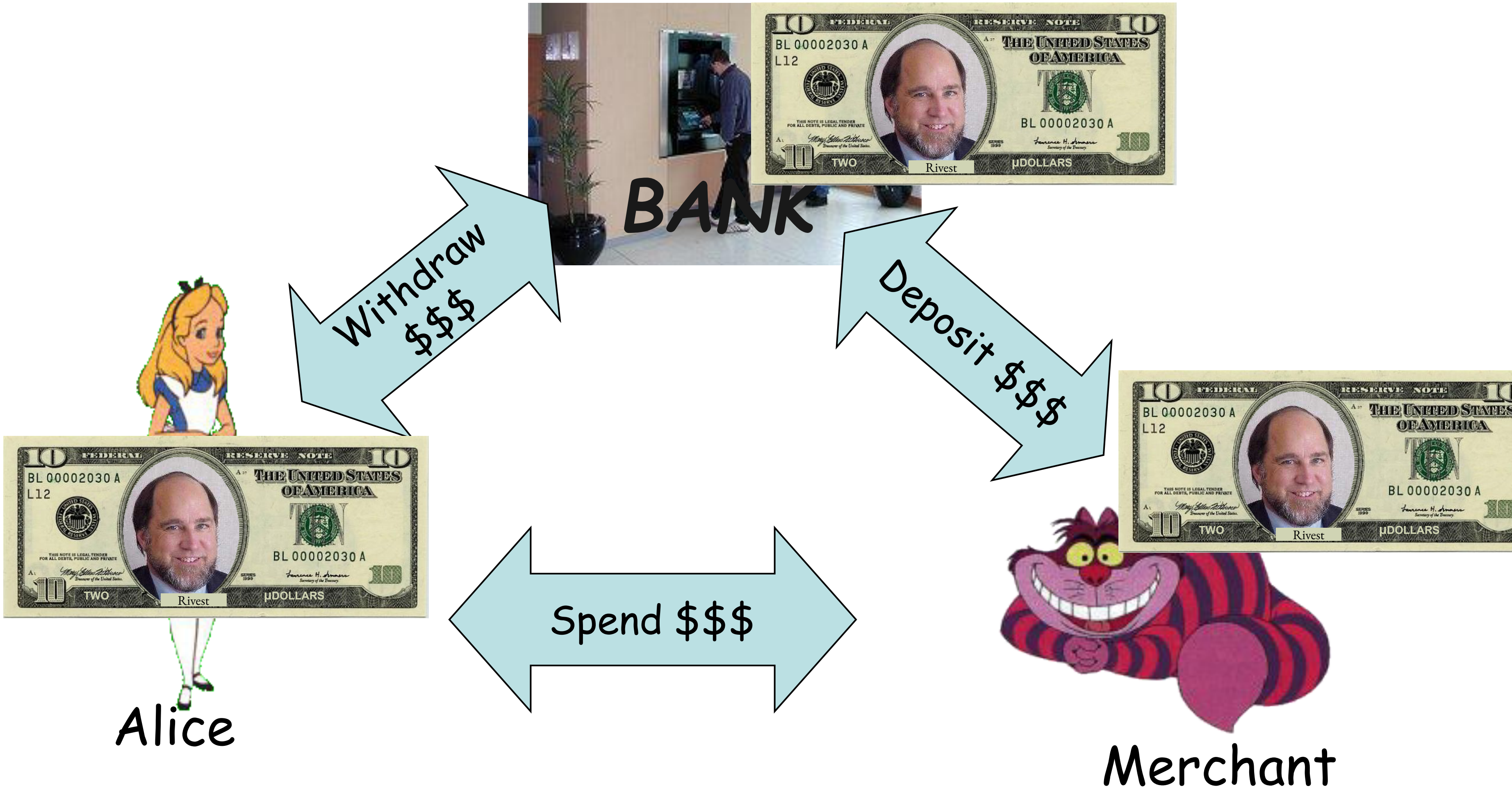
Zero-Knowledge Proofs for Balancing Privacy and Accountability

