Formal Analysis of E-Cash Protocols

Jannik Dreier[1], Ali Kassem[2] and Pascal Lafourcade[3]

[1]Institute of Information Security, ETH Zurich
[2]Université Grenoble Alpes, CNRS, VERIMAG
[3]University d'Auvergne, LIMOS

12th International Conference on Security and Cryptography
(SECRYPT 2015), Colmar

July 20, 2015

(Electronic) Cash

(Electronic) Cash

Electronic Cash = digital equivalent

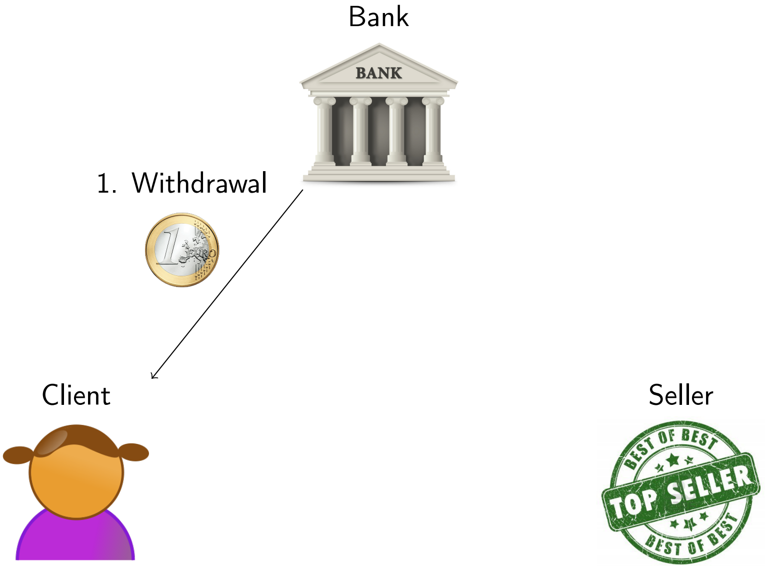
(E-)Cash: Players and Phases

Bank

Client

Seller

(E-)Cash: Players and Phases



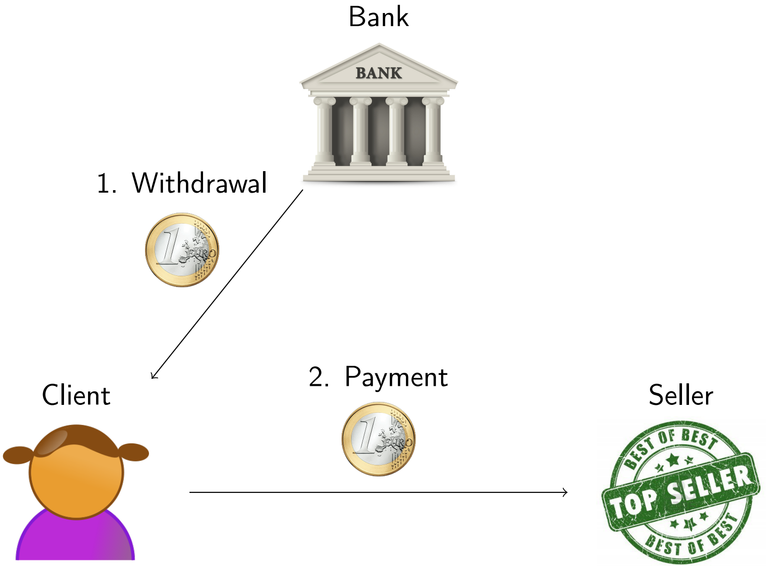
Bank

1. Withdrawal

Client

Seller

(E-)Cash: Players and Phases



Bank

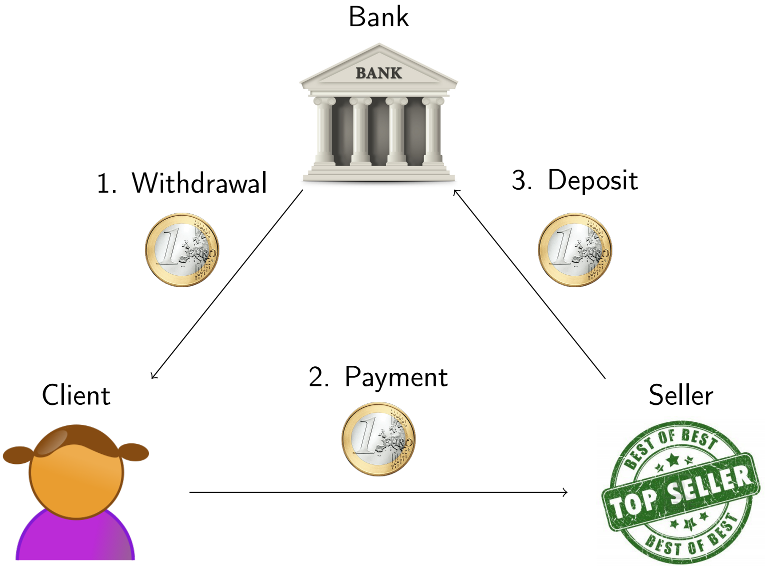
1. Withdrawal

Client

2. Payment

Seller

(E-)Cash: Players and Phases



Bank

1. Withdrawal

3. Deposit

Client

2. Payment

Seller

Security properties of physical cash

- **Unforgeability**: Only the bank can create coins.
- **Anonymity**:
    - *Weak Anonymity*: Nobody can distinguish which client makes a payment.
    - *Strong Anonymity*: Nobody is able to decide whether two payments were made by the same client.

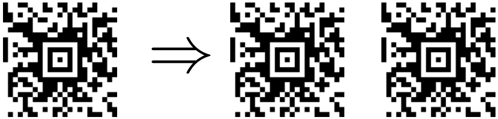
Security properties of physical cash

- **Unforgeability**: Only the bank can create coins.
- **Anonymity**:
  - *Weak Anonymity*: Nobody can distinguish which client makes a payment.
  - *Strong Anonymity*: Nobody is able to decide whether two payments were made by the same client.

- **Do they really hold?**

Security properties of electronic cash

Electronic coins can be **copied**:



Two **additional properties**:

- **Double Spending Identification**: If a client spends a coin twice, his identity is revealed.
- **Exculpability**: An attacker cannot forge a double spend by a client to blame him.

Electronic Cash vs. Electronic Payments

PayPal
VISA
Pay
MasterCard
electronic cash
Google wallet
GeldKarte

Electronic Cash vs. Electronic Payments

PayPal
electronic cash
VISA
Google wallet
Pay
MasterCard
GeldKarte

⇒ **No strong anonymity!**

Contributions

- General **formal framework** for the verification of E-Cash protocols:
  - **Formal model** in the applied π-calculus [**?**]
  - **Formal definitions** of the security properties
  - Suitable for **automated verification** using ProVerif [**?**]
- Three **case studies**:
  - Chaum's On-Line Protocol [**?**]
  - digicash Protocol [**?**]
  - Chaum's Off-Line Protocol [**?**]

Plan

Introduction

Model and Properties

- Forgery-Related Properties
- Anonymity Properties

Case Studies

- Chaum's On-Line Protocol
- DigiCash Protocol
- Chaum's Off-Line Protocol

Conclusion

Plan

Introduction

Model and Properties

- Forgery-Related Properties
- Anonymity Properties

Case Studies

- Chaum's On-Line Protocol
- DigiCash Protocol
- Chaum's Off-Line Protocol

Conclusion

Model

- **Processes** in the applied π-calculus [?]
- Annotated using two **events**:
  - *withdraw*( ) at the bank
  - *spend*( ) at the seller
- **Unforgeability** as **correspondence** between events
- **Anonymity** properties as **observational equivalence** between instances
- **Automatic** verification using ProVerif [?]

Plan

Introduction

Model and Properties

- Forgery-Related Properties
- Anonymity Properties

Case Studies

- Chaum's On-Line Protocol
- DigiCash Protocol
- Chaum's Off-Line Protocol

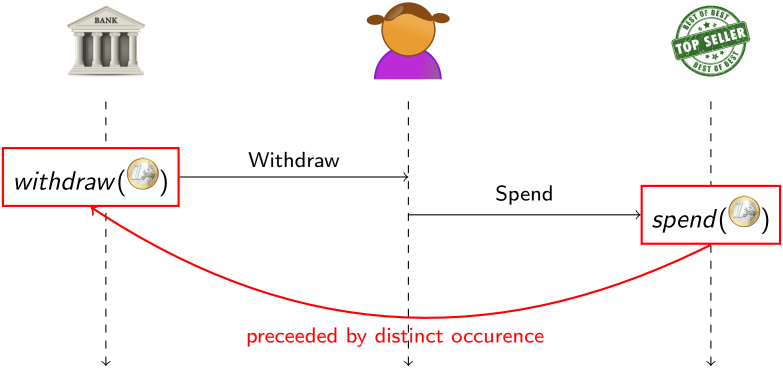
Conclusion

Unforgeability

Only the bank can create coins.

