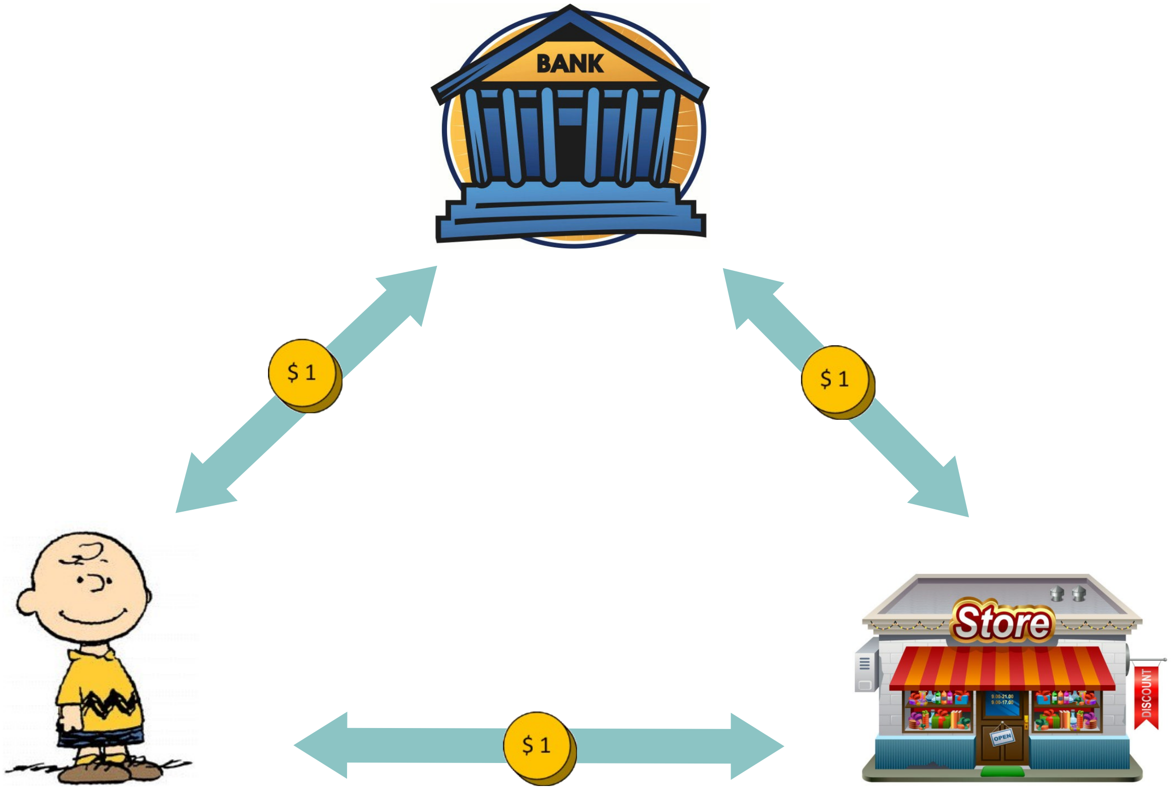
Anonymous Transferable E-cash

Foteini Baldimtsi	Boston University
Melissa Chase	Microsoft Research
Georg Fuchsbauer	IST Austria
Markulf Kohlweiss	Microsoft Research

