Foundations of Cryptography.

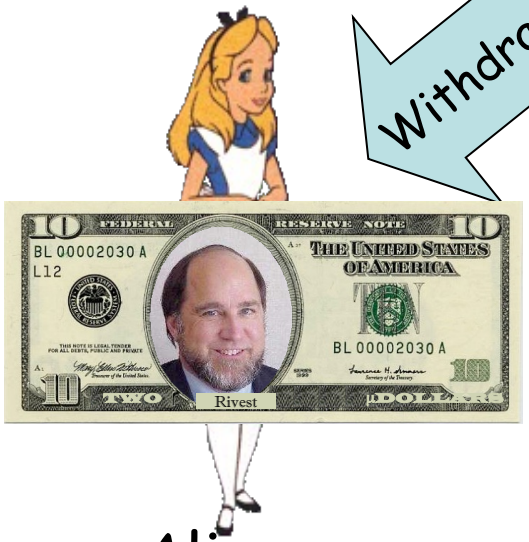
Lecture 3: Privacy-Preserving Digital Money

Anna Lysyanskaya

The Money Cycle

Withdraw $$$

Deposit $$$

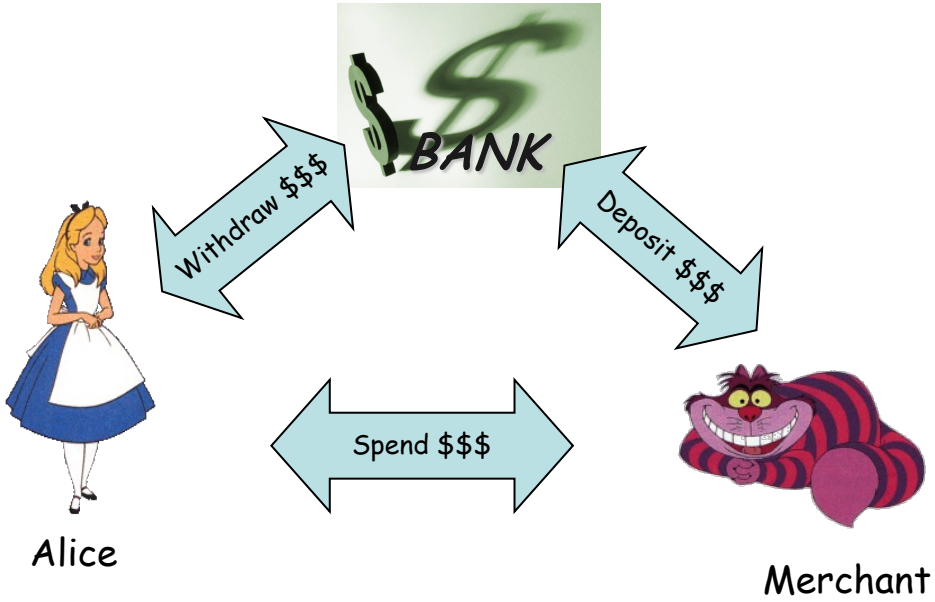


Alice

Spend $$$

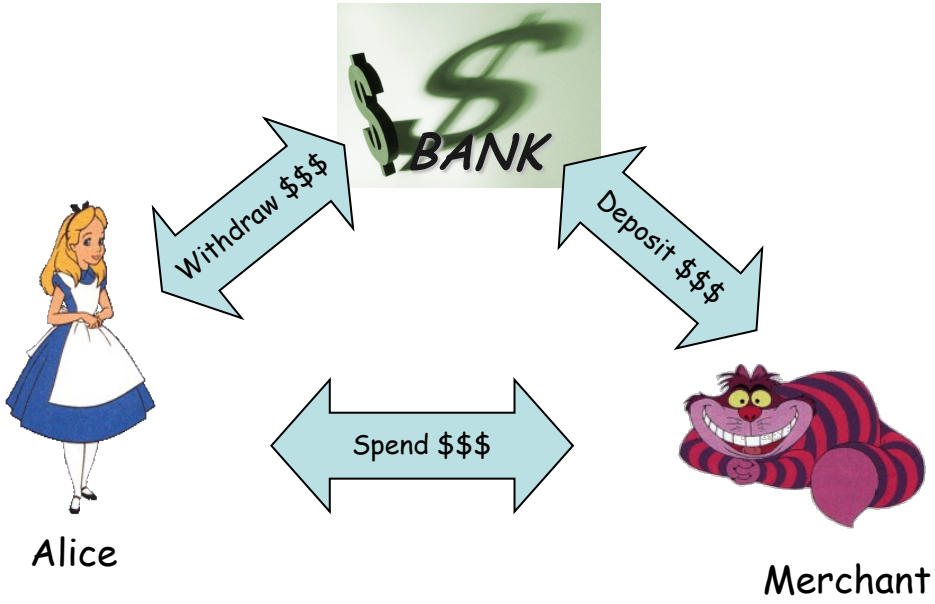
Merchant

The Money Cycle



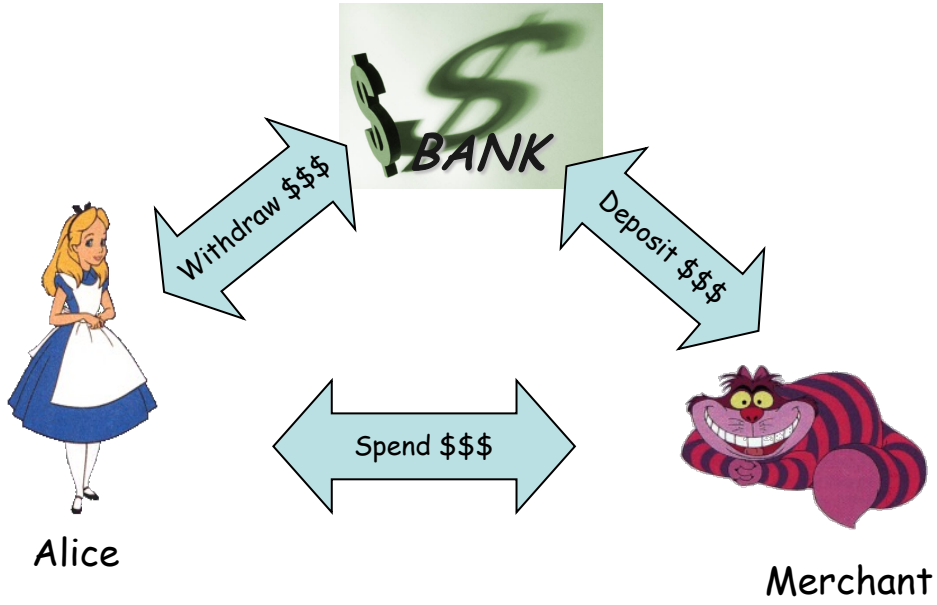
- Three protocols: Withdraw, Spend, Deposit
- Desirable properties:
  - can't forge/copy money
  - can't trace how cash was spent

Electronic Payments



- Three protocols: Withdraw, Spend, Deposit
- Desirable properties:
  - can't forge/copy money
  - can't trace how cash was spent

Ecash [Chaum82,CFN89]



- Unforgeability: Alice can't spend more $$ than she withdrew
  - Online ecash: each coin has a serial number, Merchant can't deposit unless it's unspent
  - Offline ecash: if Alice double-spent, can ID and punish her after the fact
- Privacy: colluding B&M can't trace how a coin is spent.

Roadmap for This Talk

- Main idea of off-line ecash [CFN89 + CL02] and compact ecash [CHL05] ✔
- Balancing anonymity and accountability:
  - How to prevent money laundering [CHL06]
  - How to trace rogue users' transactions
  - How to implement authorized watchlists [KLN23]

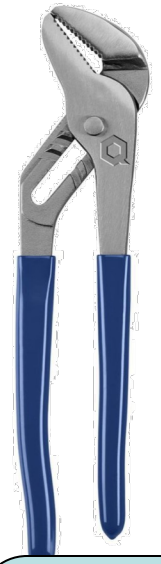
Warning: there might be a pop quiz...