Definition:

On every trace:



Double Spending Identification

If a client spends a coin twice, his identity is revealed:

$\exists$ **Test** $\mathtt{T}_{\mathtt{DSI}}$ **such that:**

- $\forall$ transactions and using the same coin

  $\mathtt{T}_{\mathtt{DSI}}\Big($, , $\Big) = \Big($, $\Big)$

- $\mathtt{T}_{\mathtt{DSI}}(\cdot,\cdot,\cdot) = \perp$ otherwise

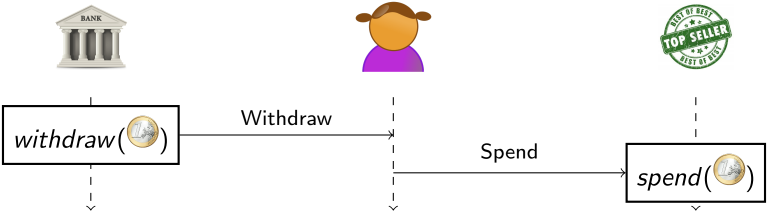
where

- is some data from the bank
- is evidence that withdrew

Exculpability

An attacker cannot forge a double spend by a client to blame him:

- **Attacker sees** , i.e.:



- **but cannot forge** such that:

$\mathrm{T}_{\mathtt{DSI}}$ (, ,) = (,)

Plan

Introduction

Model and Properties

- Forgery-Related Properties
- Anonymity Properties

Case Studies

- Chaum's On-Line Protocol
- DigiCash Protocol
- Chaum's Off-Line Protocol

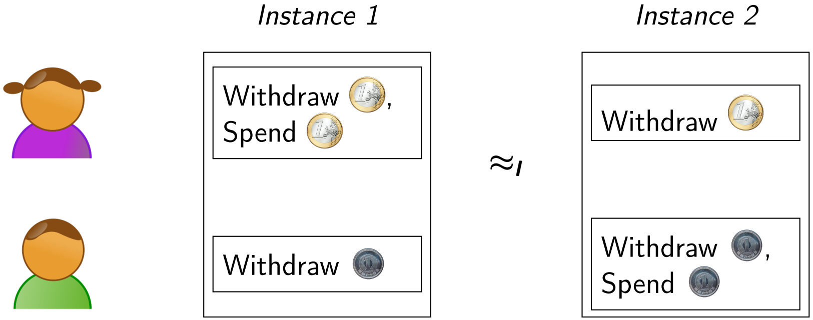
Conclusion

Weak Anonymity

Nobody can distinguish which client makes a payment.

Definition:

Observational equivalence of two instances:



Instance 1

Withdraw,
Spend

Withdraw

$\approx_I$

Instance 2

Withdraw

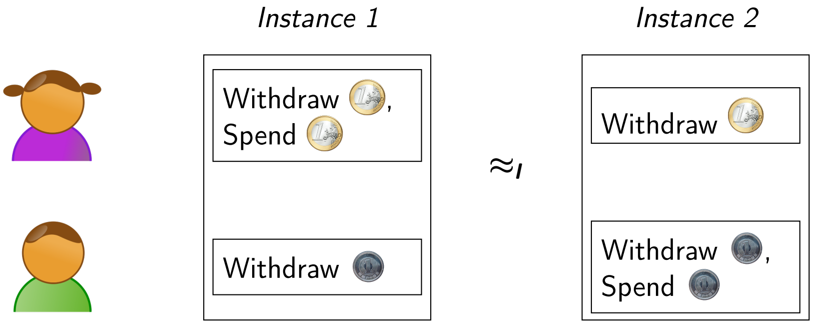
Withdraw,
Spend

Weak Anonymity

Nobody can distinguish which client makes a payment.

Definition:

Observational equivalence of two instances:



Instance 1

Withdraw, Spend

Withdraw

$\approx_l$

Instance 2

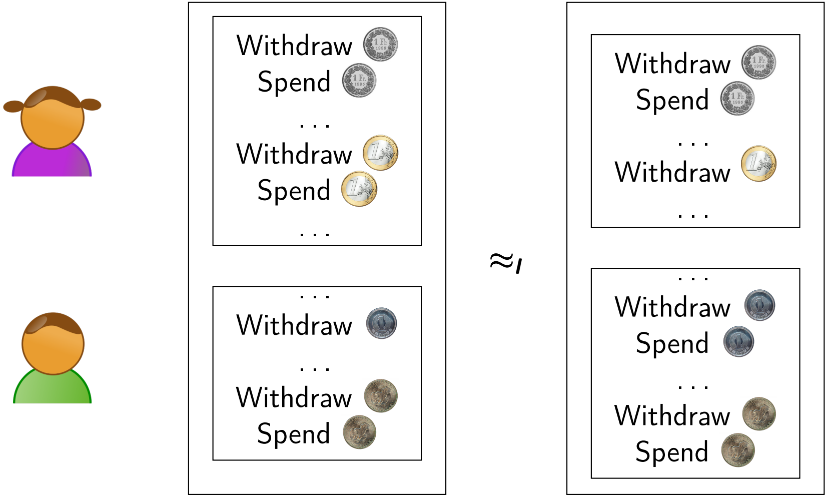
Withdraw

Withdraw, Spend

Note that the **bank** and the **seller** are **corrupted**.

Strong Anonymity

Nobody is able to decide whether two payments were made by the same client:



Withdraw

Spend

. . .

Withdraw

Spend

. . .

. . .

Withdraw

. . .

Withdraw

Spend

$\approx_I$

Withdraw

Spend

. . .

Withdraw

. . .

. . .

Withdraw

Spend

. . .

Withdraw

Spend

Plan

- Introduction
- Model and Properties
  - Forgery-Related Properties
  - Anonymity Properties
- Case Studies
  - Chaum's On-Line Protocol
  - DigiCash Protocol
  - Chaum's Off-Line Protocol
- Conclusion