PKC'15, Maryland, USA

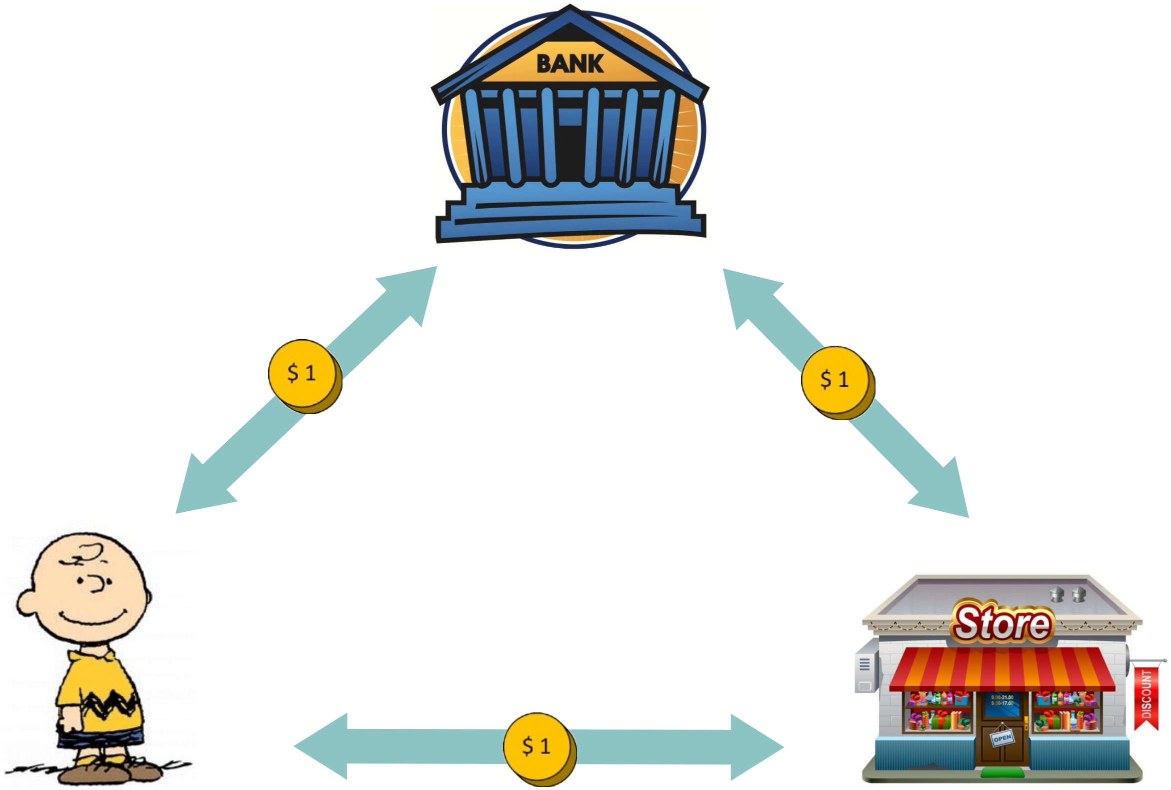
Electronic cash

Simulating traditional cash



Electronic cash

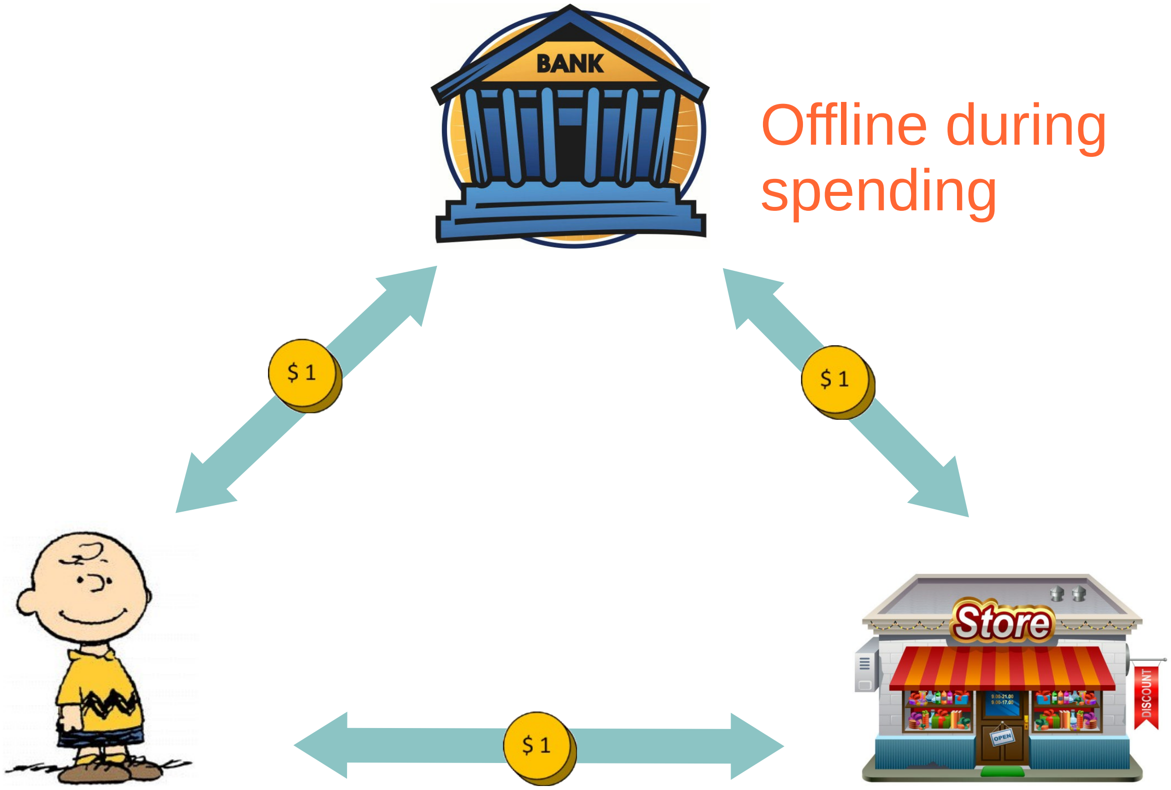
Simulating traditional cash



Anonymous

Electronic cash

Simulating traditional cash

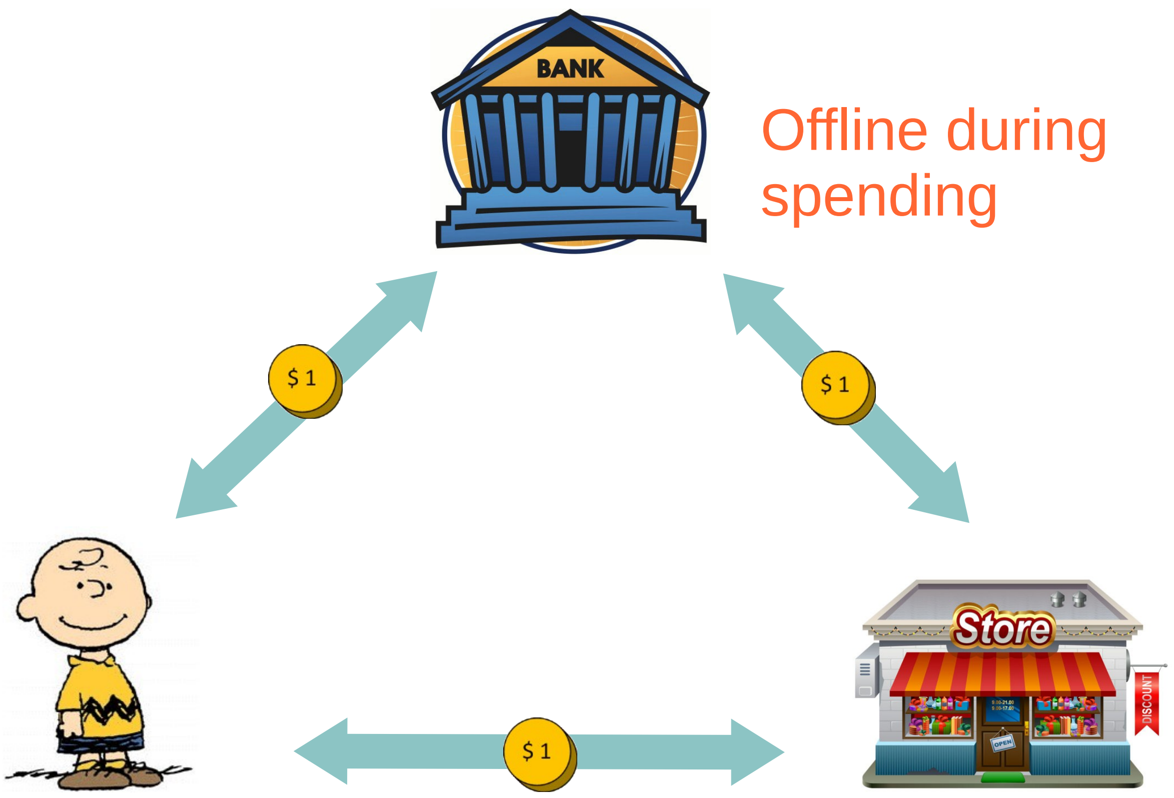


Offline during spending

Anonymous

Electronic cash