Anna Lysyanskaya
Brown University

The Money Cycle



BANK

Withdraw $$$

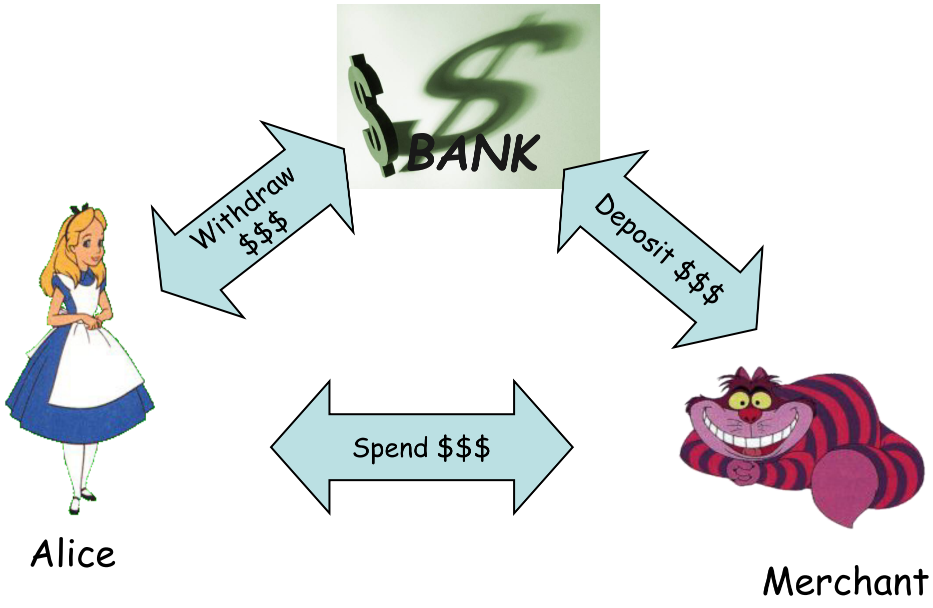
Deposit $$$

Spend $$$

Alice

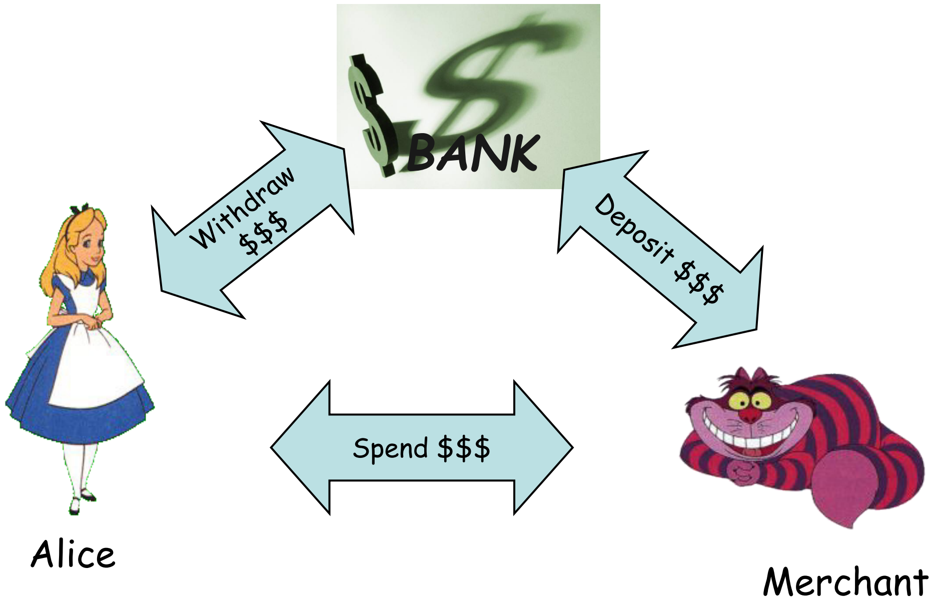
Merchant

The Money Cycle



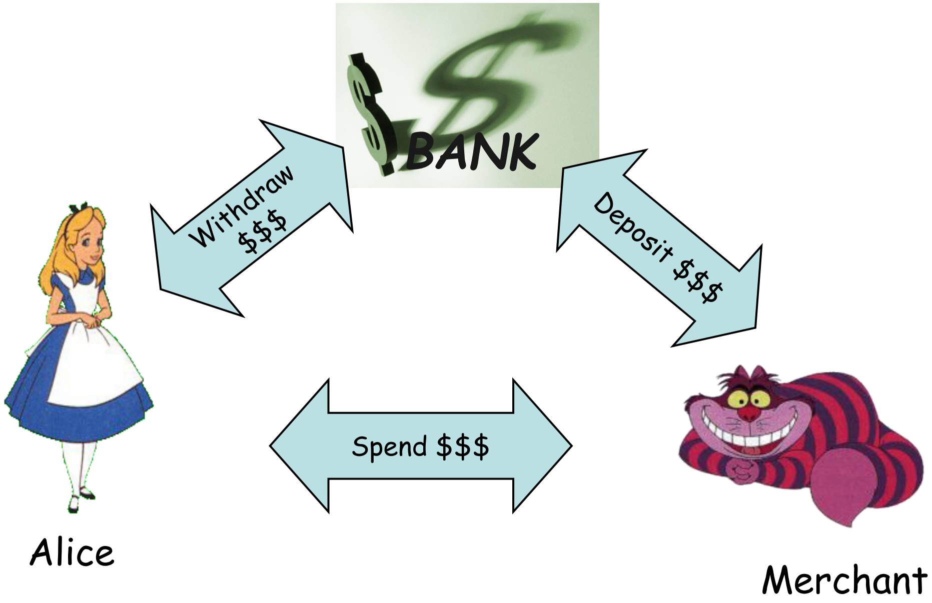
- Three protocols: Withdraw, Spend, Deposit
- Desirable properties:
  - can't forge/copy money
  - can't trace how cash was spent

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]

- What to standardize to make this a reality

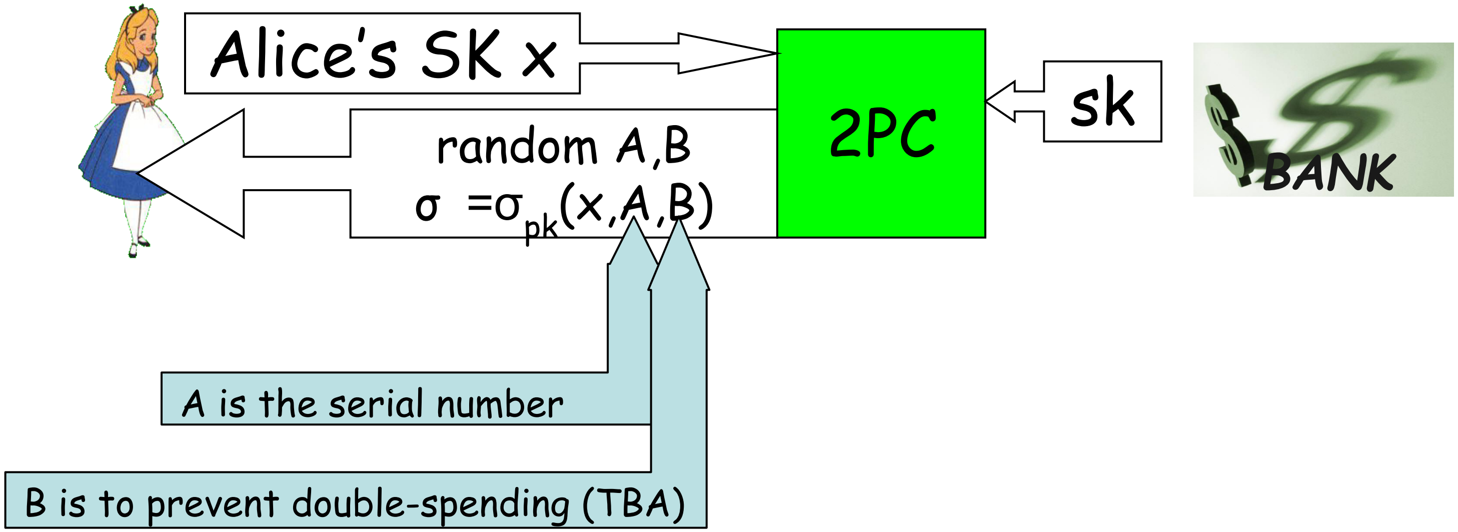
Warning: there might be a pop quiz…

Main Idea of Off-Line Ecash

- Building blocks – we will optimize them later:
  - digital signatures
  - secure two-party computation
  - ZK proofs of knowledge