Plan

Introduction

Model and Properties

- Forgery-Related Properties
- Anonymity Properties

Case Studies

- Chaum's On-Line Protocol
- DigiCash Protocol
- Chaum's Off-Line Protocol

Conclusion

Application: Chaum's On-Line Protocol

First on-line E-Cash protocol [?] using

- blind signatures
- on-line verification by the bank to prevent double spending

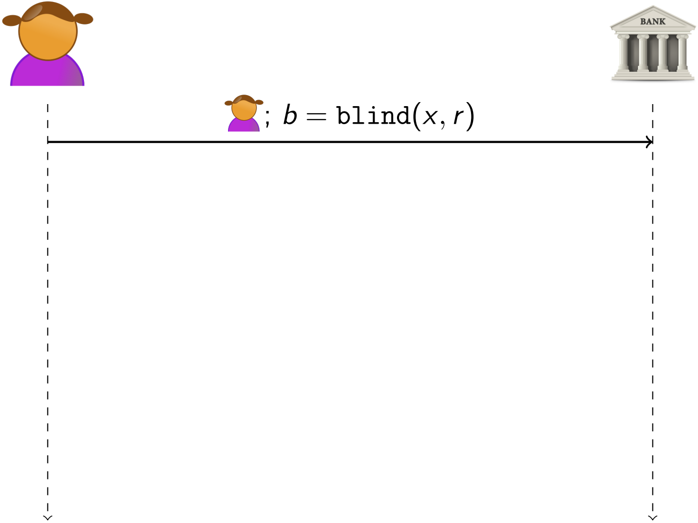
Goal: ensure

- unforgeability
- anonymity

in presence of **dishonest**

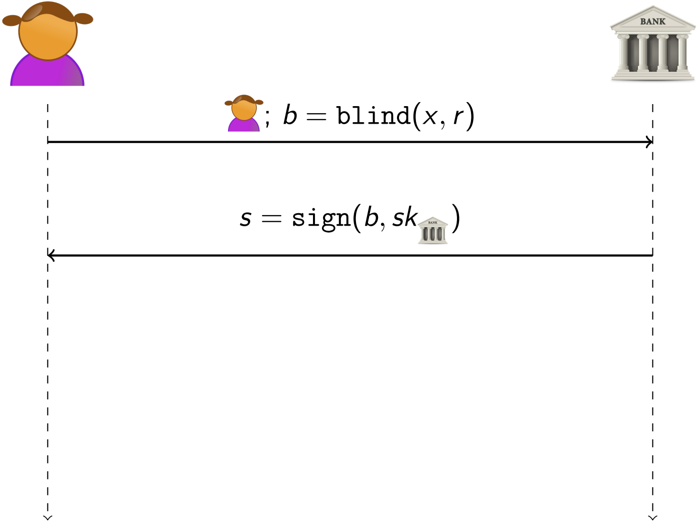
- banks
- sellers
- clients

Withdrawal Phase



; $b = \texttt{blind}(x, r)$

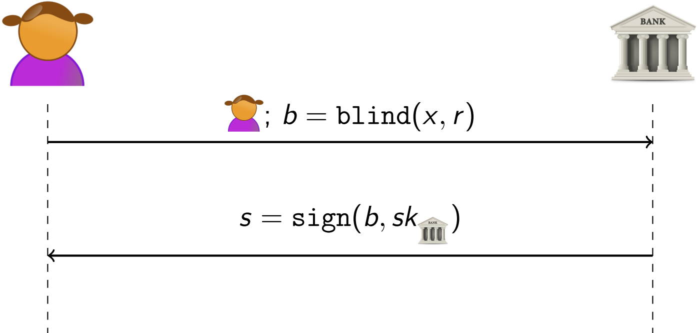
Withdrawal Phase



👧; $b = \mathtt{blind}(x, r)$

$s = \mathtt{sign}(b, sk_{🏛})$

Withdrawal Phase

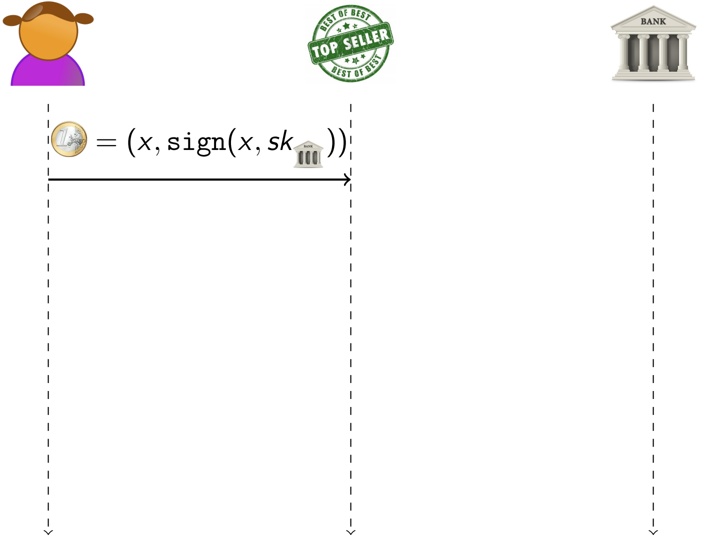


👧; $b = \texttt{blind}(x, r)$

$s = \texttt{sign}(b, sk_{🏛})$

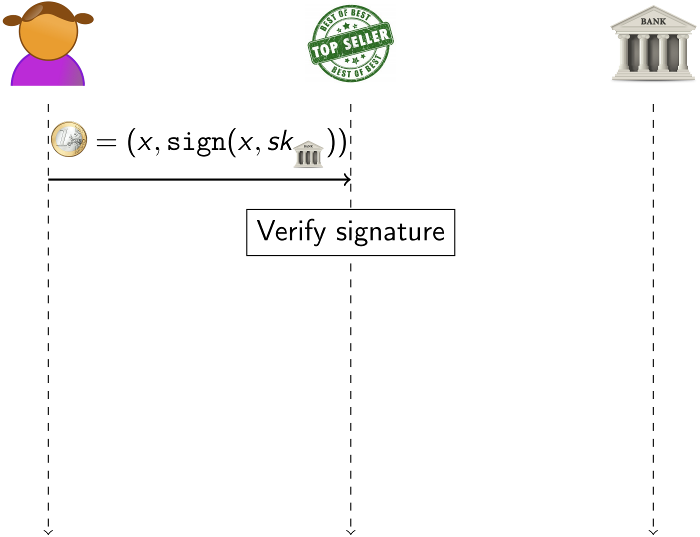
1. Verify signature s
2. Compute $y = \texttt{unblind}(s, r) = \texttt{sign}(x, sk_{🏛})$
3. Coin 🪙 $= (x, y) = (x, \texttt{sign}(x, sk_{🏛}))$

