betrugsversuch an dieser Stelle mit hoher Wahrscheinlichkeit aufgedeckt werden.

Voraussetzung 2: Addition Modulo 2

Bei der Addition modulo 2 gilt: $x \oplus x = 0$

$x \oplus y$	y: 0	y: 1
x: 0	**0**	1
x: 1	1	**0**

Diese Eigenschaft kann man sich für ein **symmetrisches Verschlüsselungsschema** zunutze machen:

- Der **geheime Schlüssel** ist eine Zufallszahl r fester Länge.
- Eine Nachricht m mit derselben Länge wie r wird **verschlüsselt**, indem $c = m \oplus r$ gebildet wird.
- **Entschlüsselt** wird mit $c \oplus r = m \oplus \boldsymbol{r} \oplus \boldsymbol{r} = m \oplus 0 = m$.
- Da ein Angreifer den geheimen Schlüssel r nicht kennt, kann er von c nicht auf m schließen.

Double Spending Protocol – Erläuterung am Beispiel

- Zur Identifikation wird die Kontonummer verwendet, ihr Wert (*acct*) sei 12 (0x0C, bzw. = 00001100_2).
- Der Kunde erzeugt nun b Zufallszahlen (im Beispiel $b = 6$) und bildet $s_i = acct \oplus b_i$.
- Diese Information (also die Menge aller s_i) wird in jede Münze eingeprägt, bevor sie durch die Bank signiert wird.

i	acct	b_i	$s_i = acct \oplus b_i$
0	0C	1B	17
1	0C	13	1F
2	0C	09	05
3	0C	05	09
4	0C	2B	27
5	0C	11	1D

Double Spending Protocol – Erläuterung am Beispiel

- Bei jedem Bezahlvorgang erhält der Händler vom Kunden den Wert für b und erzeugt eine zufällige b-Bit Zahl (z.B. 111010_2)
- Für jede Bitposition, an der diese b-Bit Zahl den Wert 1 besitzt, sendet der Kunde die Zufallszahl b_i dieser Position
- Für alle anderen Stellen sendet er den Wert s_i.
- Weder Händler noch Bank kann aus den erhaltenen b_i oder s_i den Wert von $acct$ bestimmen.

Double Spending Protocol – Erläuterung am Beispiel

i	acct	b_i	$s_i = acct \oplus b_i$	Händler würfelt	**Händler erhält**
0	0D	1B	17	0	17
1	0D	13	1F	1	13
2	0D	09	05	0	05
3	0D	05	09	1	05
4	0D	2B	27	1	2B
5	0D	11	1D	1	11

Double Spending Protocol – Erläuterung am Beispiel

- Wird dieselbe Münze bei einem **weiteren Händler** eingesetzt, würfelt dieser an mit hoher Wahrscheinlichkeit an mindestens einer der b Stellen einen anderen Wert.
- Nehmen wir im Beispiel an, dass er 000010 würfelt.
- Damit erhält dieser Händler vom Kunden folgende Werte:

i	acct	b_i	$s_i = acct \oplus b_i$	Händler 2 würfelt	**Händler 2 erhält**
0	0D	1B	17	0	17
1	0D	13	1F	0	1F
2	0D	09	05	0	05
3	0D	05	09	0	09
4	0D	2B	27	1	2B
5	0D	11	1D	0	1D

Double Spending Protocol – Erläuterung am Beispiel

Wenn **beide** Münzen bei der **Bank eingereicht** werden, gibt es nun mindestens eine Zeile, für die sowohl s_i als auch b_i bekannt ist. Nun kann die Bank durch simples Aufaddieren modulo 2 die Kundeninformation entschleiern, denn es gilt:

$$s_i \oplus b_i = (acct \oplus b_i) \oplus b_i = acct$$

i	acct	b_i	$s_i = acct \oplus b_i$	**Von Händler 1**	**Von Händler 2**	**Summe mod 2**
0	0D	1B	17	17	17	17 ⊕ 17 = 0
1	0D	13	1F	13	1F	**13 ⊕ 1F = 0D**
2	0D	09	05	05	05	05 ⊕ 05 = 0
3	0D	05	09	05	09	**05 ⊕ 09 = 0D**
4	0D	2B	27	2B	2B	2B ⊕ 2B = 0
5	0D	11	1D	11	1D	**11 ⊕ 1D = 0D**