Main Idea of Off-Line Ecash

- Building blocks:
  - digital signatures
  - secure two-party computation
  - NIZK proofs of knowledge
  - pseudorandom functions

Main Idea of Off-Line Ecash

- SETUP: the Bank sets up his key pair for a digital signature scheme
  - Signing key sk
  - Verification key pk

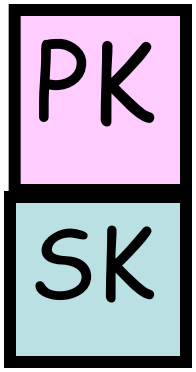


Tool #1: Digital signature scheme [RSA77]

Signature Schemes

Signature Schemes

- Setup: I run a setup algorithm to obtain my public key PK and secret key SK

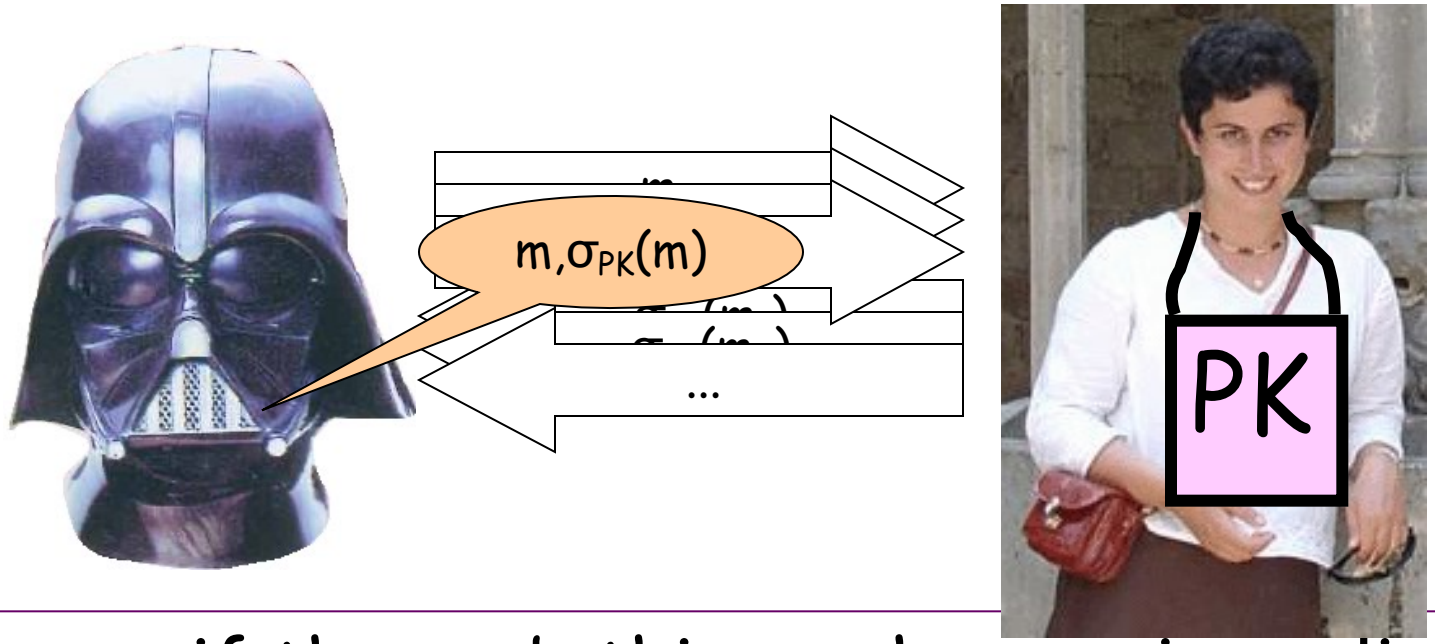


Signature Schemes

- Setup: I run a setup algorithm to obtain my public key PK and secret key SK
- Now I can sign (using SK):
  - Sign(SK,m) $\rightarrow \sigma$ (denoted $\sigma_{PK}(m)$)
- And you can verify it (using PK)
  - Verify(PK,m,σ) $\rightarrow$ Yes/No

Signature Schemes

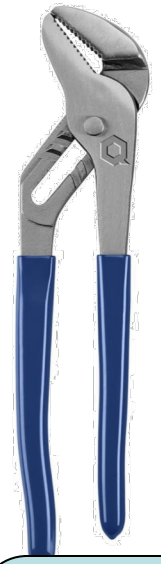
- Security: no adversary can forge a signature even after seeing sigs on messages of his choice



Secure if the prob this can happen is negligible

Main Idea of Off-Line Ecash

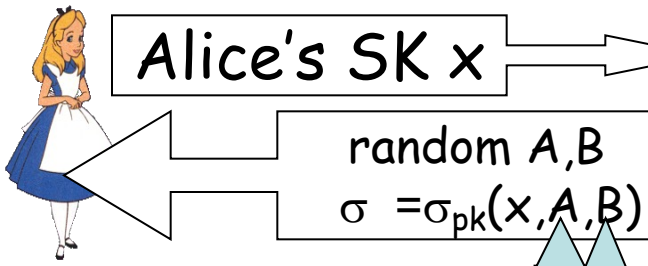
- SETUP: the Bank sets up his key pair for a digital signature scheme
  - Signing key sk
  - Verification key pk