Payment and Deposit Phase



🪙 $= (x, \texttt{sign}(x, sk_{🏛}))$

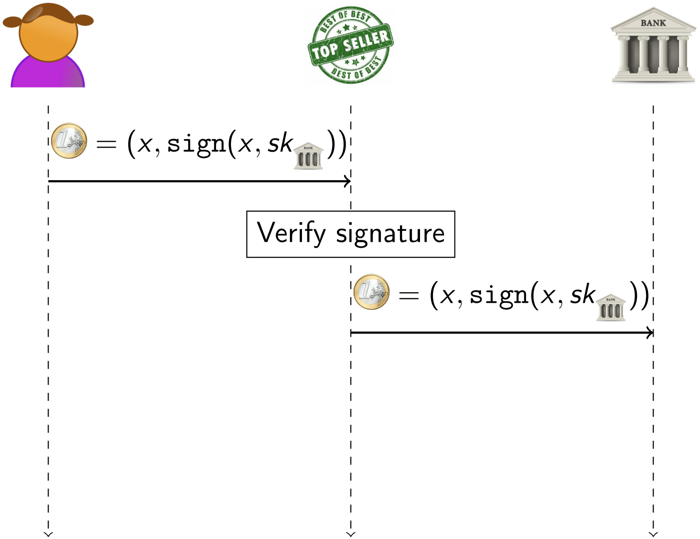
Payment and Deposit Phase



🪙 $= (x, \texttt{sign}(x, sk_{\text{🏛}}))$

Verify signature

Payment and Deposit Phase

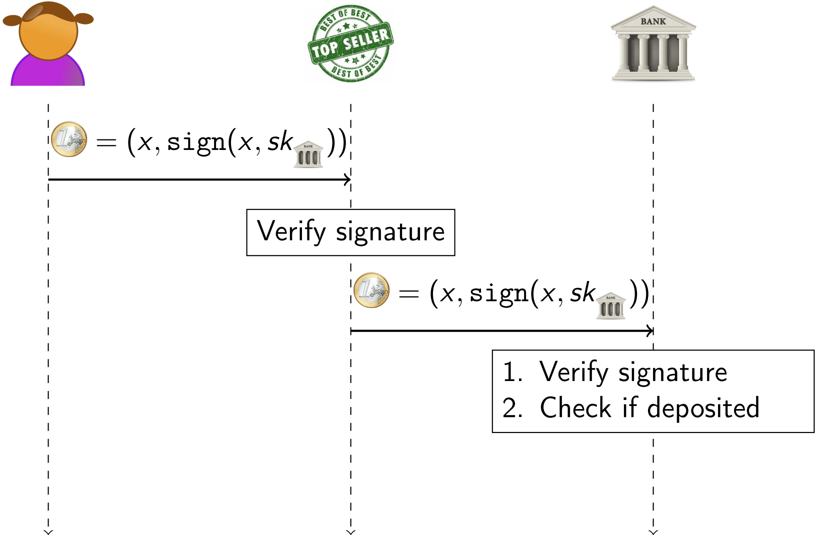


🪙 $= (x, \mathtt{sign}(x, sk_{\text{🏛}}))$

Verify signature

🪙 $= (x, \mathtt{sign}(x, sk_{\text{🏛}}))$

Payment and Deposit Phase



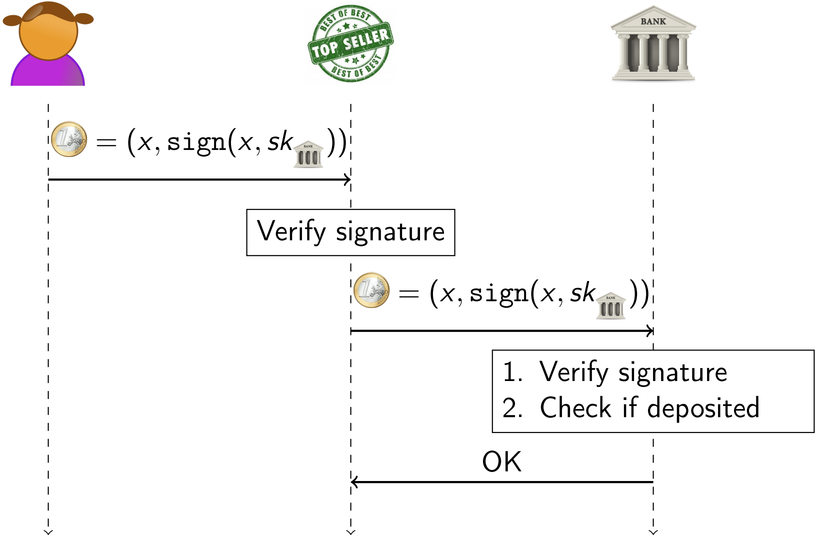
$= (x, \mathtt{sign}(x, sk))$

Verify signature

$= (x, \mathtt{sign}(x, sk))$

1. Verify signature
2. Check if deposited

Payment and Deposit Phase



1 EURO $= (x, \mathtt{sign}(x, sk_{\text{BANK}}))$

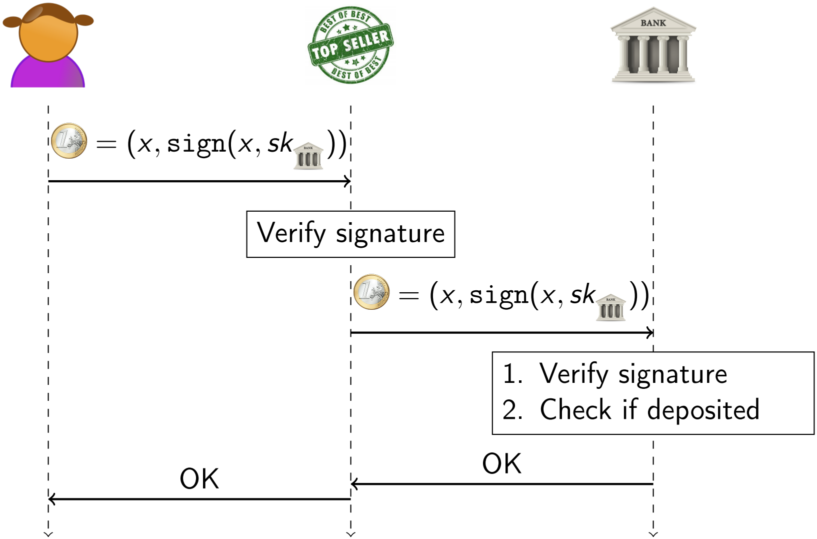
Verify signature

1 EURO $= (x, \mathtt{sign}(x, sk_{\text{BANK}}))$

1. Verify signature
2. Check if deposited

OK

Payment and Deposit Phase



🪙 $= (x, \mathtt{sign}(x, sk_{🏛}))$

Verify signature

🪙 $= (x, \mathtt{sign}(x, sk_{🏛}))$

1. Verify signature
2. Check if deposited

OK

OK

Results

Formal Verification with ProVerif [?]:

Property	**Result**	**Time**
Unforgeability	×	< 1 s
Double Spending Identification	–	–
Exculpability	–	–
Weak Anonymity	✓	< 1 s
Strong Anonymity	✓	< 1 s

- **Race condition** on the on-line verification, requires synchronization
- **Double Spending Identification** and **Exculpability** are irrelevant for on-line protocols.

Plan

Introduction

Model and Properties

- Forgery-Related Properties
- Anonymity Properties

Case Studies

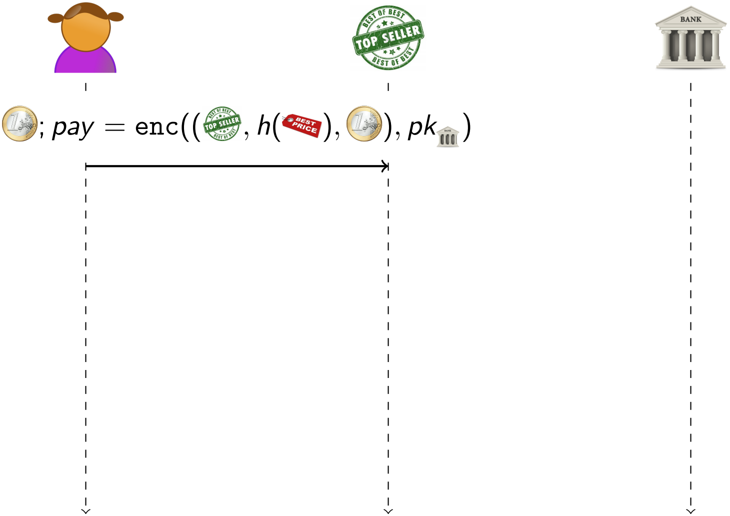
- Chaum's On-Line Protocol
- DigiCash Protocol
- Chaum's Off-Line Protocol

Conclusion

DigiCash Protocol

Variant of Chaum's On-Line protocol

- Different **payment and deposit phase**:

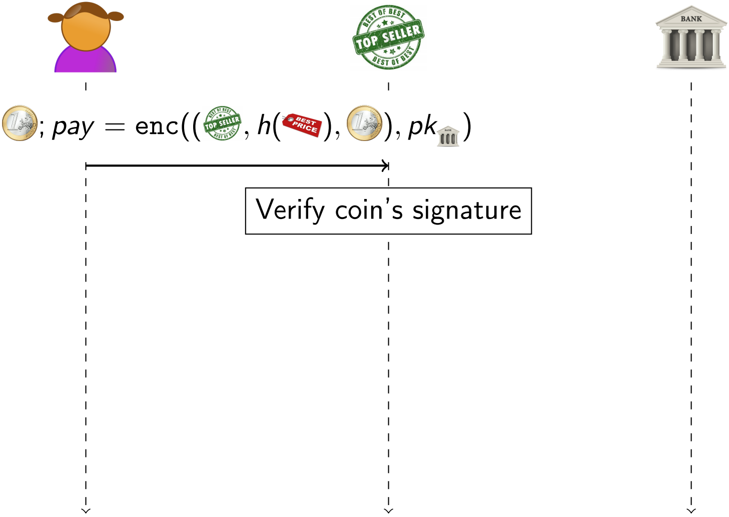


€; $pay = \mathrm{enc}((\text{TOP SELLER}, h(\text{BEST PRICE}), €), pk_{\text{BANK}})$

DigiCash Protocol

Variant of Chaum's On-Line protocol

- Different **payment and deposit phase**:



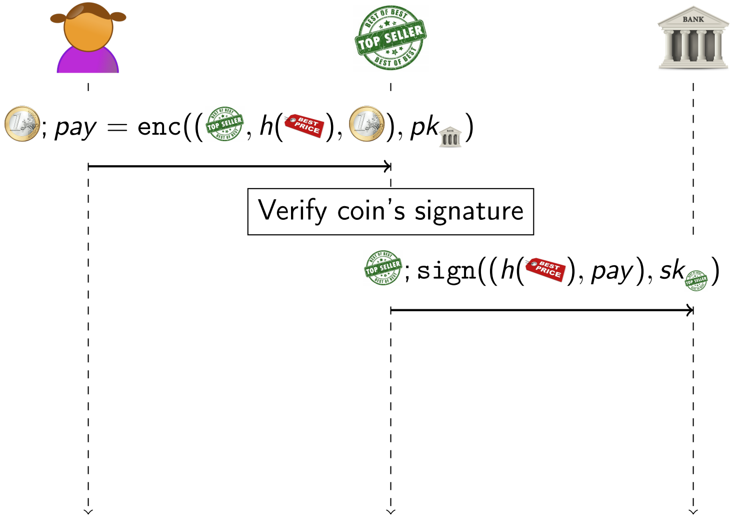
🪙; $pay = \mathtt{enc}((\text{TOP SELLER}, h(\text{BEST PRICE}), 🪙), pk_{\text{BANK}})$

Verify coin's signature

DigiCash Protocol

Variant of Chaum's On-Line protocol

- Different **payment and deposit phase**:



$\text{1 EURO}; \mathit{pay} = \mathtt{enc}((\text{TOP SELLER}, h(\text{BEST PRICE}), \text{1 EURO}), \mathit{pk}_{\text{BANK}})$