Double Spending Protocol – Erläuterung am Beispiel

Hinweise:

1. Wir haben aus Gründen der Übersichtlichkeit und Verständlichkeit das Protokoll an einigen Stellen simplifiziert. Die Frage, wie sichergestellt ist, dass der Kunde den Händler nicht bei der Angabe der s_i bzw. b_i betrügt, müssen wir offenlassen. Das Originalprotokoll hat hier entsprechend vorgesorgt. Für weitere Informationen:

   http://dl.acm.org/citation.cfm?id=88969

2. Das Aufdecken der Identität des Kunden geschieht bei der **Bank** und auch nur dann, wenn zwei **identische** Münzen ausgegeben werden!

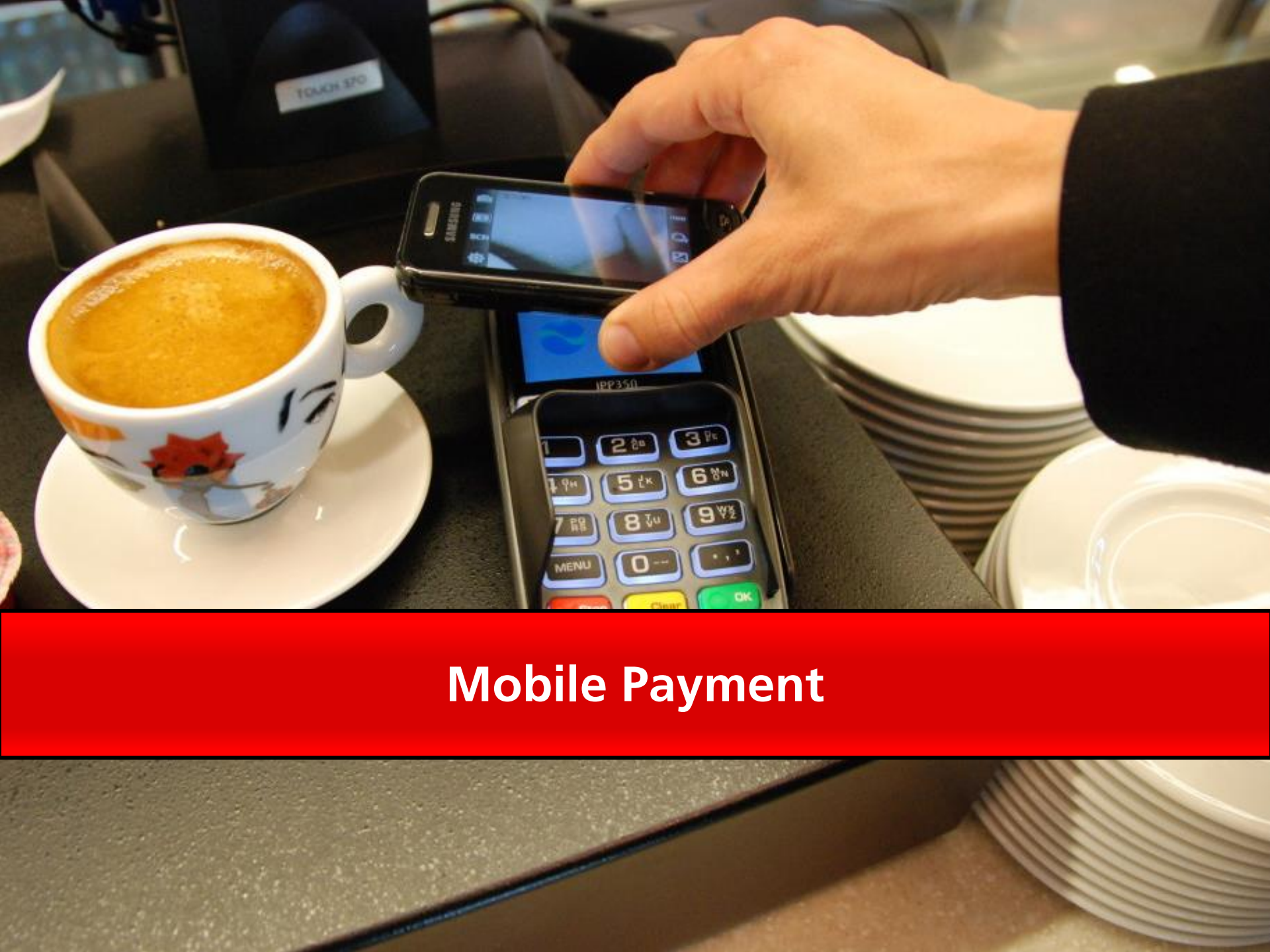
Double Spending Protocol – Aufdeckwahrscheinlichkeit

- Damit ein betrügerischer Kunde nicht beim *double-spending* entdeckt wird, müssten beide Händler dieselbe b -stellige Zufallszahl würfeln.