Tool #1: Digital signature scheme [RSA77]

Main Idea of Off-Line Ecash

- WITHDRAW a coin that will verify under the Bank's verification key pk:



Alice's SK x

random A,B

$\sigma = \sigma_{pk}(x,A,B)$

2PC

sk

A is the serial number

B is to prevent double-spending (TBA)

Tool #2: Secure 2PC [Yao82]

Protocol where:

- one party receives an output
- neither party learns anything(*) about the other's input.

Main Idea of Off-Line Ecash

- SPEND:

"fresh" nonce R
e.g. R=H(contract, rand)

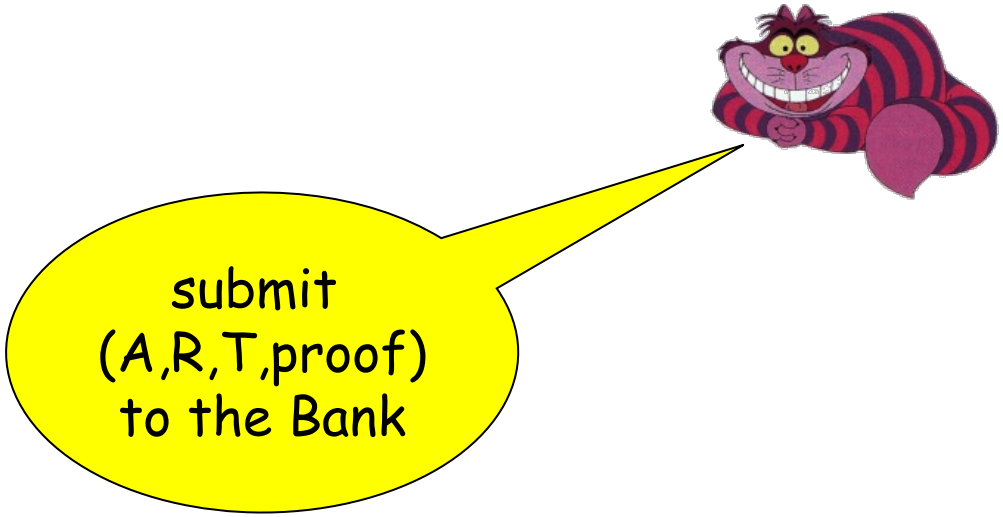
A (the coin's serial number)
T =x+RB mod Q (double-spending equation)

NIZKPOK of (x,B,σ) such that
1. T = x+RB mod Q
2. VerifySig(pk,(x,A,B), σ) = TRUE

Tool #3: NIZK proof of knowledge [GMR84…FLS91…] We saw it in Lecture 2

Main Idea of Off-Line Ecash

- DEPOSIT:



Can't Forge Money/Double-Spend

Identify algorithm:
Suppose a coin is spent twice.
Same coin => same A
Spent twice: two R's,
with high prob, R ≠ R'
T = x+RB mod Q, T' = x+R'Bmod Q
solve for x, id and punish Alice

R

A (the coin's serial number)
T =x+RB mod Q (double-spending equation)

NIZKPOK of (x,B,σ) such that
1. T = x+RB
2. VerifySig(pk,(x,A,B), σ) = TRUE

Deposit: submit (A,R,T,proof) to the Bank

User Privacy

A and T are random;
proof is zero-knowledge.

R

A (the coin's serial number)
T =x+RB mod Q (double-spending equation)

NIZKPOK of (x,B,σ) such that
1. T = x+RB
2. VerifySig(pk,(x,A,B), σ) = TRUE

Deposit: submit (A,R,T,proof) to the Bank

Real-Life Money (again)