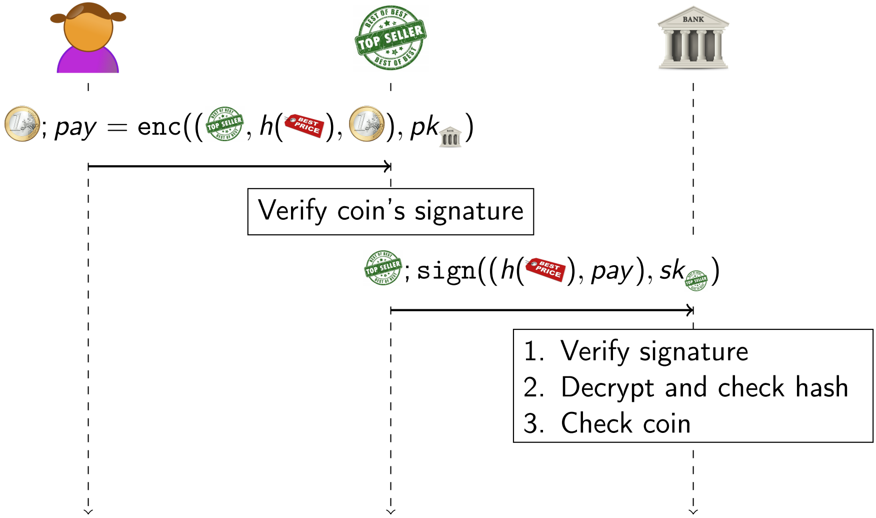
Verify coin's signature

$\text{TOP SELLER}; \mathtt{sign}((h(\text{BEST PRICE}), \mathit{pay}), \mathit{sk}_{\text{TOP SELLER}})$

DigiCash Protocol

Variant of Chaum's On-Line protocol

- Different **payment and deposit phase**:



🪙; $pay = \mathtt{enc}((🏅, h(🏷), 🪙), pk_{🏛})$

Verify coin's signature

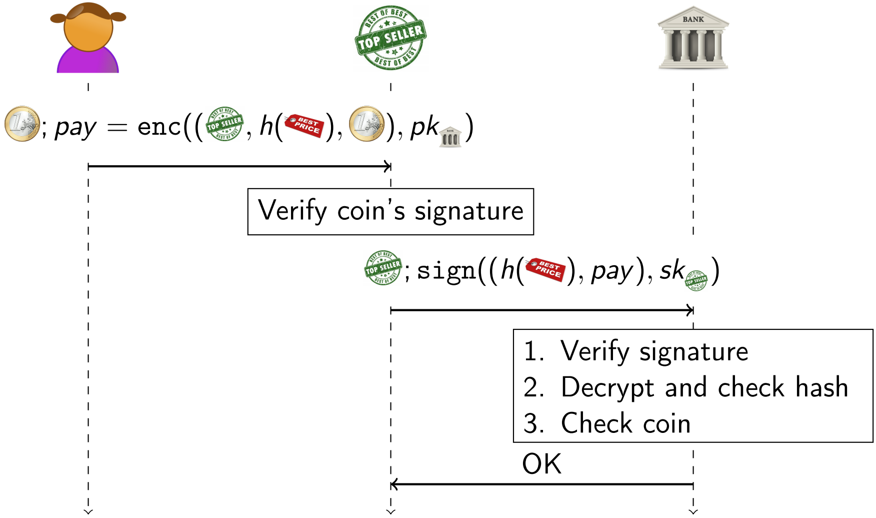
🏅; $\mathtt{sign}((h(🏷), pay), sk_{🏅})$

1. Verify signature
2. Decrypt and check hash
3. Check coin

DigiCash Protocol

Variant of Chaum's On-Line protocol

- Different **payment and deposit phase**:



1 EURO; $pay = \mathtt{enc}((\text{TOP SELLER}, h(\text{BEST PRICE}), \text{1 EURO}), pk_{\text{BANK}})$

Verify coin's signature

TOP SELLER; $\mathtt{sign}((h(\text{BEST PRICE}), pay), sk_{\text{TOP SELLER}})$

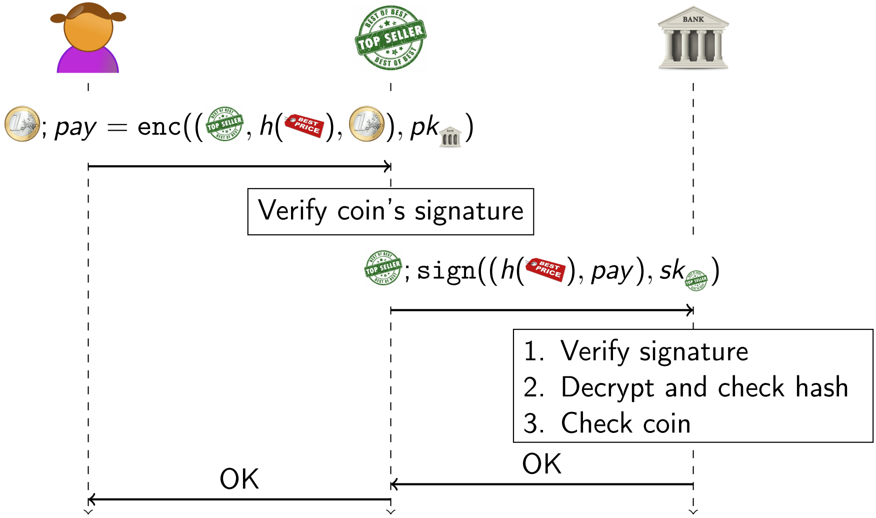
1. Verify signature
2. Decrypt and check hash
3. Check coin

OK

DigiCash Protocol

Variant of Chaum's On-Line protocol

- Different **payment and deposit phase**:



🪙; *pay* = enc((🏪, *h*(🏷), 🪙), *pk*🏦)

Verify coin's signature

🏪; sign((*h*(🏷), *pay*), *sk*🏪)

1. Verify signature
2. Decrypt and check hash
3. Check coin

OK

OK

Results

Formal Verification with ProVerif:

Property	**Result**	**Time**
Unforgeability	×	< 1 s
Double Spending Identification	–	–
Exculpability	–	–
Weak Anonymity	✓	< 1 s
Strong Anonymity	✓	< 1 s

Same observations:

- **Race condition** on the on-line verification, requires synchronization
- **Double Spending Identification** and **Exculpability** are irrelevant for on-line protocols.