Simulating traditional cash

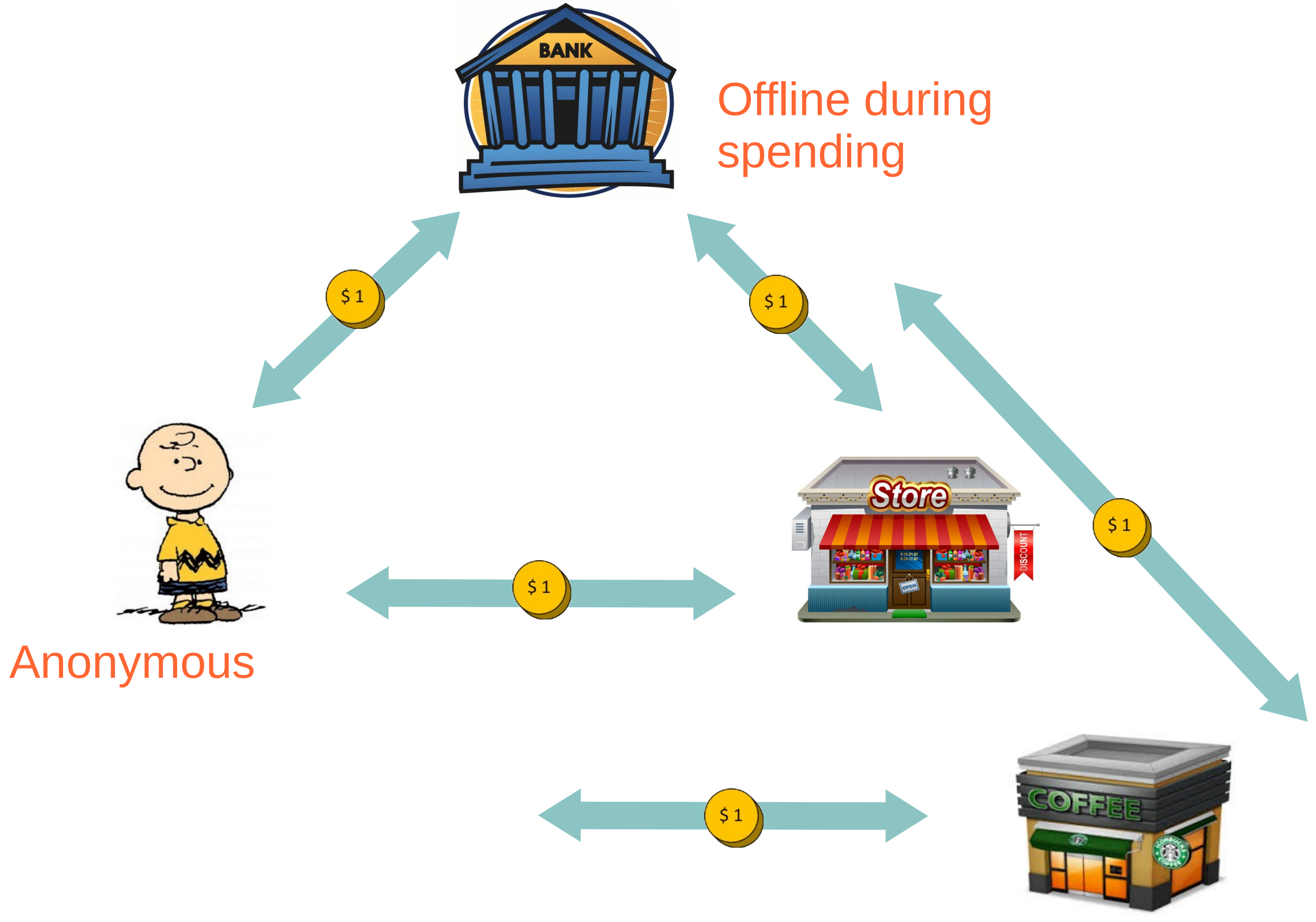


Offline during spending

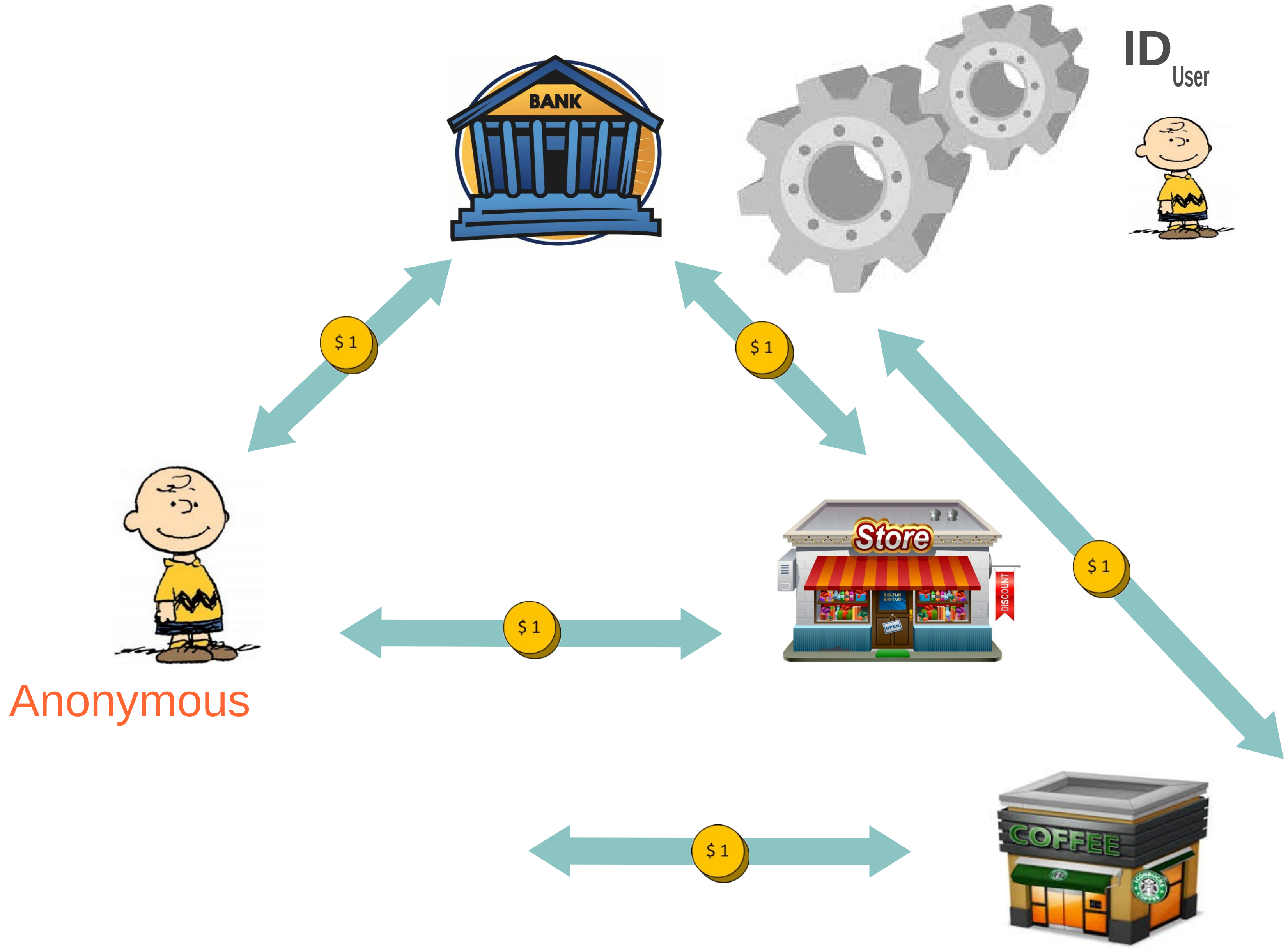
Anonymous

$ 1 unforgeable

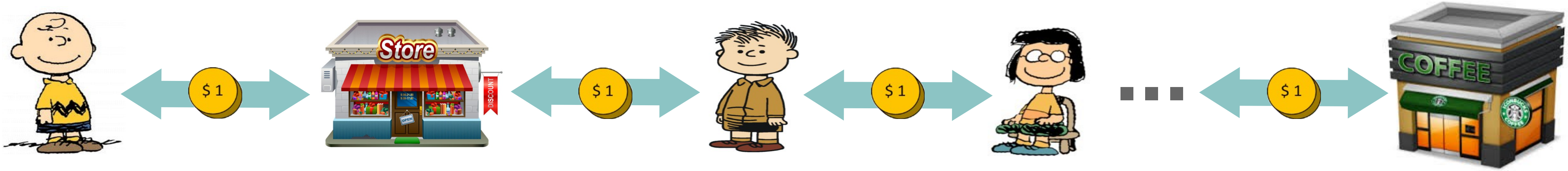
Double spending



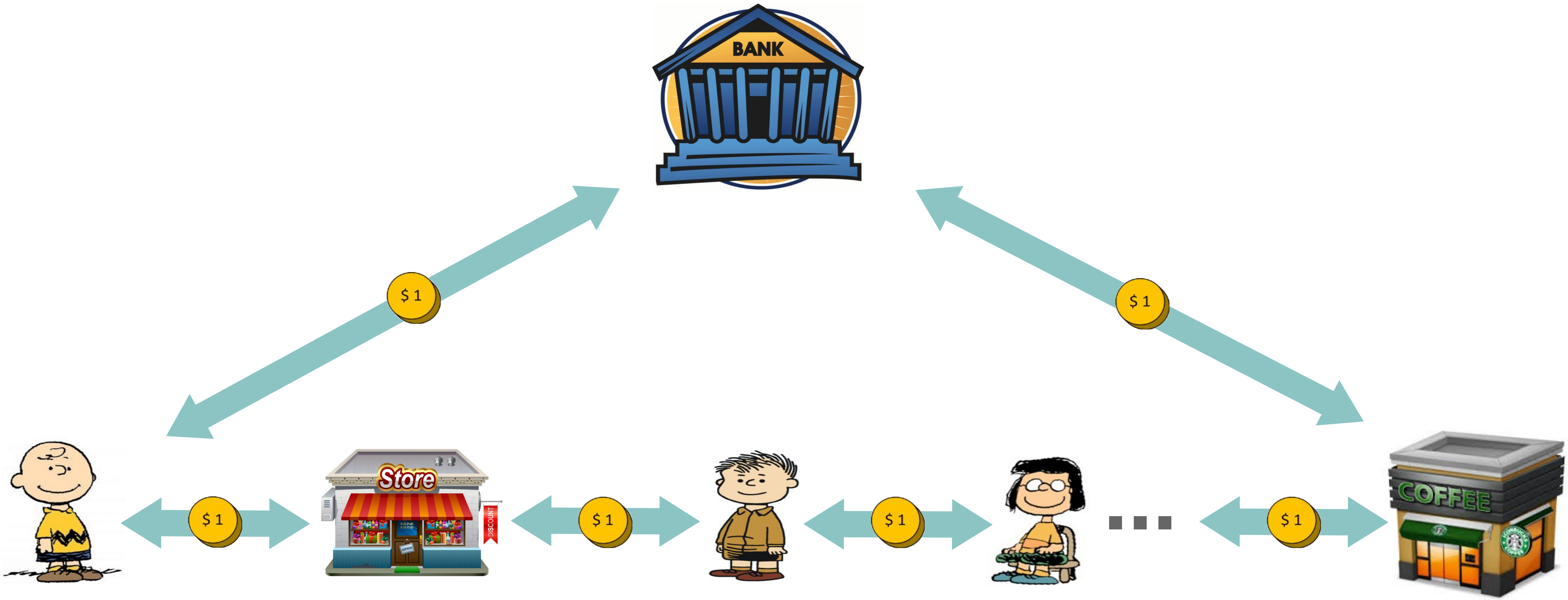
Double spending



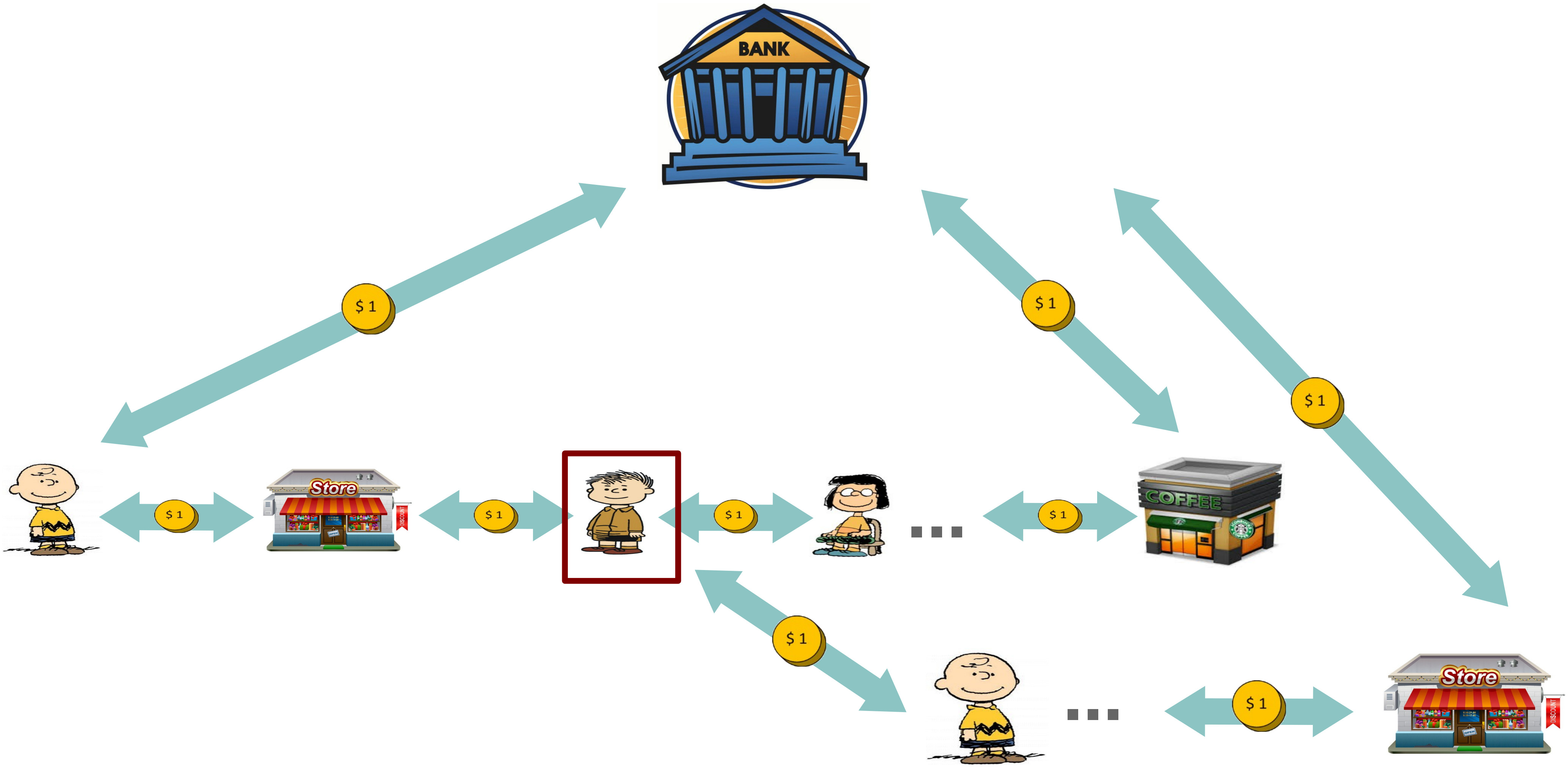
What if?