Main Idea of Off-Line Ecash

- WITHDRAW a coin under Bank's public key pk:



Alice's SK x

2PC

sk

BANK

random A,B

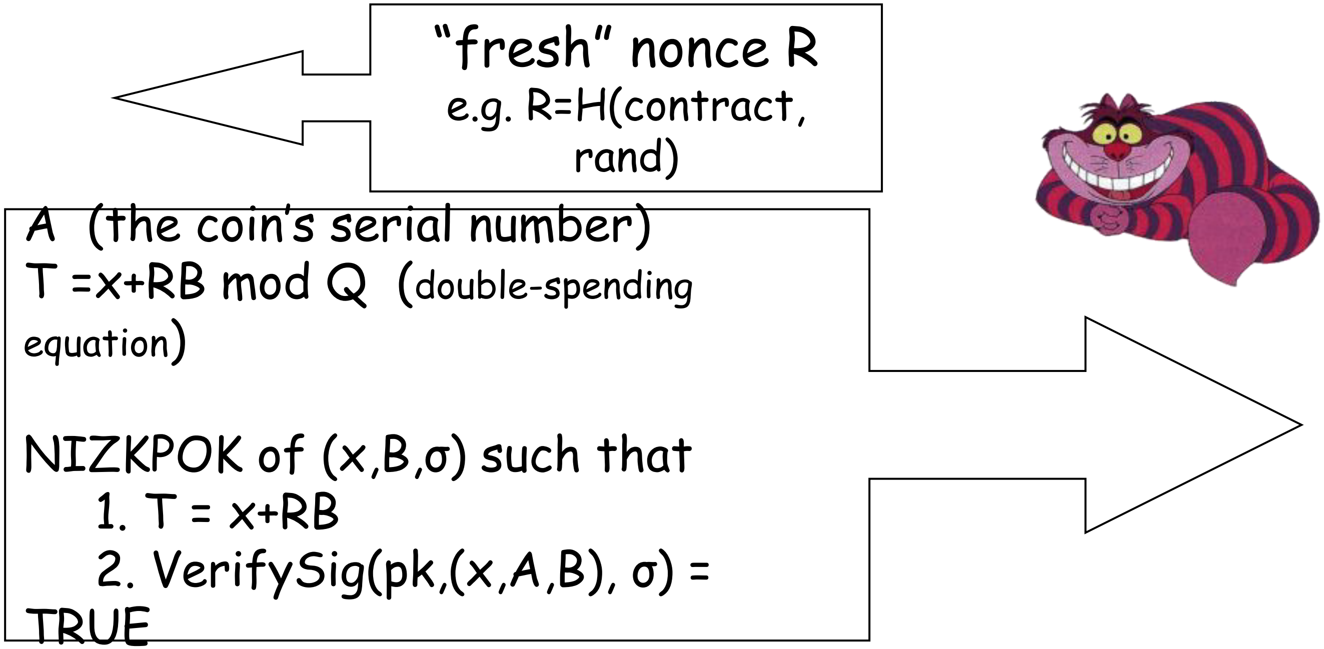
$\sigma = \sigma_{pk}(x,A,B)$

A is the serial number

B is to prevent double-spending (TBA)

Main Idea of Off-Line Ecash

- SPEND:



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

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

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

Main Idea of Off-Line Ecash

- DEPOSIT:

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

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, $R \neq R'$
$T = x+RB \bmod Q$, $T' = x+R'B\text{mod}$
Q
solve for x, id and punish Alice

R

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

NIZKPOK of (x,B,σ) such that
1. $T = x+RB$
2. $\text{VerifySig}(pk,(x,A,B), \sigma) =$ TRUE

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

User Privacy

The ZK simulator picks random A,T, creates a simulated proof.