Plan

Introduction

Model and Properties

- Forgery-Related Properties
- Anonymity Properties

Case Studies

- Chaum's On-Line Protocol
- DigiCash Protocol
- Chaum's Off-Line Protocol

Conclusion

Chaum's Off-Line Protocol

Off-line variant [?] of Chaum's on-line protocol [?] using

- blind signatures
- cryptographic hash
- XOR

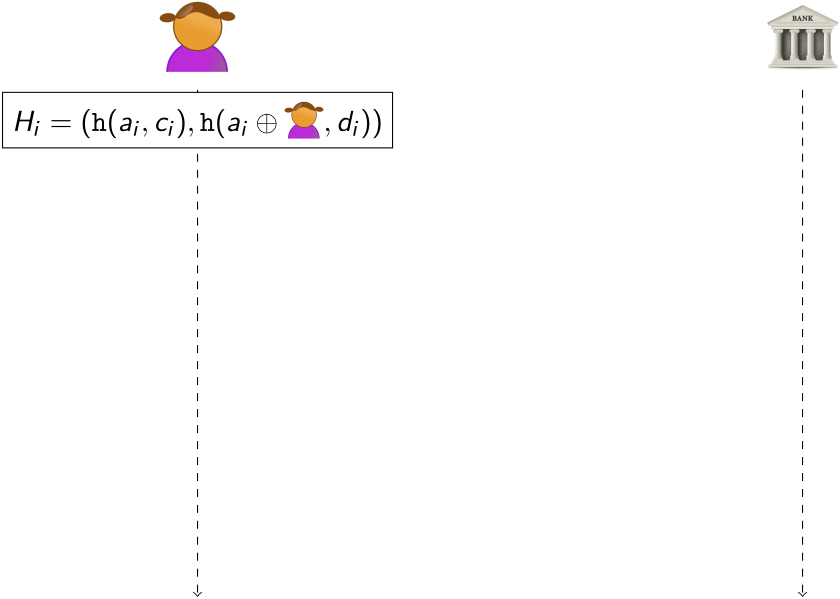
Goal: ensure

- unforgeability
- double spending identification
- exculpability
- anonymity

in presence of **dishonest**

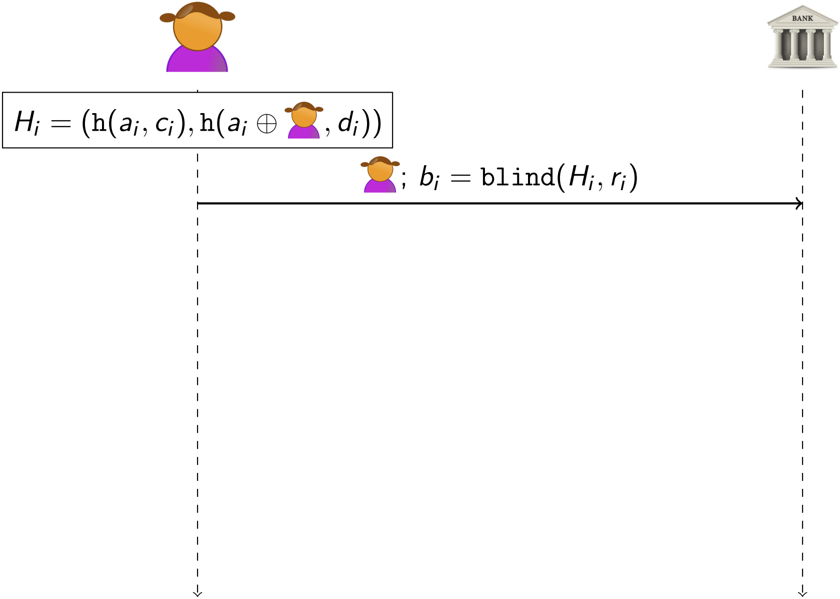
- banks
- sellers
- clients

Withdrawal Phase



$$\boxed{H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus \text{👧}, d_i))}$$

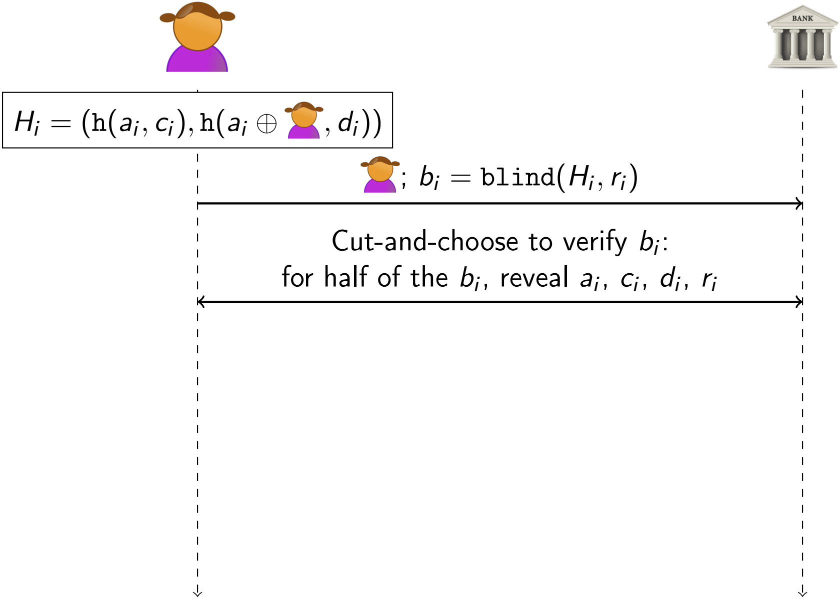
Withdrawal Phase



$H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus$ 👧$, d_i))$

👧; $b_i = \mathtt{blind}(H_i, r_i)$

Withdrawal Phase

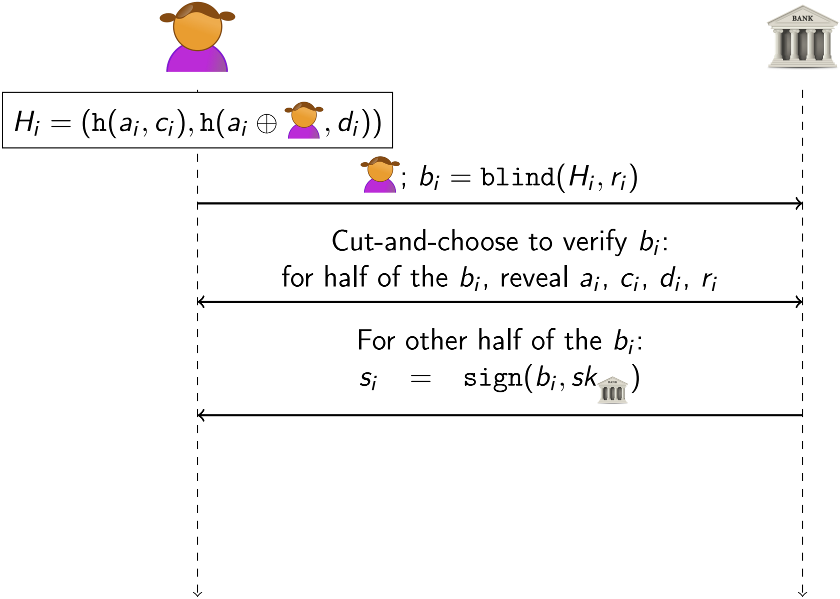


$H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus$ 👧 $, d_i))$

👧 $; \ b_i = \mathtt{blind}(H_i, r_i)$

Cut-and-choose to verify b_i:
for half of the b_i, reveal a_i, c_i, d_i, r_i

Withdrawal Phase



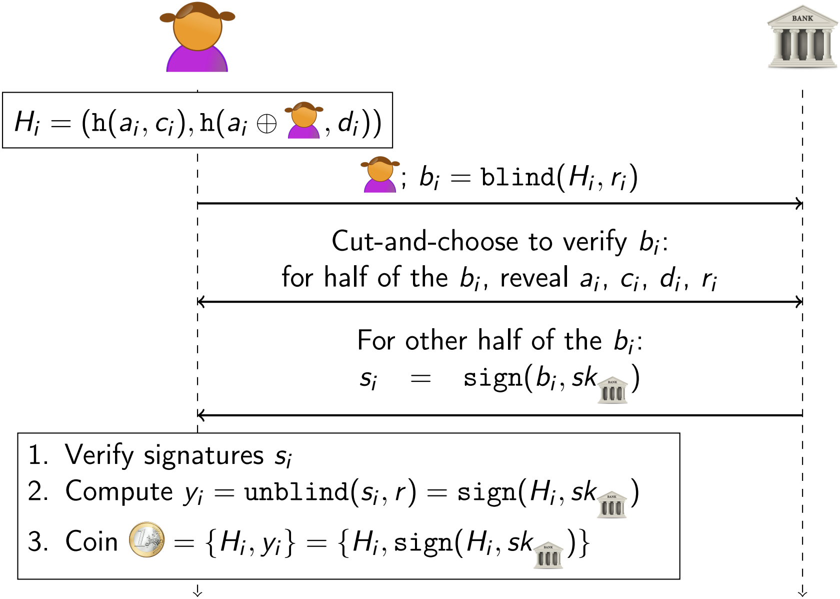
$H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus$ 👧 $, d_i))$

👧 ; $b_i = \mathtt{blind}(H_i, r_i)$

Cut-and-choose to verify b_i:
for half of the b_i, reveal a_i, c_i, d_i, r_i

For other half of the b_i:
$s_i \quad = \quad \mathtt{sign}(b_i, sk_{\text{BANK}})$

Withdrawal Phase



$H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus$ 👧 $, d_i))$

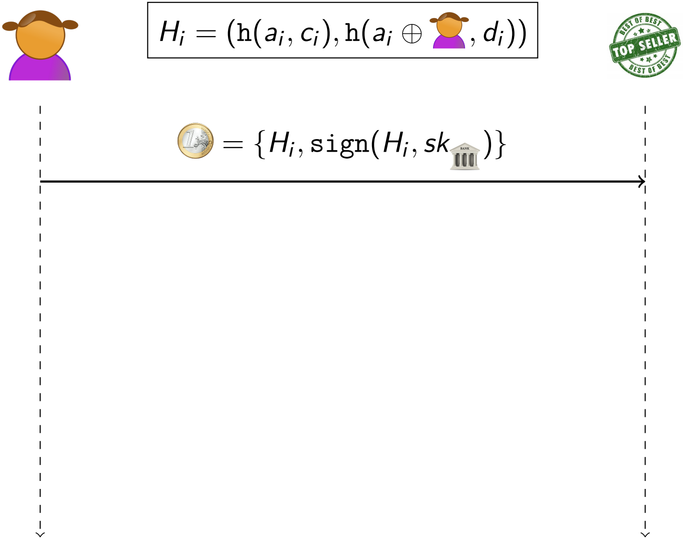
👧 ; $b_i = \mathtt{blind}(H_i, r_i)$

Cut-and-choose to verify b_i:
for half of the b_i, reveal a_i, c_i, d_i, r_i

For other half of the b_i:
$s_i = \mathtt{sign}(b_i, sk_{🏛})$

1. Verify signatures s_i
2. Compute $y_i = \mathtt{unblind}(s_i, r) = \mathtt{sign}(H_i, sk_{🏛})$
3. Coin 🪙 $= \{H_i, y_i\} = \{H_i, \mathtt{sign}(H_i, sk_{🏛})\}$