Transferable E-Cash



Double spending detection



Our Contributions

The first practical, truly anonymous transferable e-cash scheme

Our Contributions

The first practical, truly anonymous transferable e-cash scheme

- ✔ On double spending, only the identity of the malicious user is revealed [FPV'09]
- ✔ No trusted 3rd party that can de-anonymize users [BCFGST'11]

Our Contributions

The first practical, truly anonymous transferable e-cash scheme

- ✔ On double spending, only the identity of the malicious user is revealed [FPV'09]
- ✔ No trusted 3rd party that can de-anonymize users [BCFGST'11]

- New Detailed definitions of transferable e-cash security
- New Generic construction based on malleable signatures
- New An efficient double-spending detection technique

Transferable E-Cash Security

Unforgeability: An adversary cannot spend more coins than the number of coins he withdrew.

Double Spending: An adversary cannot spend a coin twice (double-spend) without his identity being revealed.

Transferable E-Cash Security

Unforgeability: An adversary cannot spend more coins than the number of coins he withdrew.

Double Spending: An adversary cannot spend a coin twice (double-spend) without his identity being revealed.

Chaum & Pedersen '92:

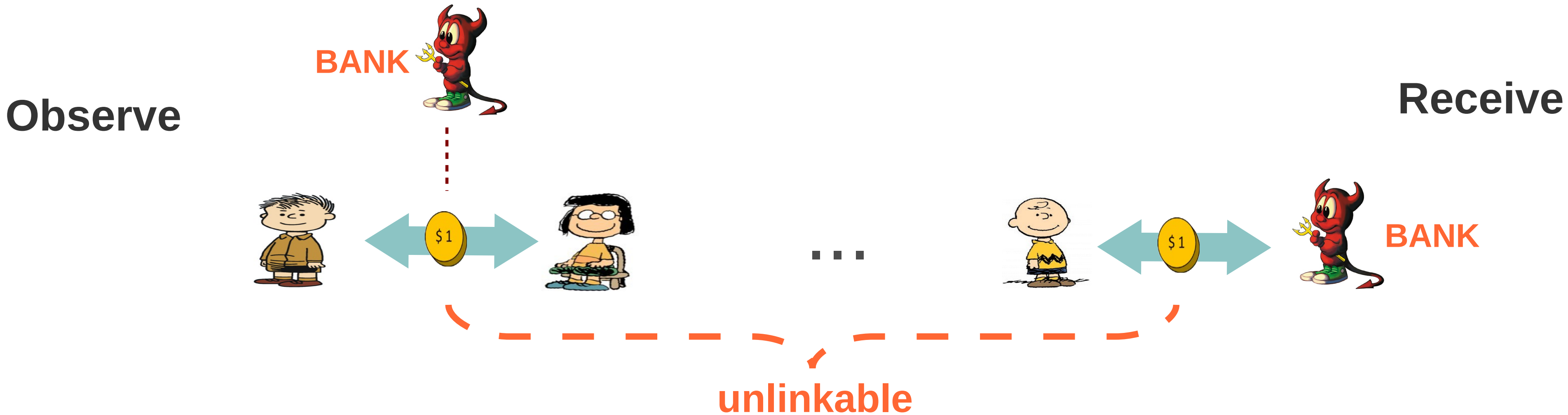
An unbounded adversary can always recognize coins he has already owned

Canard & Gouget '08:

A bounded adversary, impersonating the bank, can always recognize coins he has already owned (using the DS mechanism)

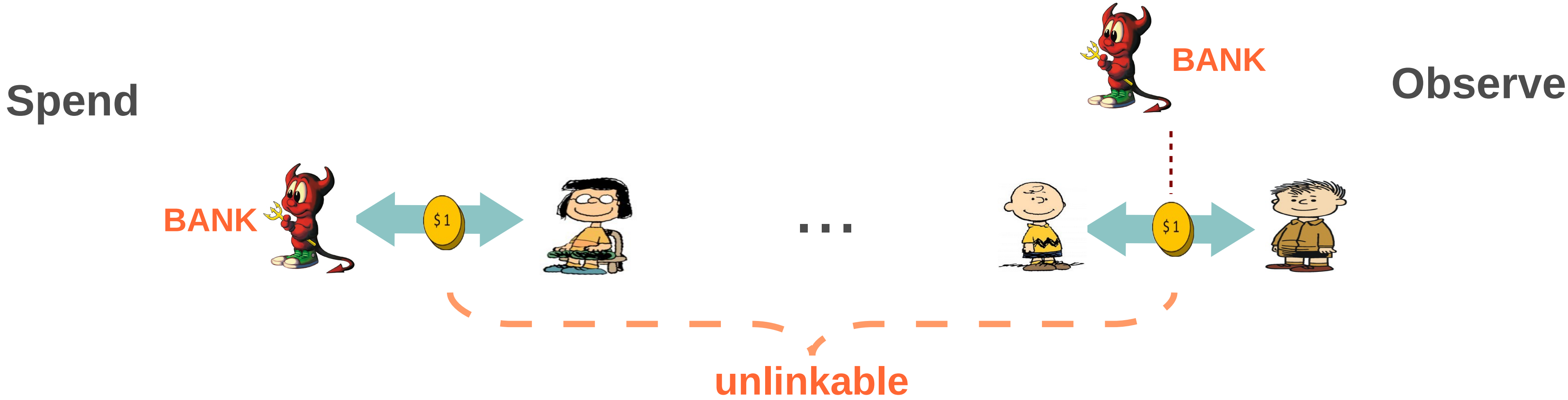
Transferable E-Cash Anonymity

- Observe-then-Receive (**OtR**): an attacker, impersonating the bank, cannot link a coin he receives to a previously (passively) observed transfer between honest users



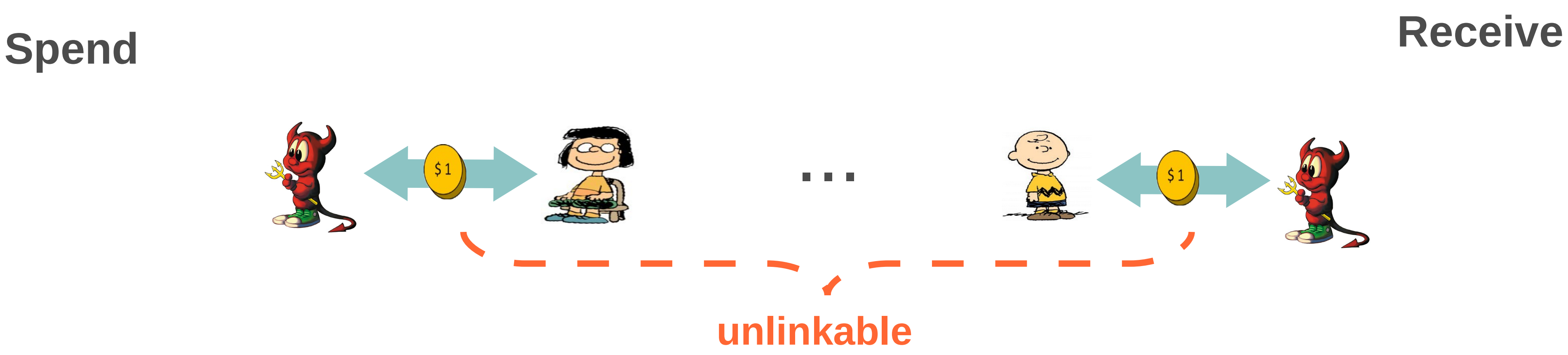
Transferable E-Cash Anonymity

- Observe-then-Receive (**OtR**)
- Spend-then-Observe (**StO**): an attacker (impersonating the bank) cannot link a passively observed coin transferred between two honest users to a coin he has already owned