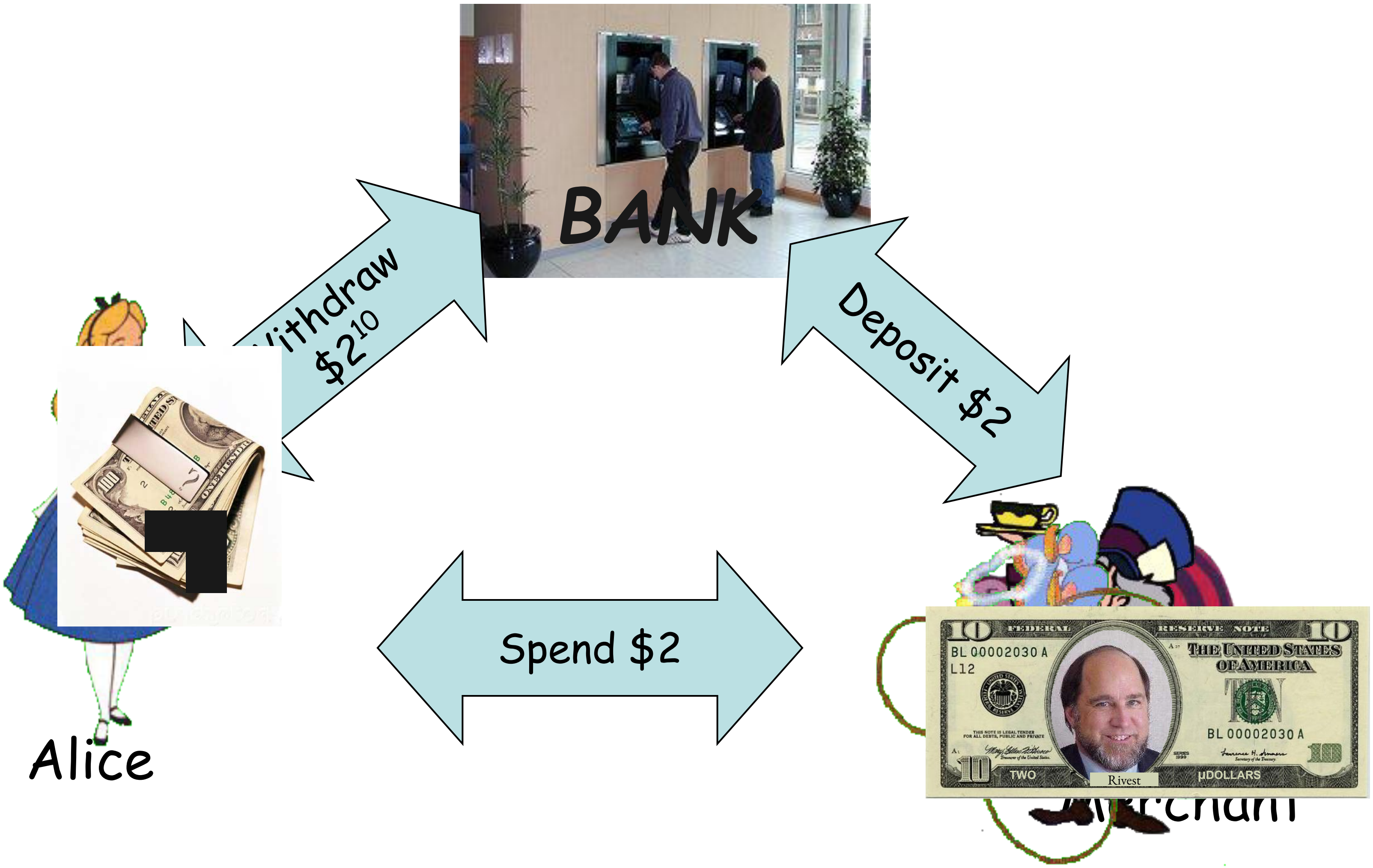
R

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

NIZKPOK of (x, B, σ) such that
1. $T = x + RB$
2. $\mathrm{VerifySig}(pk, (x, A, B), \sigma) =$ TRUE

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

Real-Life Money (again)



BANK

Withdraw $\$2^{10}$

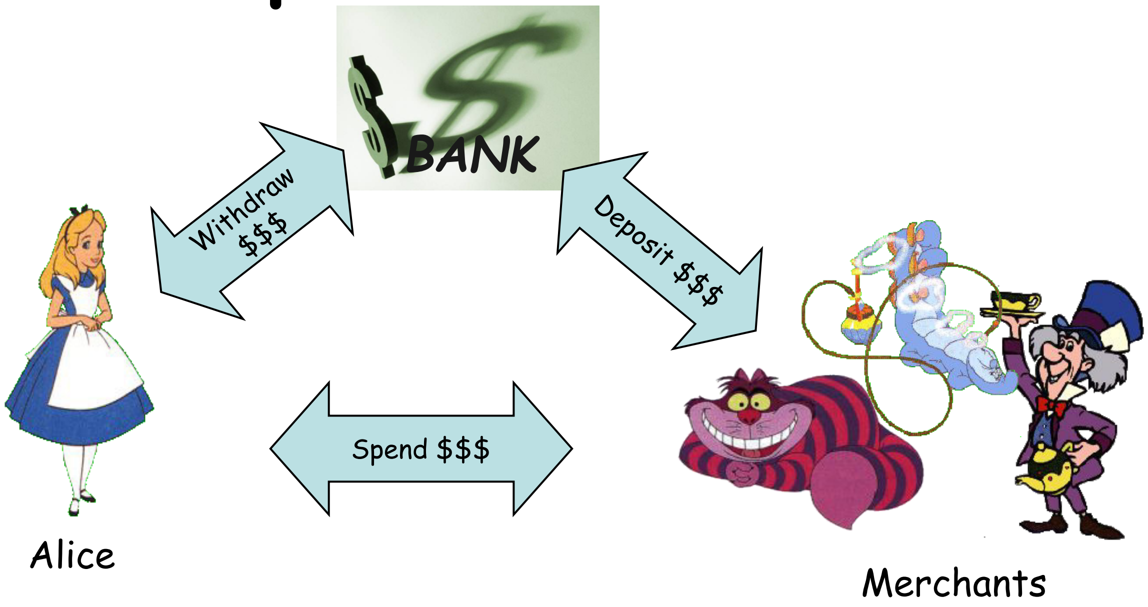
Deposit $2

Spend $2

Alice

Merchant

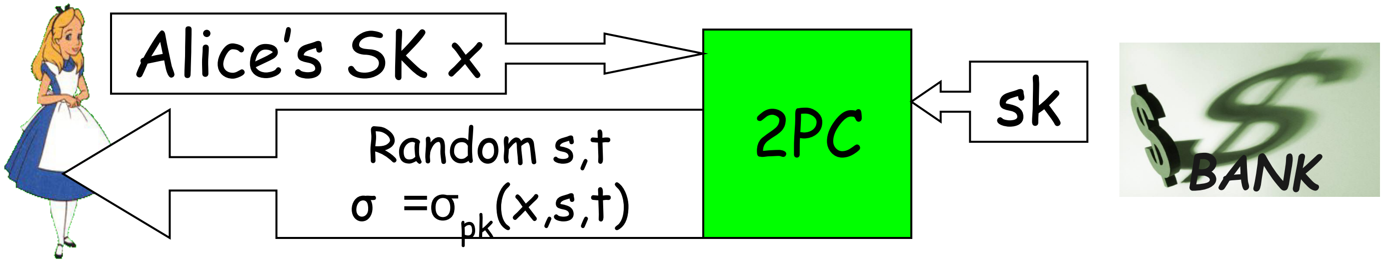
Compact Ecash



- 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)

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. $1 \leq i \leq N$
2. $A = F_s(i)$
3. $T = x+RF_t(i)$
4. $\mathrm{VerifySig}(pk,(x,s,t), \sigma) = \mathrm{TRUE}$

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

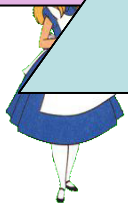
Compact Ecash: Main Idea [CHL05]