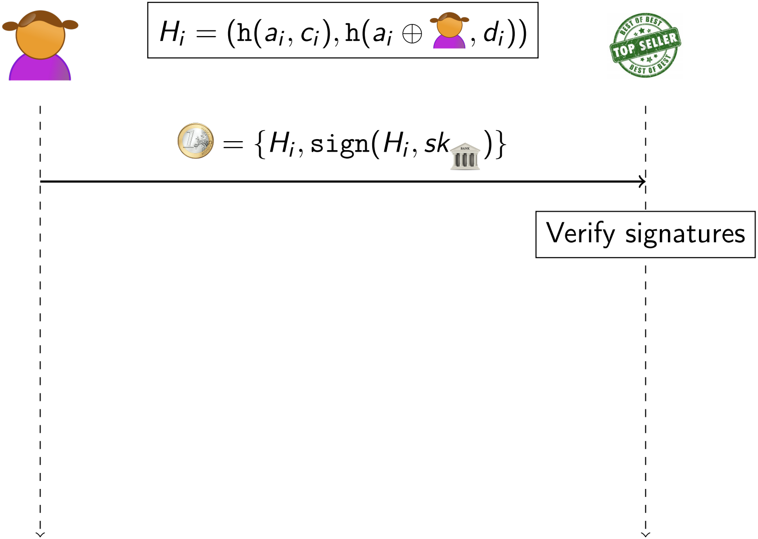
Payment Phase



$$\boxed{H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus \text{👤}, d_i))}$$

$$\text{🪙} = \{H_i, \mathtt{sign}(H_i, sk_{\text{🏛}})\}$$

Payment Phase

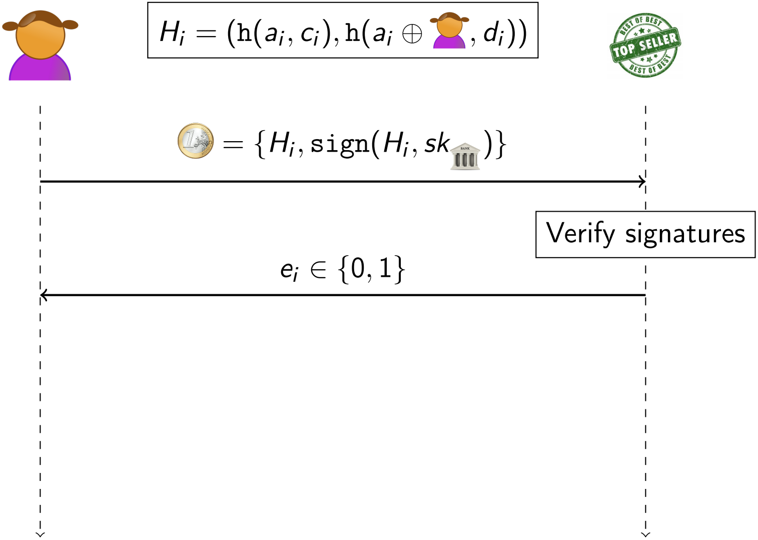


$$H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus \text{👧}, d_i))$$

$$\text{🪙} = \{H_i, \mathtt{sign}(H_i, sk_{\text{🏛}})\}$$

Verify signatures

Payment Phase



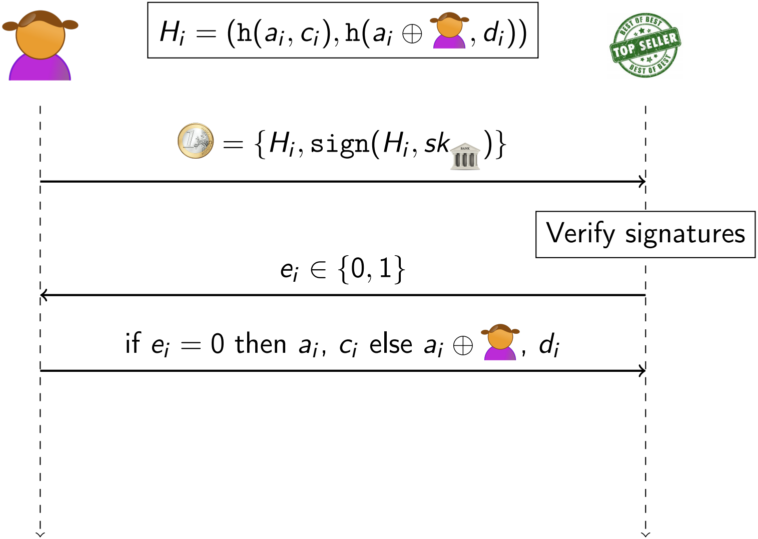
$H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus$ 👧$, d_i))$

🪙 $= \{H_i, \mathtt{sign}(H_i, sk_{\text{BANK}})\}$

Verify signatures

$e_i \in \{0, 1\}$

Payment Phase



$H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus \text{👧}, d_i))$

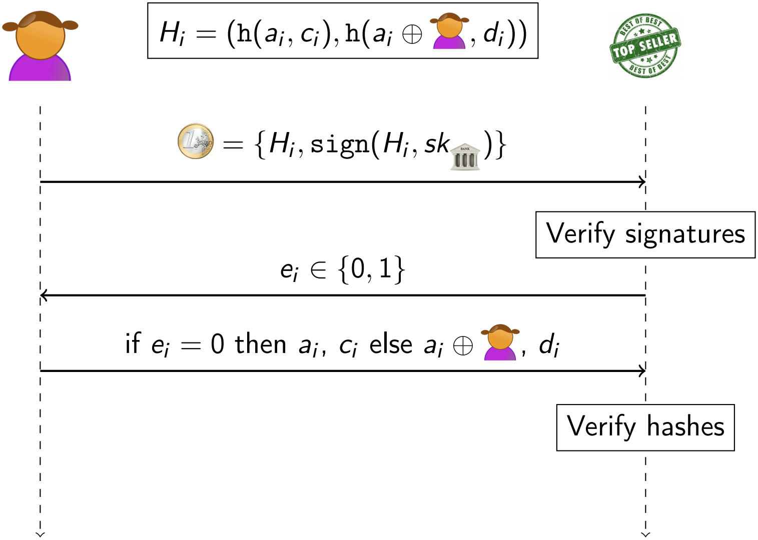
$\text{🪙} = \{H_i, \mathtt{sign}(H_i, sk_{\text{🏦}})\}$