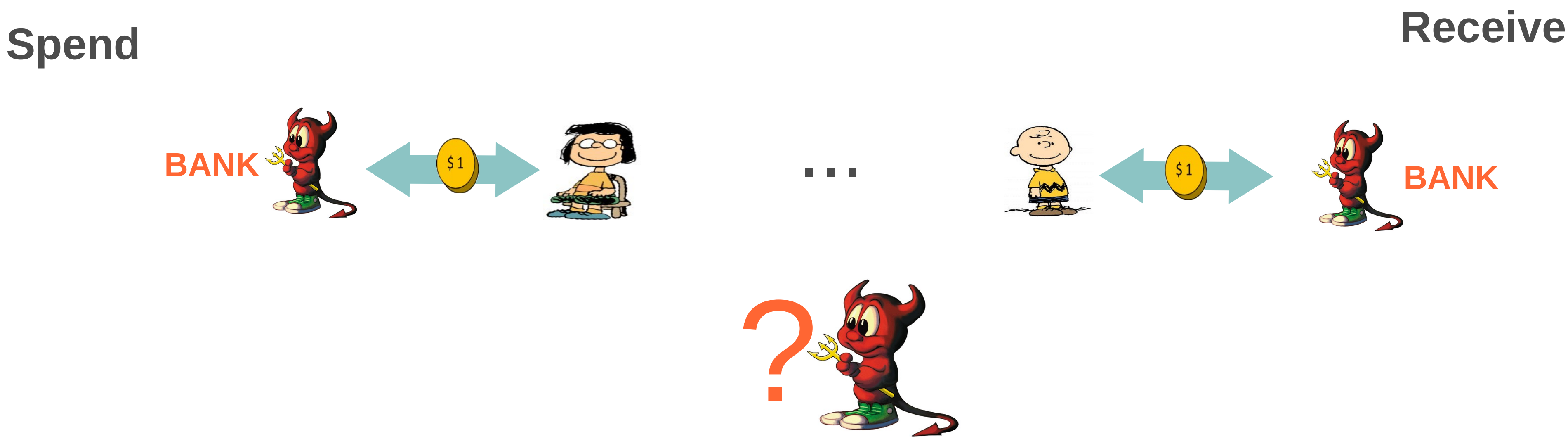
Transferable E-Cash Anonymity

- Observe-then-Receive (**OtR**)
- Spend-then-Observe (**StO**)
- Spend-then-Receive (**StR**): when the bank is honest, an attacker cannot link two transactions involving the same coin



Transferable E-Cash Anonymity

- Observe-then-Receive (**OtR**)
- Spend-then-Observe (**StO**)
- Spend-then-Receive (**StR**)
- Spend-then-Receive*(**StR***): an adversary, impersonating the bank, receives a coin he owned before he shouldn't be able to recognize the "chain" of honest users the coin followed



Our Construction

$U_1(ID_1,pk_1,sk_1)....U_n(ID_n,pk_n,sk_n)$

Coin List: CL

Our Construction

$U_1(ID_1,pk_1,sk_1)....U_n(ID_n,pk_n,sk_n)$

Coin List: CL

$c = \sigma(SN, DS)$

where $SN=SN_1 || \ldots || SN_k$ & $DS=DS_1 || \ldots || DS_{k-1}$

Our Construction

$U_1(ID_1,pk_1,sk_1)....U_n(ID_n,pk_n,sk_n)$

Coin List: CL

$c = \sigma(SN, DS)$

where $SN = SN_1 \,||\, \ldots \,||\, SN_k$ & $DS = DS_1 \,||\, \ldots \,||\, DS_{k-1}$

If a double-spending happened, then in CL there will be 2 coins where:

$$SN = SN_1 \,||\, \ldots \,||\, SN_j \,||\, \ldots \,||\, SN_k$$

$$=$$

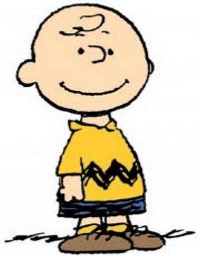
$$SN' = SN_1 \,||\, \ldots \,||\, SN'_j \,||\, \ldots \,||\, SN'_k$$

Double Spending Mechanism

$c = \sigma(SN, DS)$

sk_B, pk_B

(ID_1, pk_1, sk_1)



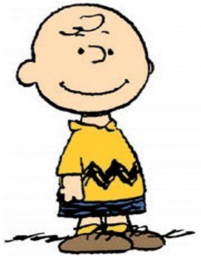
Withdrawal

Double Spending Mechanism

$c = \sigma(SN, DS)$

sk_B, pk_B

(ID_1, pk_1, sk_1)



$com(SN_1)$

pick SN_1

Withdrawal

Double Spending Mechanism

$c = \sigma(SN, DS)$

sk_B, pk_B

(ID_1, pk_1, sk_1)

$com(SN_1)$ ← pick SN_1

↔ 

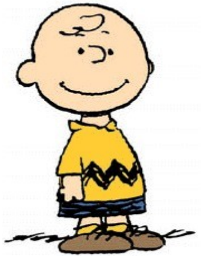
Withdrawal

Double Spending Mechanism

$c = \sigma(SN, DS)$

sk_B, pk_B

(ID_1, pk_1, sk_1)

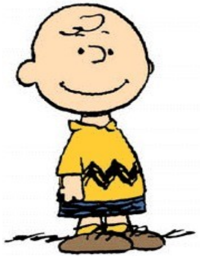


$com(SN_1)$ ←

pick SN_1

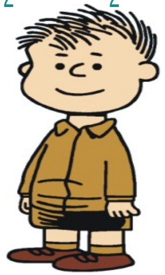
↔ $1

(ID_1, pk_1, sk_1)



$1 SN_1

(ID_2, pk_2, sk_2)