- WITHDRAW $

Sup

Privacy for Alice: the ZK simulator can pick random A,T, simulate proof

-



$A = F_s(i)$ (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. $1 \leq i \leq N$
2. $A = F_s(i)$
3. $T = x+RF_t(i)$
4. $\text{VerifySig}(pk,(x,s,t), \sigma) = \text{TRUE}$

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]

- What to standardize to make this a reality

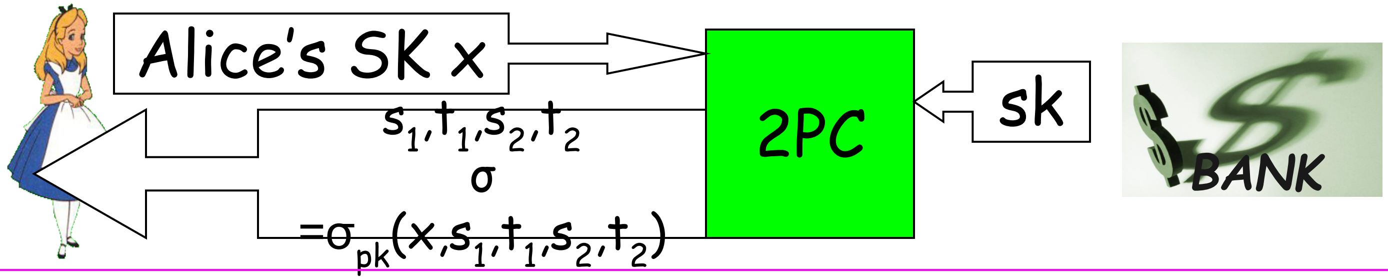
POP QUIZ:

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

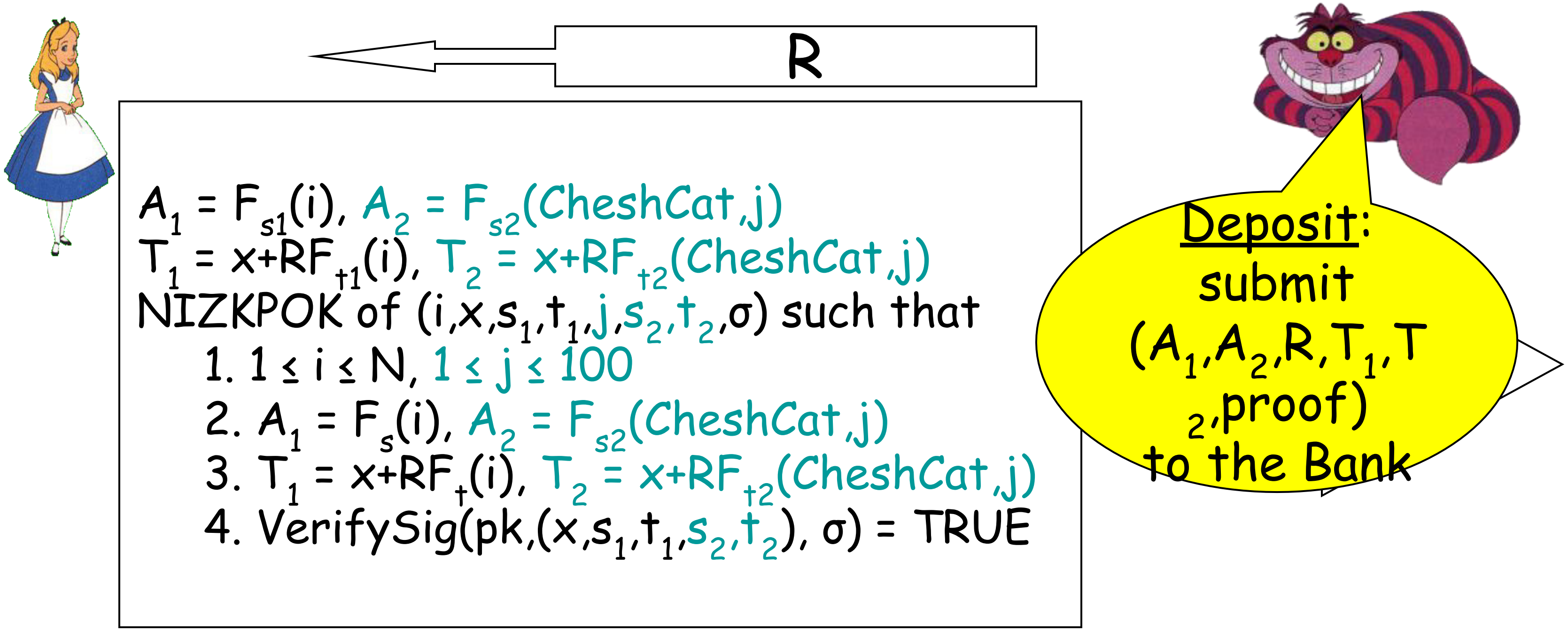
Preventing Money Laundering [CHL06]

- WITHDRAW $N:



s_1,t_1,s_2,t_2
$\sigma = \sigma_{pk}(x,s_1,t_1,s_2,t_2)$

- SPEND the ith coin; this is the jth time with this Merchant



$A_1 = F_{s1}(i)$, $A_2 = F_{s2}(\text{CheshCat},j)$
$T_1 = x+RF_{t1}(i)$, $T_2 = x+RF_{t2}(\text{CheshCat},j)$
NIZKPOK of $(i,x,s_1,t_1,j,s_2,t_2,\sigma)$ such that

1. $1 \leq i \leq N$, $1 \leq j \leq 100$
2. $A_1 = F_s(i)$, $A_2 = F_{s2}(\text{CheshCat},j)$
3. $T_1 = x+RF_t(i)$, $T_2 = x+RF_{t2}(\text{CheshCat},j)$
4. VerifySig$(pk,(x,s_1,t_1,s_2,t_2), \sigma)$ = 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 simulator can create the view w/o knowing the user's identity:
pick random A_1, T_1, A_2, T_2
simulate proof