Withdraw $2

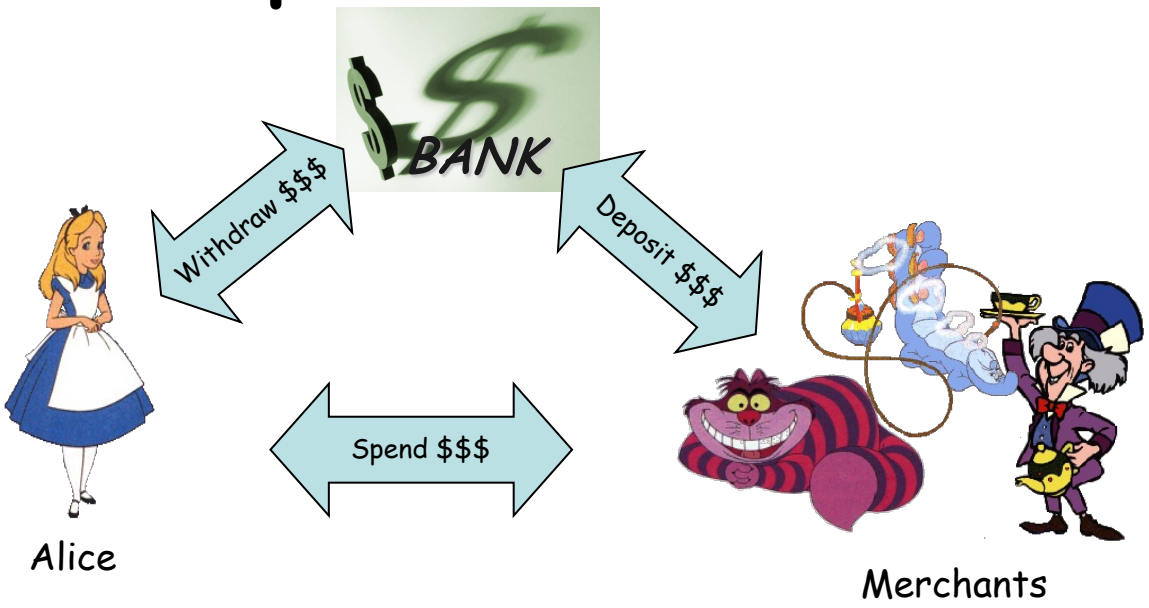
Deposit $2

Alice

Spend $2

Merchant

Compact Ecash



- Algs: Setup, Withdraw, Spend, Deposit, Identify
- Withdraw: a wallet with N coins
- Spend, deposit: just one coin
- Want: complexity of protocols O(log N), not O(N)

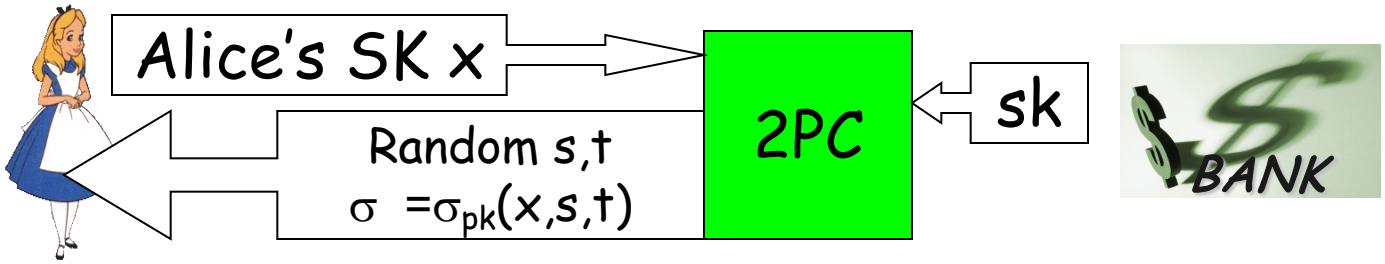
Tools for Compact Ecash

- Building blocks:
  - digital signatures
  - secure two-party computation
  - NIZK proofs of knowledge
  - pseudorandom functions

Tool #4: Pseudorandom function [GGM]: we saw it in Lecture 1

Compact Ecash: Main Idea [CHL05]

- WITHDRAW $N:



- SPEND $1 for the i^{th} time: Let $F_{(\ )}(\ )$ be a pseudorandom function family

R

A = $F_s(i)$ (the coin's serial number)
T = $x+RF_t(i) \bmod Q$ (double-spending equation)

NIZKPOK of (i,x,s,t,σ) such that
1. $1 \le i \le N$
2. $A = F_s(i)$
3. $T = x+RF_t(i)$
4. VerifySig(pk,(x,s,t), σ) = TRUE

Deposit: submit (A,R,T,proof) to the Bank

Compact Ecash: Main Idea [CHL05]

- WITHDRAW $

Sup

Privacy for Alice: the ZK
A and T are pseudorandom,
proof is zero-knowledhe

INK

-

$A = F_s(i)$ (the coin's serial number)
$T = x+RF_t(i) \bmod Q$ (double-spending equation)

NIZKPOK of (i,x,s,t,σ) such that
1. $1 \le i \le N$
2. $A = F_s(i)$
3. $T = x+RF_t(i)$
4. VerifySig(pk,(x,s,t), σ) = TRUE

Deposit: submit
(A,R,T,proof)
to the Bank

Coming up soon: a POP QUIZ!