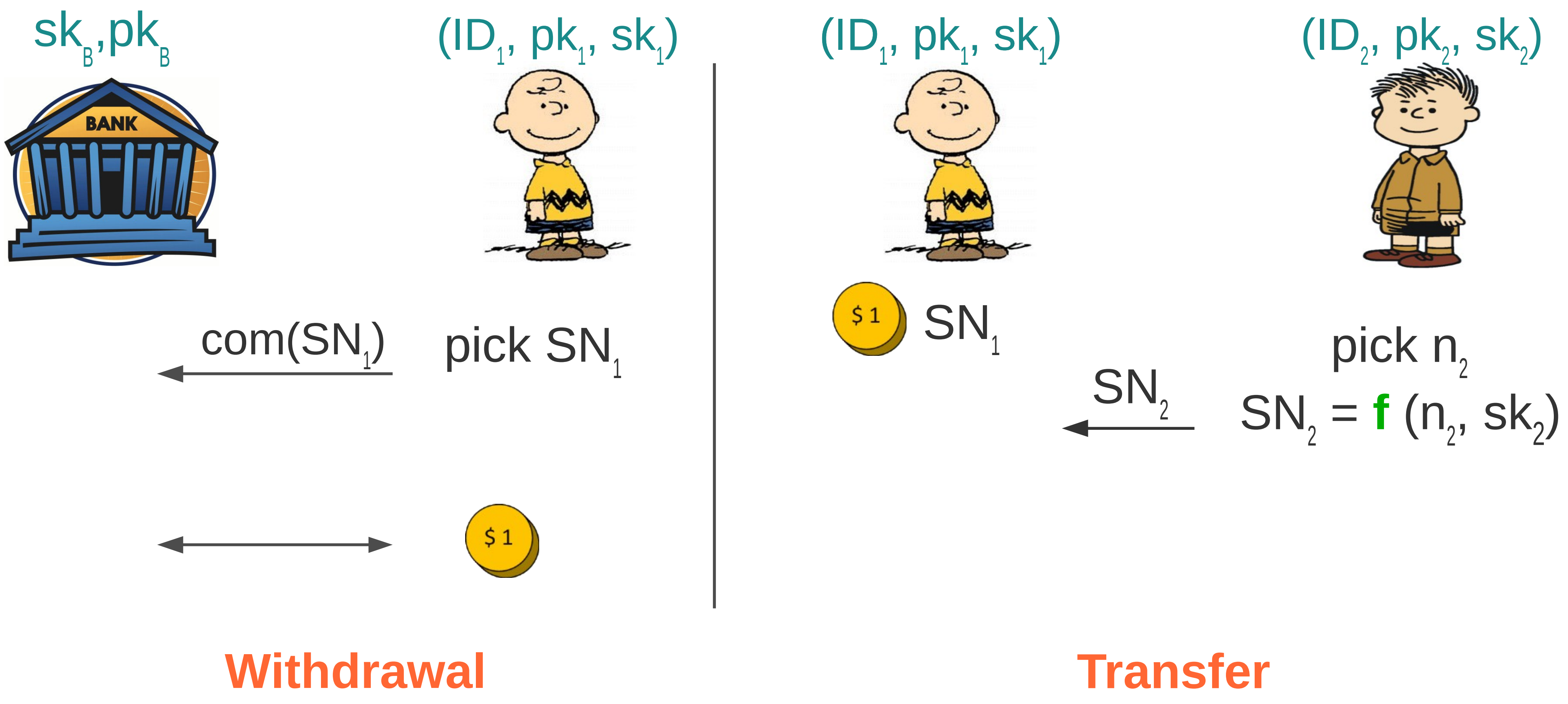


Withdrawal

Transfer

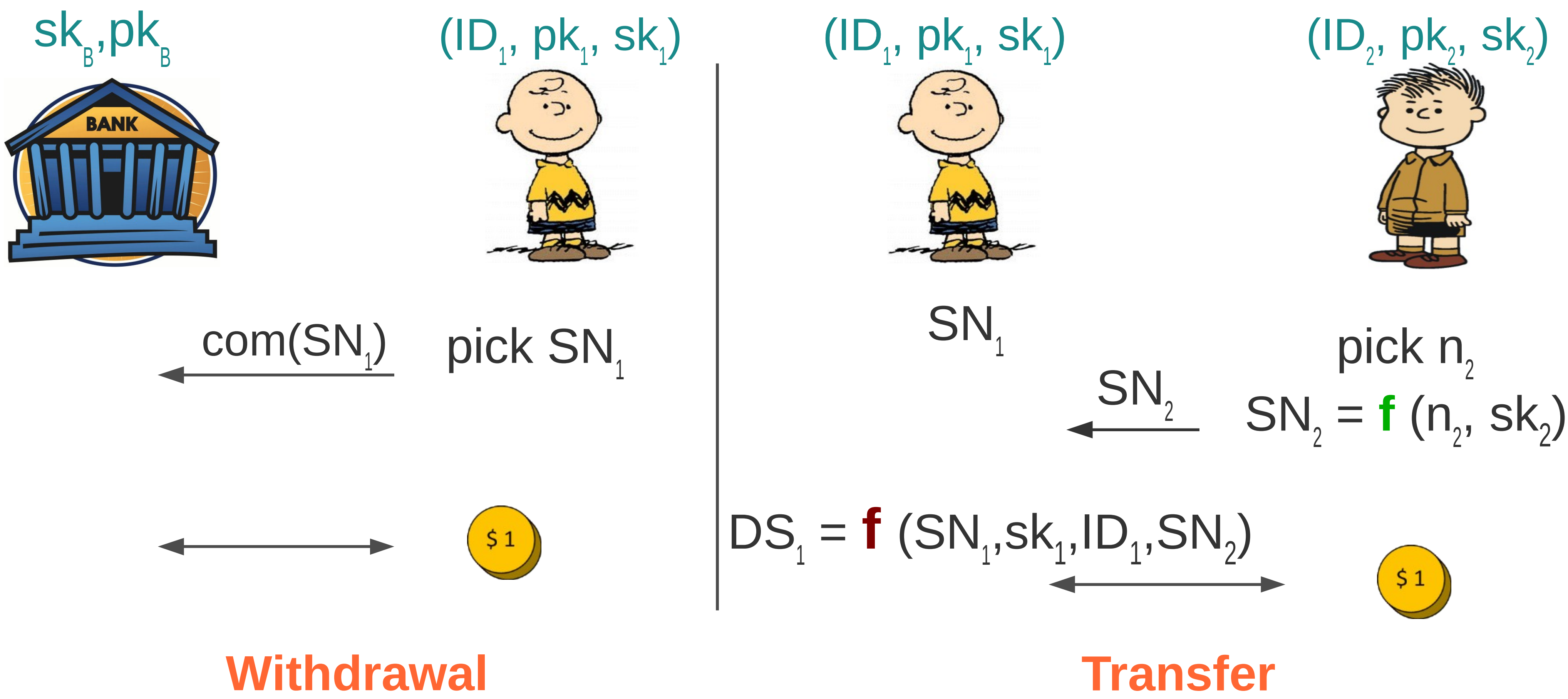
Double Spending Mechanism

$c = \sigma(SN, DS)$



Double Spending Mechanism

$c = \sigma(SN, DS)$

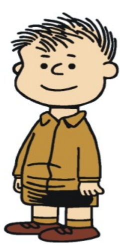


Double Spending Mechanism

(ID_1, pk_1, sk_1)

SN_1

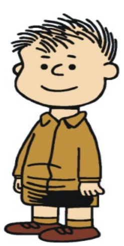
SN_2



Double Spending Mechanism

(ID_1, pk_1, sk_1)

SN_1



$\mathbf{SN_2}$

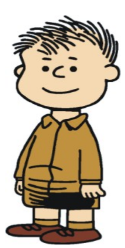
$DS_1 = \mathbf{f}\ (SN_1, sk_1, ID_1, \mathbf{SN_2})$

Double Spending Mechanism

(ID_1, pk_1, sk_1)

SN_1

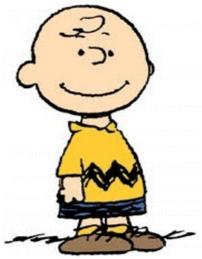
$\mathbf{SN_2}$



$DS_1 = \mathbf{f}\ (SN_1, sk_1, ID_1, \mathbf{SN_2})$

Double Spending Mechanism

(ID_1, pk_1, sk_1)