$A_1 = F_{s1}(i)$, $A_2 = F_{s2}(\text{CheshCat},j)$
$T_1 = x + RF_{t1}(i)$, $T_2 = x + RF_{t2}(\text{CheshCat},j)$
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. $A_1 = F_s(i)$, $A_2 = F_{s2}(\text{CheshCat},j)$
3. $T_1 = x + RF_t(i)$, $T_2 = x + RF_{t2}(\text{CheshCat},j)$
4. $\text{VerifySig}(pk,(x,s_1,t_1,s_2,t_2), \sigma) = \text{TRUE}$

Deposit: submit $(A_1,A_2,R,T_1,T_2,\text{proof})$ 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 LM06]

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. $1 \leq i \leq N$
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. $\text{VerifySig}(pk,(x,s,t,u,v,L,z_1,z_2,z_3), \sigma)$

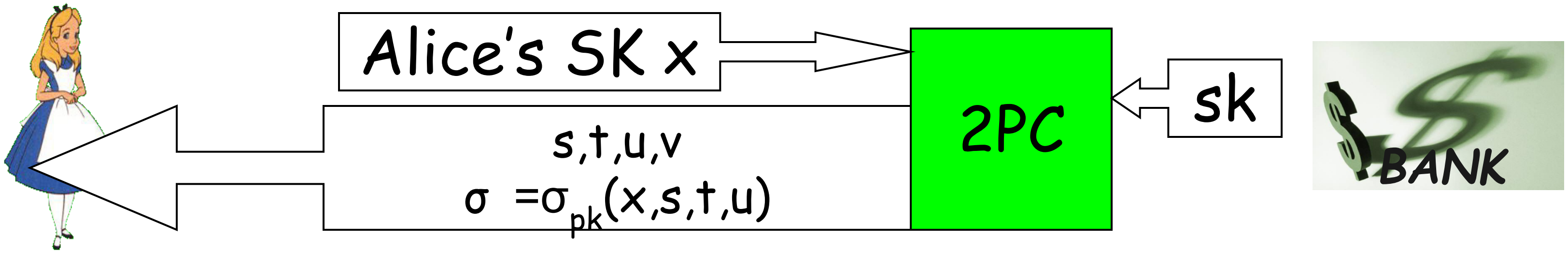
POP QUIZ 3:

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:



Alice's SK x

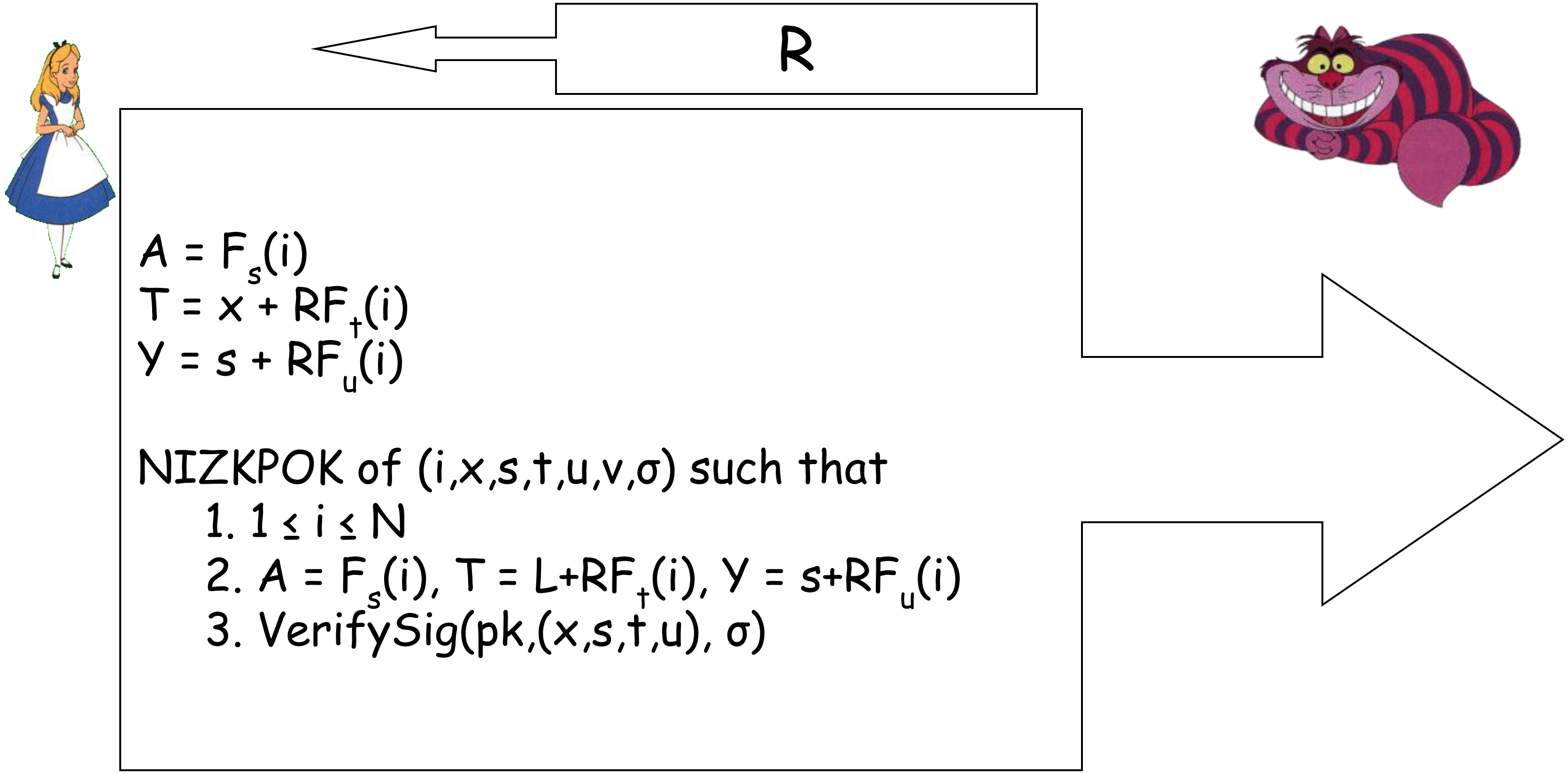
2PC

sk

s,t,u,v

$\sigma = \sigma_{pk}(x,s,t,u)$

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



R

$A = F_s(i)$
$T = x + RF_t(i)$
$Y = s + RF_u(i)$

NIZKPOK of (i,x,s,t,u,v,σ) such that

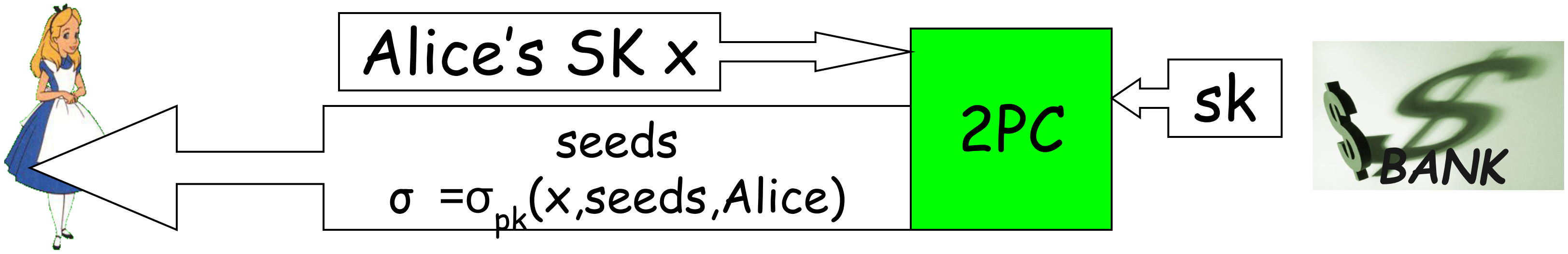
1. $1 \leq i \leq N$
2. $A = F_s(i)$, $T = L+RF_t(i)$, $Y = s+RF_u(i)$
3. $\mathrm{VerifySig}(pk,(x,s,t,u), \sigma)$

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 [CHL06]
  - How to implement authorized watchlists [KLN23]
- What to standardize to make this a reality

Watchlists [KLN23]

- WITHDRAW:



Alice's SK x

2PC

sk

seeds

σ =σ_{pk}(x,seeds,Alice)

- SPEND:

Encrypted watchlist

E-coin's info (as before)

Escrow = { Enc(name) if name is on the watchlist
Enc(random) otherwise

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

NIZKPOK of (x,seeds,name,σ) such that

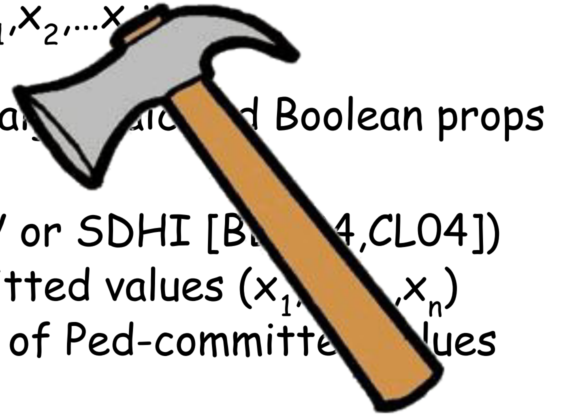
1. E-coin was computed correctly from seeds
2. Escrow was computed correctly from name and encrypted watchlist
3. VerifySig(pk,(x,seeds,name), σ)

Roadmap for This Talk

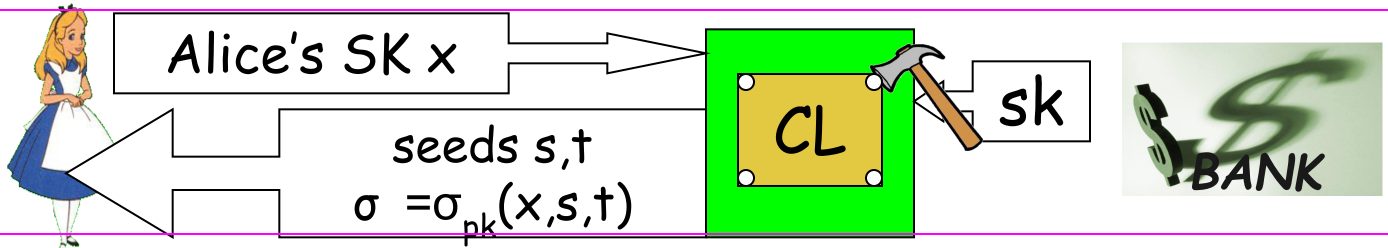
- Main idea of off-line ecash [CFN89 + CL02] and compact ecash [CHL05]
- Balancing anonymity and accountability:
  - How to prevent money laundering [CHL06]
  - How to trace rogue users' transactions [CHL06]
  - How to implement authorized watchlists [KLN23]
- What to standardize to make this a reality