Roadmap for This Talk

- Main idea of off-line ecash [CFN89 + CL02] and compact ecash [CHL05] ✔

- Balancing anonymity and accountability:
  - How to prevent money laundering [CHL06]
  - How to trace rogue users' transactions
  - How to implement authorized watchlists [KLN23]

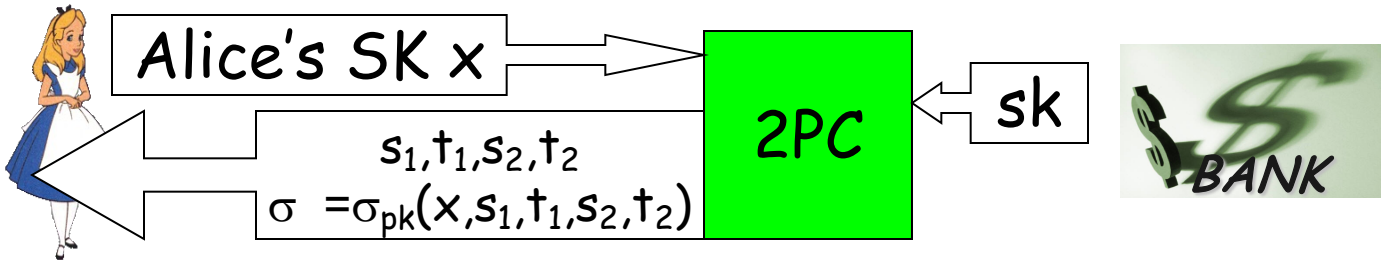
POP QUIZ:

Each user is allowed to spend only up to 100 coins with the Cheshire Cat. Modify the Compact Ecash construction so that the 101st spend with the Chesire Cat leads the Bank to identify the user

Hint: a coin can have multiple serial numbers

Preventing Money Laundering [CHL06]

- WITHDRAW $N:



- SPEND the i^{th} coin; this is the j^{th} time with this Merchant

R

$A_1 = F_{s1}(i)$, $A_2 = F_{s2}(CheshCat,j)$
$T_1 = x+RF_{t1}(i)$, $T_2 = x+RF_{t2}(CheshCat,j)$
NIZKPOK of $(i,x,s_1,t_1,j,s_2,t_2,\sigma)$ such that
1. $1 \le i \le N$, $1 \le j \le 100$
2. $A_1 = F_s(i)$, $A_2 = F_{s2}(CheshCat,j)$
3. $T_1 = x+RF_t(i)$, $T_2 = x+RF_{t2}(CheshCat,j)$
4. VerifySig(pk,(x,s_1,t_1,s_2,t_2), σ) = TRUE

Deposit: submit $(A_1,A_2,R,T_1,T_2,proof)$ to the Bank

- Cannot be done with physical cash! Was an open problem too, for a while.

Preventing Money Laundering [CHL06]

- WITHDRAW $
- SP

Privacy for Alice: the ZK pick random A_1, T_1, A_2, T_2 are pseudorandom, proof is zero-knowledge

$A_1 = F_{s1}(i)$, $A_2 = F_{s2}(CheshCat,j)$
$T_1 = x+RF_{t1}(i)$, $T_2 = x+RF_{t2}(CheshCat,j)$
NIZKPOK of $(i,x,s_1,t_1,j,s_2,t_2,\sigma)$ such that
1. $1 \le i \le N$, $1 \le j \le 100$
2. $A_1 = F_s(i)$, $A_2 = F_{s2}(CheshCat,j)$
3. $T_1 = x+RF_t(i)$, $T_2 = x+RF_{t2}(CheshCat,j)$
4. VerifySig(pk,(x,s_1,t_1,s_2,t_2), σ) = TRUE

Deposit: submit $(A_1,A_2,R,T_1,T_2,proof)$ to the Bank

- Cannot be done with physical cash! Was an open problem too, for a while.