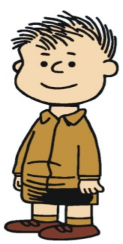


SN_1

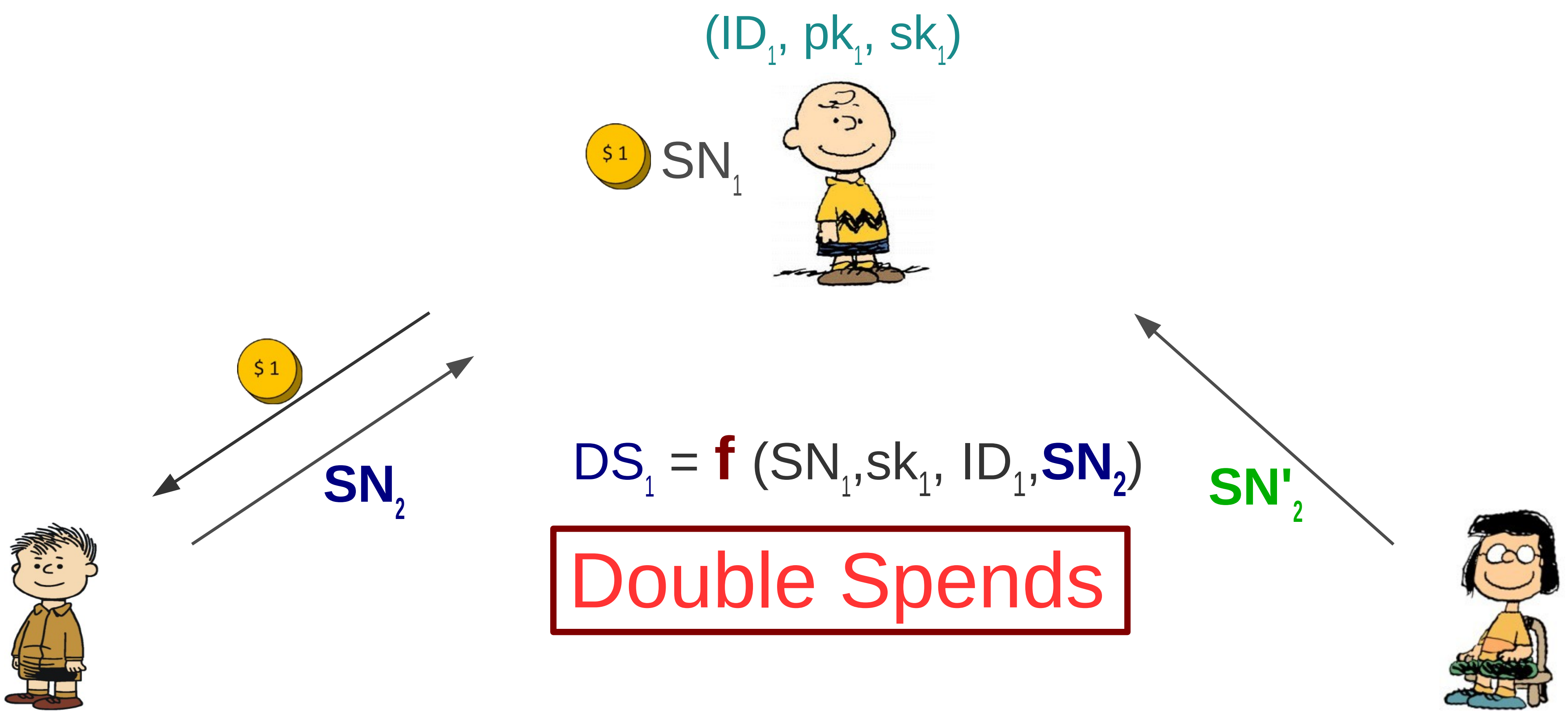
$\mathbf{SN_2}$

$DS_1 = \mathbf{f}\,(SN_1, sk_1, ID_1, \mathbf{SN_2})$

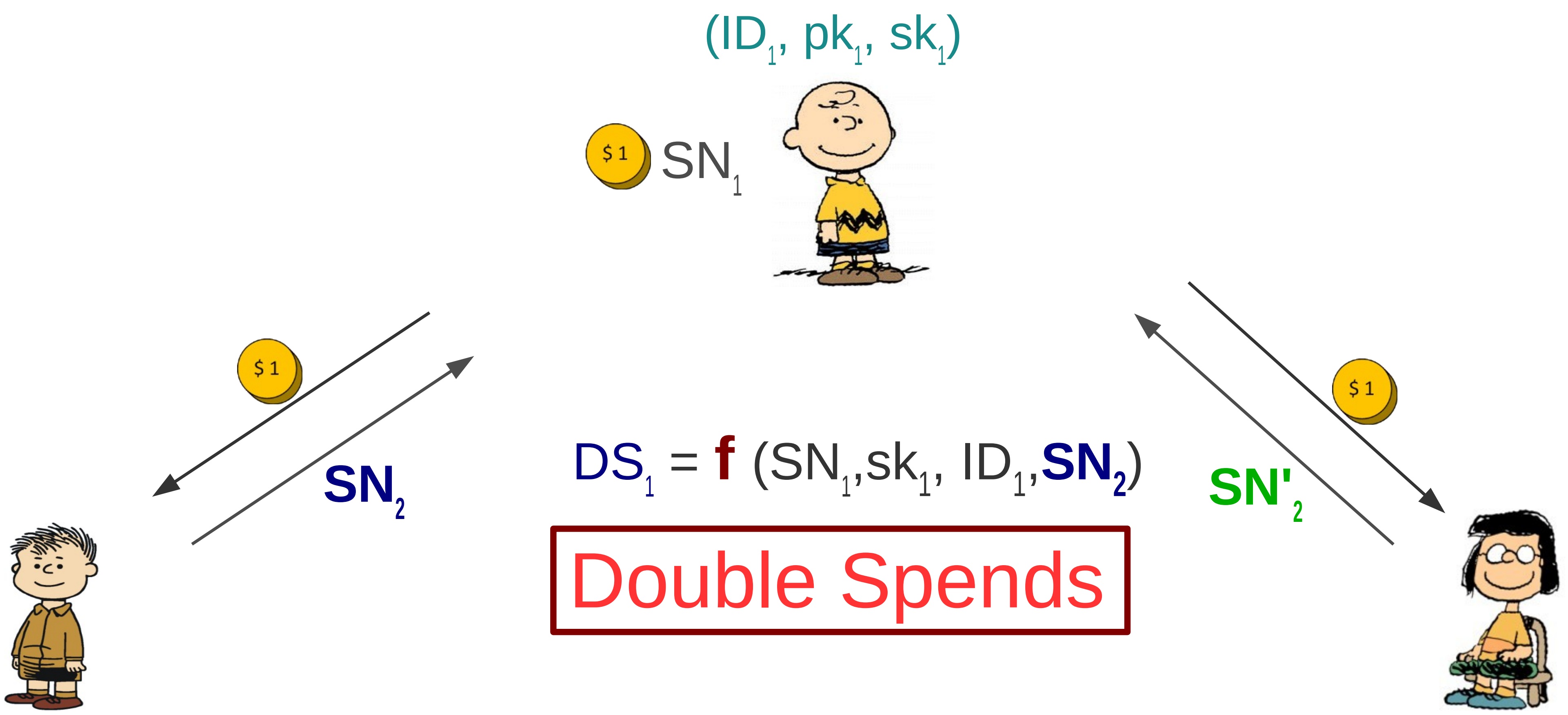
Double Spends



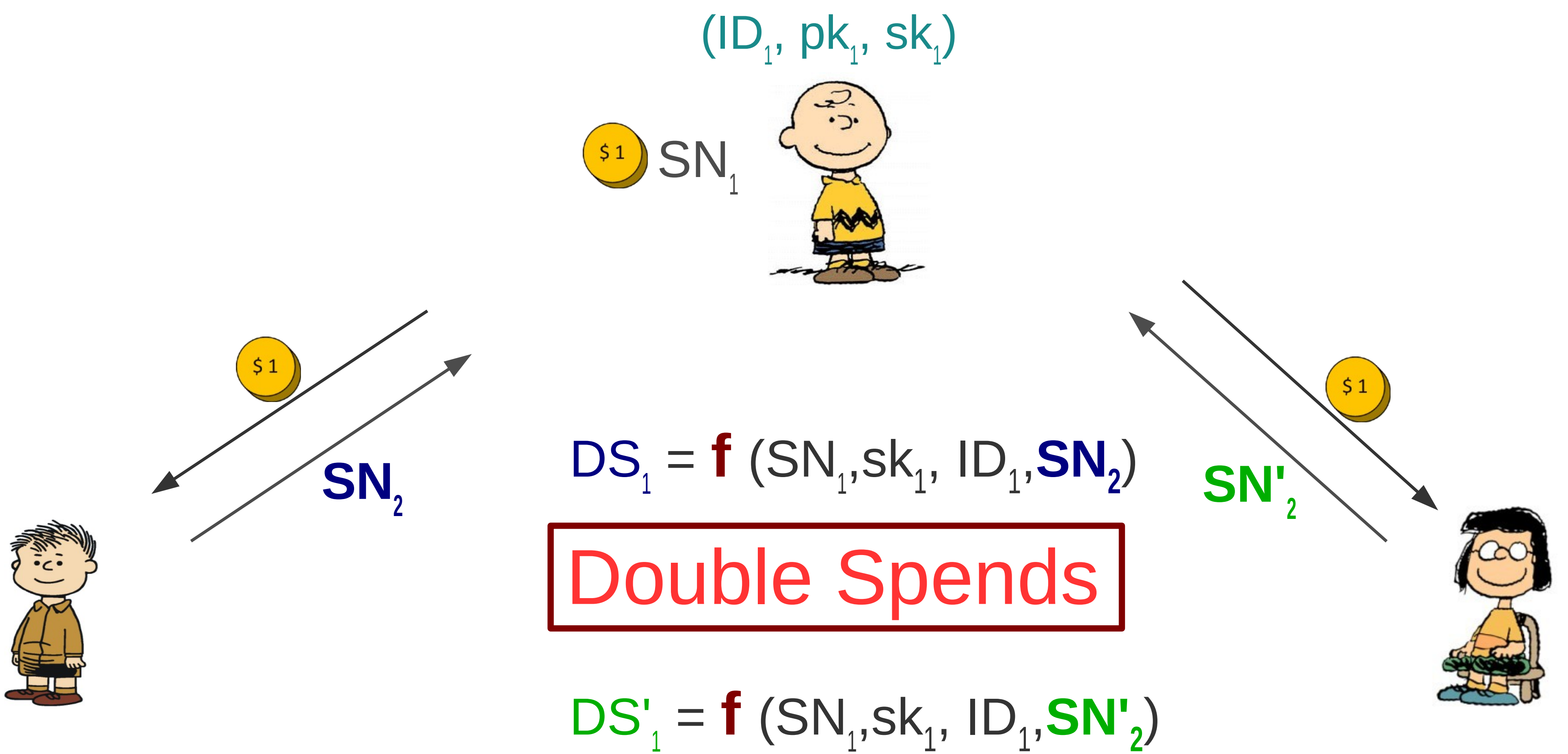
Double Spending Mechanism



Double Spending Mechanism



Double Spending Mechanism



Double Spending Mechanism

SN' = SN_1 || **SN_2 || … || SN'_k**

SN' = SN_1 || **SN'_2 || … || SN'_k**

Double Spending Mechanism

$\mathbf{SN'} = SN_1 \,||\, \mathbf{SN_2 \,||\, \ldots \,||\, SN'_k}$

$\mathbf{SN'} = SN_1 \,||\, \mathbf{SN'_2 \,||\, \ldots \,||\, SN'_k}$

$\mathbf{DS_1} = \mathbf{f}\,(SN_1, sk_1, ID_1, \mathbf{SN_2})$

$\mathbf{ID_1}$

$\mathbf{DS'_1} = \mathbf{f}\,(SN_1, sk_1, ID_1, \mathbf{SN'_2})$

Double Spending Mechanism

$\mathbf{SN'} = SN_1 \;||\; \mathbf{SN_2 \;||\; \ldots \;||\; SN'_k}$

$\mathbf{SN'} = SN_1 \;||\; \mathbf{SN'_2 \;||\; \ldots \;||\; SN'_k}$

$\mathbf{DS_1} = \mathbf{f}\,(SN_1, sk_1, ID_1, \mathbf{SN_2})$

$\mathbf{DS'_1} = \mathbf{f}\,(SN_1, sk_1, ID_1, \mathbf{SN'_2})$

Bank needs to check:

$\mathbf{x = y^{ID}}$

for every ID registered

Thm. "*Our DS mechanism is anonymous under DDH.*"

Constructing transferable e-cash

- ✔ Ensure that coins contain all the valid information in order for double spending detection to be successful and correct.
- ✔ Need to encode all the identities of the users who ever owned the coin in a way that ensures anonymity.

Constructing transferable e-cash

- ✔ Ensure that coins contain all the valid information in order for double spending detection to be successful and correct.

- ✔ Need to encode all the identities of the users who ever owned the coin in a way that ensures anonymity.

What is left?

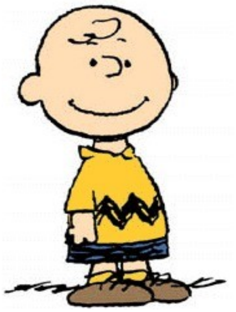
Make sure that coins are valid and unforgeable.