Building Blocks to Standardize

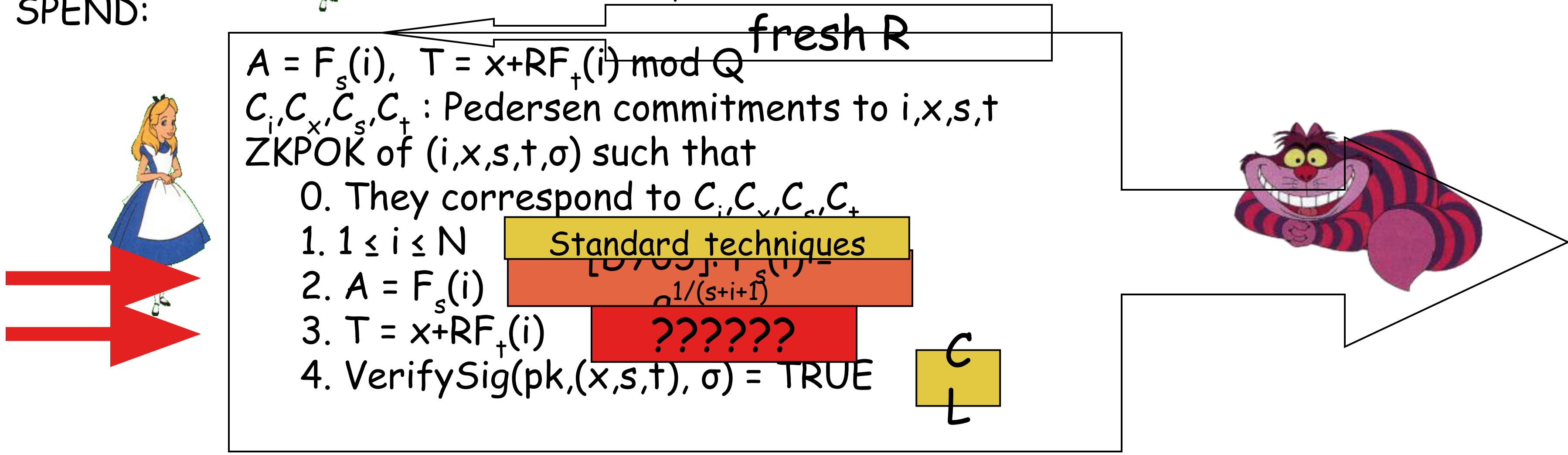
- Pedersen commitments [Ped92]:
  - If G is a group with generators $g_1,g_2, \dots, g_n$, h commit to $x_1,x_2,\dots x_n$:
    $C = g_1^{x1}g_2^{x2}\dots g_n^{xn}h^r$ for random $r < |G|$
  - [Krenn,Orrù,ZKProof'21]: ZKPOKs of committed values w algebraic and Boolean props
- CL sigs -- the one that's a serious contender is BBS+
  - Efficient, provably secure sig (Strong RSA [CL02], LRSW or SDHI [BB04,CL04])
  - Efficient protocol for getting a sig on a set of Ped-committed values $(x_1,\dots,x_n)$
  - Efficient protocol for proving knowledge of a sig on a set of Ped-committed values



- WITHDRAW:



- SPEND:



$A = F_s(i)$, $T = x+RF_t(i) \bmod Q$

C_i,C_x,C_s,C_t : Pedersen commitments to i,x,s,t

ZKPOK of (i,x,s,t,σ) such that

0. They correspond to C_i,C_x,C_s,C_t
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

Standard techniques

[DY05]: $F_s(i) = g^{1/(s+i+1)}$

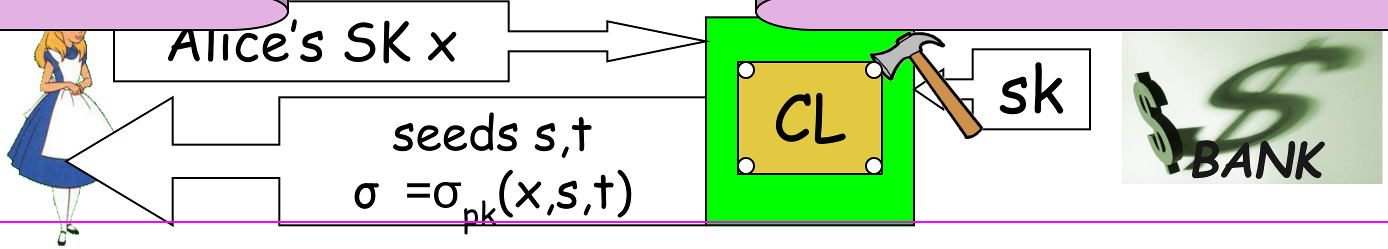
??????

Building Blocks to Standardize

- Pedersen commitm
  - ops
- CL si
  - Eff CL04])
  - Eff x_n)
  - F lues

Suppose i'th coin is spent twice.
Same coin => same A
Spent twice: two random R's,
with high prob, $R_1 \neq R_2$
$T_1 = g^x(F_t(i))^{R1}$, $T_2 = g^x(F_t(i))^{R2}$
solve for $F_t(i) = (T_1/T_2)^{1/(R1-R2)}$
solve for $g^x = T_1/(F_t(i)^{R1})$

- WITHDRAW:



- SPEND:

$A = F_s(i)$, $T = x+RF_t(i)$ mod Q
C_i,C_x,C_s,C_t : Pedersen commitments to i,x,s,t
ZKPOK of (i,x,s,t,σ) such that
0. They correspond to C_i,C_x,C_s,C_t
1. $1 \leq i \leq N$ Standard techniques
2. $A = F_s(i)$ [DY05]: $F_s(i) = g^{1/(s+i+1)}$
3. $T = x+RF_t(i)$??????
4. VerifySig(pk,(x,s,t), σ) = TRUE CL

Building Blocks to Standardize

- Pedersen commitments [Ped92]:
  - If G is a group with generators $g_1, g_2, \ldots, g_n$, h commit to $x_1, x_2, \ldots x_n$:
    $C = g_1^{x1} g_2^{x2} \ldots g_n^{xn} h^r$ for random $r < |G|$
  - [Krenn,Orrù,ZKProof'21]: NIZKPOKs of committed values w algebraic and Boolean props
- CL sigs -- the one that's a serious contender is BBS+
  - Efficient, provably secure sig (Strong RSA [CL02], LRSW or SDHI [BBS04,CL04])
  - Efficient protocol for getting a sig on a set of Ped-committed values $(x_1, x_2, \ldots, x_n)$
  - Efficient protocol for proving knowledge of a sig on a set of Ped-committed values
- Dodis-Yampolsky PRF with proof protocols (based on NIZKPOKs above)
- For watchlists: ElGamal encryption
  - NIZK proof that escrow was computed correctly is also based on the same NIZKPOK proof systems

LEGO · In stock

Identity and Landscape Kit 2000430 ...

Conclusion + Discussion

- In theory, we can have our cake and eat it too! What's stopping us in practice?
    - Policy makers are not aware/mistrustful of these tools?
        - https://www.aclu.org/documents/paths-toward-acceptable-public-digital-currency
    - Lack of standards and practical implementations?
        - https://datatracker.ietf.org/doc/draft-irtf-cfrg-bbs-signatures/
        - Hyperledger project's implementation
    - What are the practical use cases? E.g. what does the Federal Reserve want/need a digital dollar to look like?