Verify signatures

$e_i \in \{0, 1\}$

if $e_i = 0$ then a_i, c_i else $a_i \oplus \text{👧}$, d_i

Payment Phase



$H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus$ 👧 $, d_i))$

🪙 $= \{H_i, \mathtt{sign}(H_i, sk_{🏛}\,)\}$

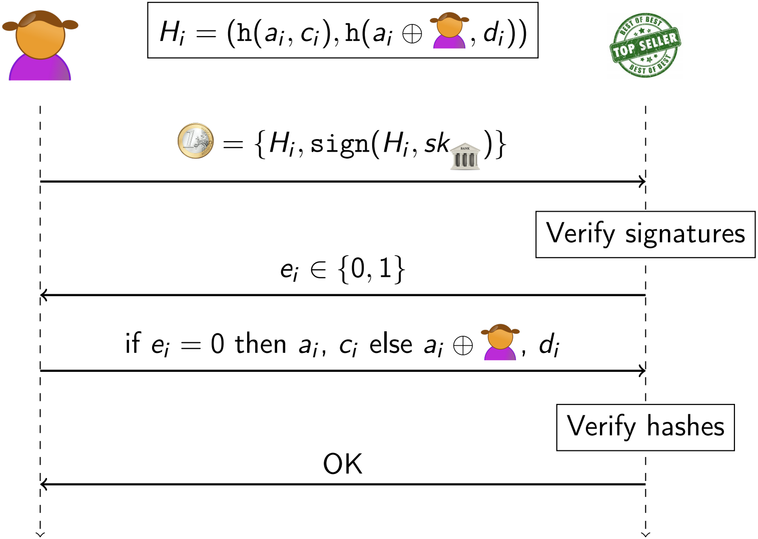
Verify signatures

$e_i \in \{0, 1\}$

if $e_i = 0$ then a_i, c_i else $a_i \oplus$ 👧 , d_i

Verify hashes

Payment Phase



$H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus$ 👧$, d_i))$

€ $= \{H_i, \mathtt{sign}(H_i, sk_{🏛}$$)\}$

Verify signatures

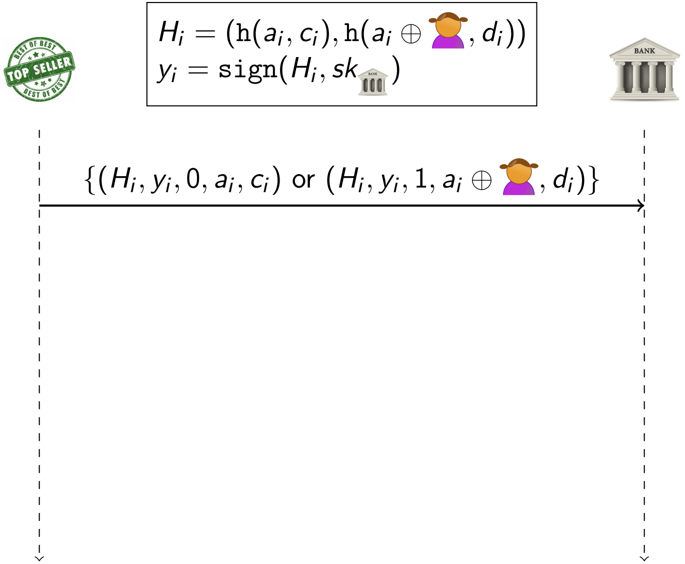
$e_i \in \{0, 1\}$

if $e_i = 0$ then a_i, c_i else $a_i \oplus$ 👧, d_i

Verify hashes

OK

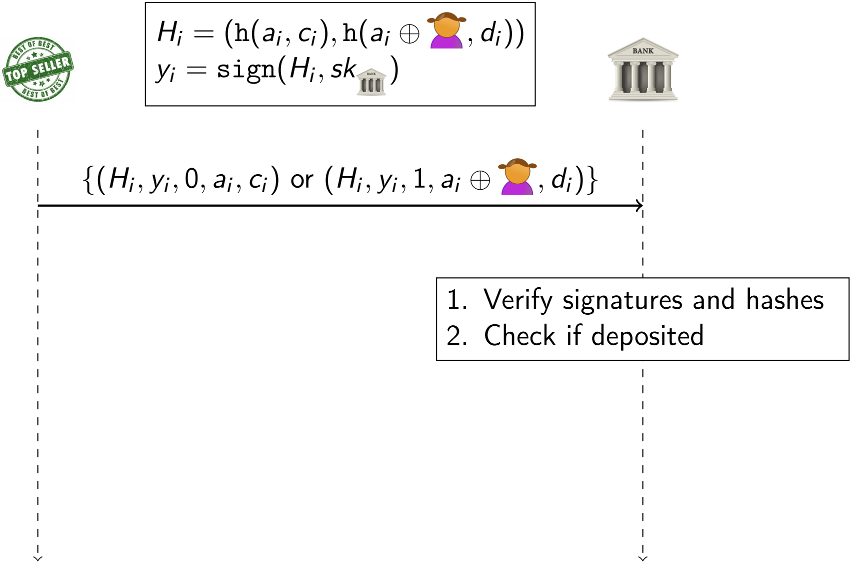
Deposit Phase



$$H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus 👧, d_i))$$
$$y_i = \mathtt{sign}(H_i, sk_{\text{BANK}})$$

$\{(H_i, y_i, 0, a_i, c_i)$ or $(H_i, y_i, 1, a_i \oplus 👧, d_i)\}$

Deposit Phase



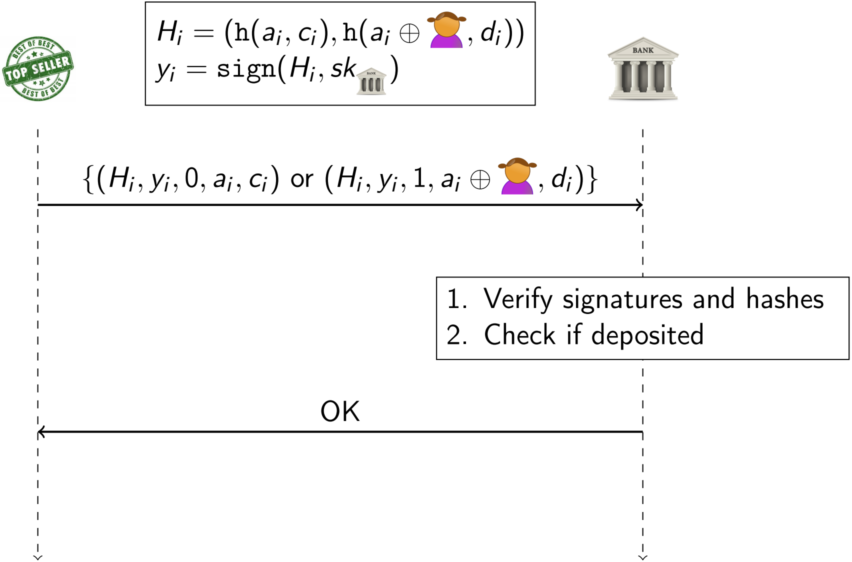
$$H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus \text{👤}, d_i))$$

$$y_i = \mathtt{sign}(H_i, sk_{\text{🏛}})$$

$\{(H_i, y_i, 0, a_i, c_i)$ or $(H_i, y_i, 1, a_i \oplus \text{👤}, d_i)\}$

1. Verify signatures and hashes
2. Check if deposited

Deposit Phase



$$H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus \text{👤}, d_i))$$
$$y_i = \mathtt{sign}(H_i, sk_{\text{🏛}})$$

$\{(H_i, y_i, 0, a_i, c_i)$ or $(H_i, y_i, 1, a_i \oplus \text{👤}, d_i)\}$

1. Verify signatures and hashes
2. Check if deposited

OK

Double Spending Identification

If BANK receives **two transactions** with the **same coin**, with high probability it has for at least one i

- a_i
- $a_i \oplus$ 👧
- c_i
- d_i

Allows to compute: $a_i \oplus (a_i \oplus$ 👧$) =$ 👧

Double Spending Identification

If BANK receives **two transactions** with the **same coin**, with high probability it has for at least one i

- a_i
- $a_i \oplus$ 👧
- c_i
- d_i

Allows to compute: $a_i \oplus (a_i \oplus \text{👧}) = \text{👧}$

- **However:** BANK can **forge double-spending** as it knows 👧
- **Fix:** $H_i = (\mathtt{h}(a_i, c_i), \mathtt{h}(a_i \oplus (\text{👧}, z_i, z_i'), d_i))$ and client deposits signature on $\mathtt{h}(z_i, z_i')$ at withdrawal

Results

Formal Verification with ProVerif:

Property	**Result**	**Time**
Unforgeability	×	< 1 s
Double Spending Identification	×	< 2 s
Double Spending Identification*	✓	< 2 s
Exculpability	×	< 6 s
Exculpability†	✓	< 6 s
Weak Anonymity	✓	< 1 s
Strong Anonymity	✓	< 1 s

Observations:

- **Double spending** possible, violating unforgeability
- Double Spending Identification requires **cut-and-choose** (*)
- **Exculpability** needs **fix** (†).

Plan

Introduction

Model and Properties

- Forgery-Related Properties
- Anonymity Properties

Case Studies

- Chaum's On-Line Protocol
- DigiCash Protocol
- Chaum's Off-Line Protocol

Conclusion

Conclusion

- **E-cash** can offer anonymous payment
- **Formal framework** for analysis of e-cash protocols:
  - Formal model in the **applied** π**-calculus**
  - **Definitions** for central forgery-related and anonymity properties
- **Automated verification in ProVerif** of three case studies:
  - Chaum's On-Line Protocol: race condition on online verification
  - DigiCash Protocol: same race condition on online verification
  - Chaum's Off-Line Protocol: requires cut-and-choose and fix
- **Future work**: verification with synchronization and XOR, dividable and transferable coins

Thank you for your attention!

Questions?

jannik.dreier@inf.ethz.ch