POP QUIZ 2:

If you double-spend < 4 e-tokens, these e-tokens are linked, but your identity cannot be established. If you double-spend 4 times, you are identified.

Hint: use multiple $R_1, \ldots, R_L$

Glitc … M061

Suppose spend N+4 coins
=> repeating $A=F_s(i)$ for some i
(possibly for i_1, i_2, i_3, i_4)
=> L pops out of repeating A
using T, T', R, R'
=> link them together!
=> $F_u(i)$ pops out of repeating A
using Y, Y', R, R'
=> each overspending gives
$x + r_1z_1 + r_2z_2 + r_3z_3 = Z - F_u(i)$

R, r_1, r_2, …

$A = F_s(i)$
$T = L + RF_t(i)$
$Y = F_u(i) + RF_v(i)$
$Z = x + r_1z_1 + r_2z_2 + r_3z_3 + F_u(i)$

NIZKPOK of $(i,x,s,t,u,v,L,z_1,z_2,z_3,\sigma)$ such that
1. $1 \le i \le N$
2. $A = F_s(i)$, $T = L + RF_t(i)$, $Y = F_u(i) + RF_v(i)$
3. $Z = x + r_1z_1 + r_2z_2 + r_3z_3 + F_u(i)$
4. VerifySig(pk,(x,s,t,u,v,L,z_1,z_2,z_3), σ)

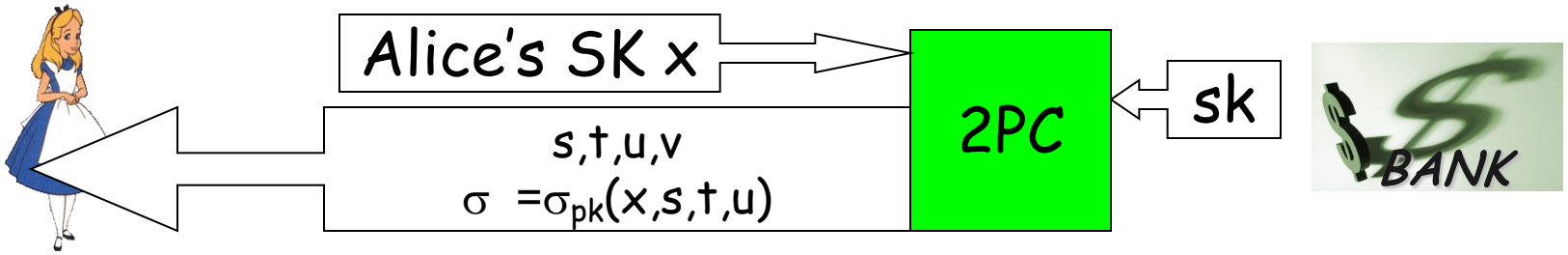
POP QUIZ 3:

Construct an ecash scheme where double-spending leads not just to identification, but also to traceability of past transactions from the same wallet.