- Die Wahrscheinlichkeit, dass ein gleichverteilter Zufallsgenerator bei zwei Würfen dieselbe b-Bit Zahl würfelt, ist $p(b) = \frac{1}{2^b} = 2^{-b}$

$$
\begin{aligned}
p(1) &= 0.5 \\
p(10) &\approx .001 \\
p(20) &\approx 1/1{,}000{,}000 \\
p(30) &\approx 1/1{,}000{,}000{,}000 \\
p(64) &\approx 5 \cdot 10^{-20} \\
p(128) &\approx 3 \cdot 10^{-39}
\end{aligned}
$$

- Das Protokoll **garantiert nicht** die Detektion von *double-spending*, macht es aber praktisch sinnlos.


Mobile Payment

Mobile Payment

Hinweis: Bezahlung unter Einbeziehung von Mobiltechnologien ist technisch betrachtet keine nennenswerte Herausforderung.

Der Grund dafür ist, dass bei allen aktuellen Verfahren auf die **Anonymität** des Kunden gegenüber der Bank verzichtet wird, der Kunde identifiziert sich also stets gegenüber der Bank. Dadurch und durch die Existenz bestehender Verfahren (SSL, TLS) zur sicheren Übertragung der Bezahldaten zwischen Bank und Mobilgerät sind keine komplexen Verfahren zur Bestätigung einer Zahlung notwendig.

Daher erfolgt an dieser Stelle nur ein kurzer Überblick über die grundsätzlichen Verfahren und die Situation in Deutschland.

Mobile Payment

- Eine grobe **Klassifizierung** ist zwischen Verfahren mit zusätzlicher Hardwareunterstützung und reinen Softwarelösungen sinnvoll.
- Feiner kann weiterhin klassifiziert werden, ob die **Gerätekennung** im Mobilfunknetz zur Identifizierung herangezogen wird (bspw. bei Bezahlung via Smartphone) oder nicht.
- Zur Zeit hat sich in Deutschland noch kein Standard durchgesetzt, es gibt parallel und unabhängig agierende Anbieter.

Mobile Payment

- Das Grundprinzip ist dennoch bei allen Anbietern vergleichbar:
  - Der Kunde autorisiert eine Zahlung. Dies geschieht entweder mittels PIN und/oder TAN (sofern die Geräte-ID den Kunden bereits identifiziert) oder durch eine Kombination aus Benutzernamen und Passwort.
  - Die Bank bestätigt die Zahlung online bspw. mittels digitaler Signatur.
  - Der Händler erhält die Bestätigung der Bank entweder durch elektronische Datenübertragung oder mittels lokaler Datenübertragung vom Gerät des Kunden.

Mobile Payment

- **Varianten:**
  - Lesegerät (bspw. per QR-Code, der auf dem Gerät des Kunden angezeigt wird).
  - *Near-Field-Communication* (NFC), also standardisierte Funkübertragung auf sehr kurze Distanzen.

- Händler mit Mobile-Payment unterstützen oft mehrere Bezahlanbieter, sodass der Kunde jeweils seinen eigenen Anbieter nutzen kann.

Elektronische Bezahlsysteme – Zusammenfassung

- SET beschränkt die den Beteiligten zugänglichen Informationen auf das Notwendige und erreicht trotzdem eine für alle akzeptable Garantie der Korrektheit des Bezahlvorgangs.
- Elektronische Münzen im Sinne von *eCash* würden anonyme Bezahlung im Internet erlauben, haben sich jedoch bisher nicht durchsetzen können.
- Mobile Bezahlung in Deutschland noch nicht standardisiert, viele Anbieter existieren parallel. Dennoch erste Konvergenz der Verfahren durch Übertragungsstandards wie *Near-Field-Communication* (NFC).