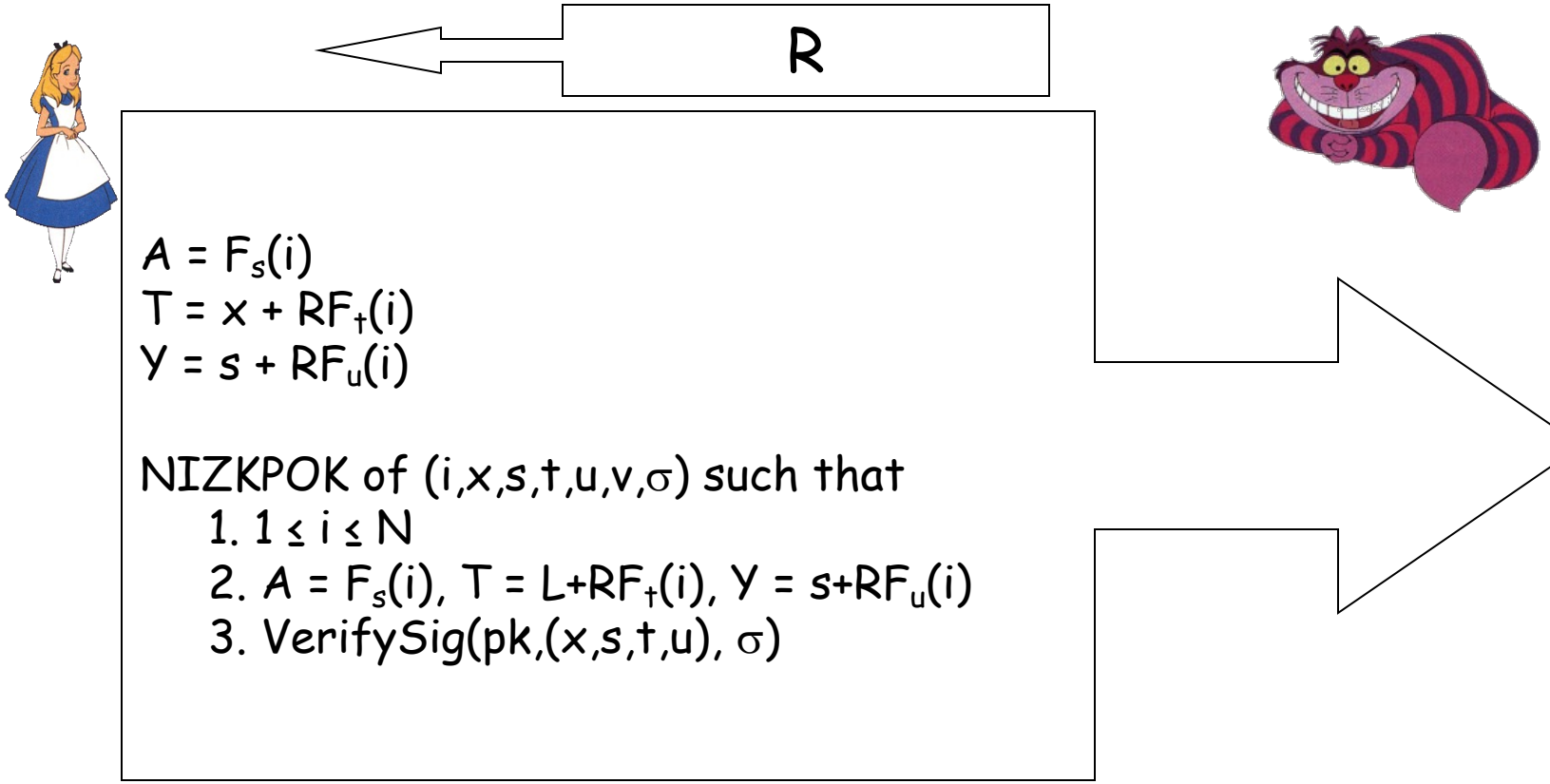
Hint: double-spending makes s recoverable

Traceability [CHKLM06]

- WITHDRAW:



- SPEND $1 for the i^{th} time:



Roadmap for This Talk

- Main idea of off-line ecash [CFN89 + CL02] and compact ecash [CHL05] ✔

- Balancing anonymity and accountability:
  - How to prevent money laundering [CHL06] ✔
  - How to trace rogue users' transactions ✔
  - How to implement authorized watchlists [KLN23]

Watchlists [KLN23]

- WITHDRAW:

Alice's SK x

seeds

$\sigma = \sigma_{pk}(x, seeds, Alice)$

2PC

sk

- SPEND:

Encrypted watchlist

E-coin's info (as before)

$$\text{Escrow} = \begin{cases} \text{Enc(name) if name is on the watchlist} \\ \text{Enc(random) otherwise} \end{cases}$$

NOTE: using homomorphic encryption can compute Escrow without knowing the watchlist

NIZKPOK of $(x, seeds, name, \sigma)$ such that

1. E-coin was computed correctly from seeds
2. Escrow was computed correctly from name and encrypted watchlist
3. VerifySig(pk,(x,seeds,name), σ)

Roadmap for This Talk

- Main idea of off-line ecash [CFN89 + CL02] and compact ecash [CHL05] ✔

- Balancing anonymity and accountability:
  - How to prevent money laundering [CHL06] ✔
  - How to trace rogue users' transactions ✔
  - How to implement authorized watchlists [KLN23] ✔

Conclusions

- Many interesting topics, we only covered a small subset.
- The Goldreich book is good reading, and you should be able to read it on your own.
- Other topics to explore: multi-party computation, two-party computation

- Some upcoming events if you are able to travel: https://iacr.org/schools/