Malleable Signatures

sk,vk

$\sigma_{sk}(m)$

MSign



[ABCSsW'12, ALP'12,CKLM'13]

Malleable Signatures

sk,vk

$\sigma_{sk}(m)$

MSign

$m^* = T(m)$

Where T is an allowed transformation

[ABCSsW'12, ALP'12,CKLM'13]

Malleable Signatures

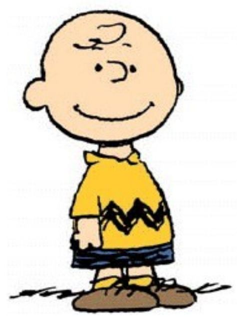
sk,vk

$\sigma_{sk}(m)$

MSign

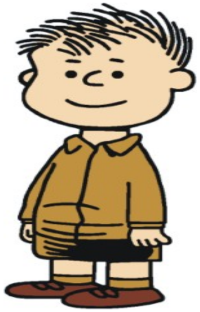
$m^* = T(m)$

Where T is an allowed transformation



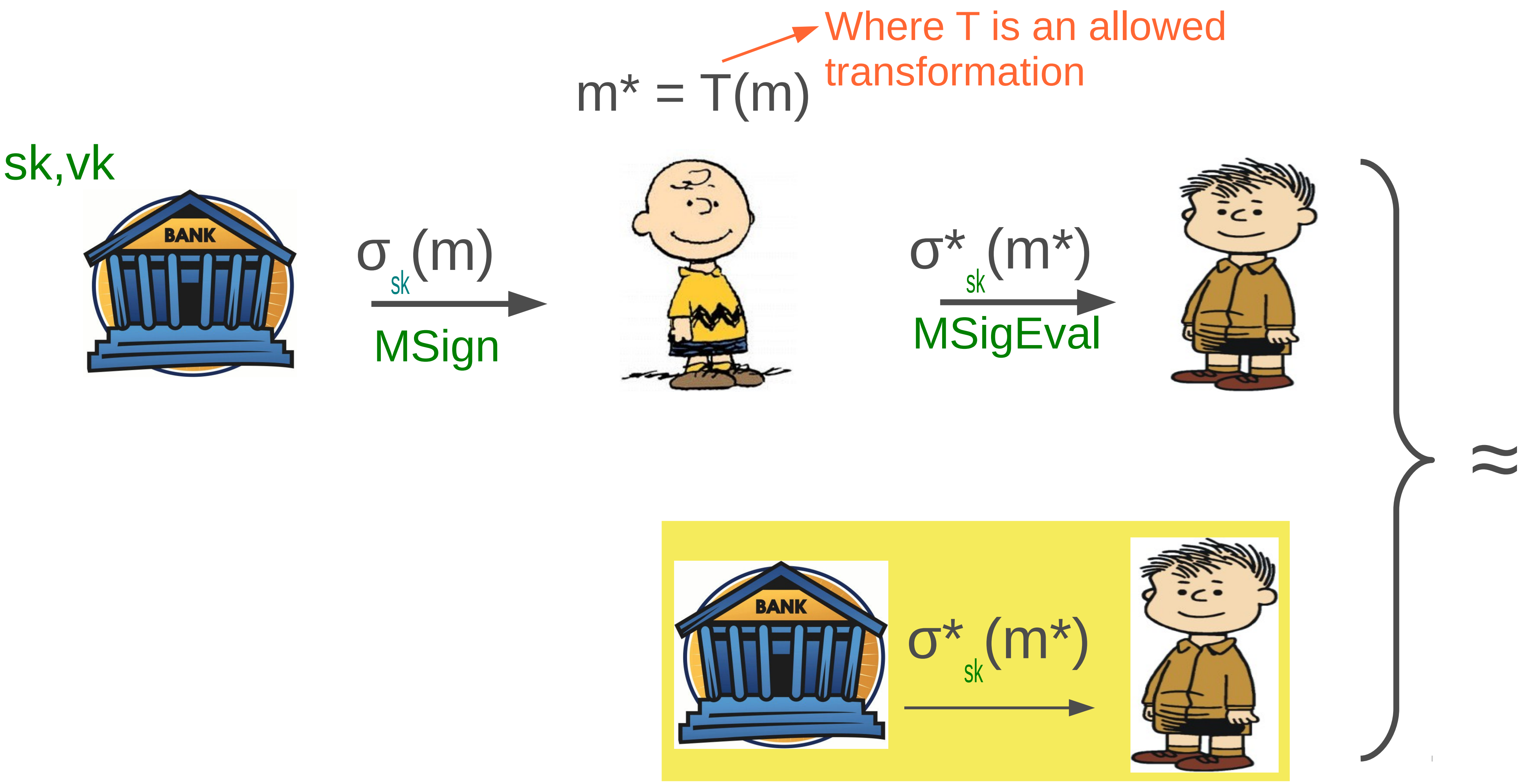
$\sigma^*_{sk}(m^*)$

MSigEval



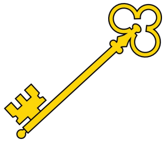
[ABCSsW'12, ALP'12,CKLM'13]

Malleable Signatures



[ABCSsW'12, ALP'12,CKLM'13]

Our Construction – Withdrawal

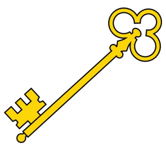
 

com ←

pick SN_1

com = Commit(SN_1)

Our Construction – Withdrawal



com

pick SN_1

σ = MSign(com)

com = Commit(SN_1)

σ

Our Construction – Withdrawal

σ = MSign(com)

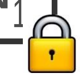
com ←

σ →

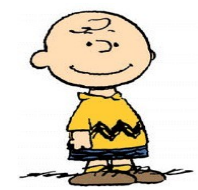
pick SN_1

com = Commit(SN_1)

σ^* = MSigEval(T, com, σ)

for T(com) = $\boxed{SN_1}$ 

Our Construction – Withdrawal



com

pick SN_1

com = Commit(SN_1)

σ = MSign(com)

σ

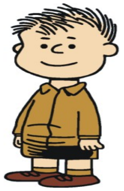
σ* = MSigEval(T, com, σ)

for T(com) = $\boxed{SN_1}$

 = ($\boxed{SN_1}$, σ*)

Our Construction - Spending

= (SN , DS , σ)



Our Construction - Spending

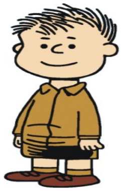
 = (SN, DS, σ)

SN= $SN_1 \,||\, SN_2 \,||\, \ldots \,||\, SN_j$

DS= $DS_1 \,||\, DS_2 \,||\, \ldots \,||\, DS_{j-1}$

Our Construction - Spending

 = (SN, DS, σ)

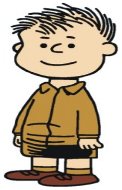


SN_{i+1} ←

pick SN_{i+1}

Our Construction - Spending

 = (SN, DS, σ)



compute DS_i

pick SN_{i+1}

Our Construction - Spending

= (SN, DS, σ)

compute DS_i

SN_{i+1} ←

pick SN_{i+1}

σ^* = MsigEval(T, SN, DS, σ)

Our Construction - Spending

 = (SN, DS, σ)

compute DS_i ← SN_{i+1} — pick SN_{i+1}

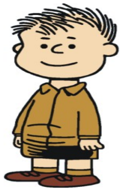
$\sigma^* =$ MsigEval(T, SN, DS, σ)

where T (SN, DS) =

($SN||SN_{j+1}$, $DS||DS_j$)

Our Construction - Spending

= (SN , DS , σ)



compute DS_i

SN_{i+1} ←

pick SN_{i+1}

σ* = MsigEval(T, SN , DS ,σ)

σ*, ($SN||SN_{j+1}$, $DS||DS_j$) →

where T (SN DS) =

($SN||SN_{j+1}$, $DS||DS_j$)

Our Construction - Spending

= (SN, DS, σ)

compute DS$_i$

SN$_{i+1}$ ←

pick SN$_{i+1}$

σ* = MsigEval(T, SN, DS, σ)

σ*, (SN||SN$_{j+1}$, DS||DS$_j$) →

where T (SN, DS) =

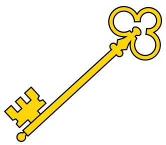
(SN||SN$_{j+1}$, DS||DS$_j$)

= (SN||SN$_{j+1}$, DS||DS$_j$, σ*)

Our Construction - Deposit

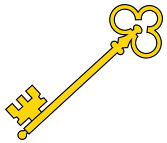
$\$1 = (\mathbf{SN}||SN_{j+1}, \mathbf{DS}||DS_j, \sigma')$

Run Spending



Our Construction - Deposit

$= (\mathbf{SN}||SN_{j+1}, \mathbf{DS}||DS_j, \sigma')$



Run Spending

Decrypt

$\mathbf{SN}=SN_1 \;||\; \dots \;||\; SN_{i+1}$

$\mathbf{DS}=DS_1 \;||\; \dots \;||\; DS_i$

If there exists a coin with same SN_1 then a double spent happened!

Our Construction - Security

We rely on the security properties of the underlying schemes:

1) Malleable signatures
2) Signature scheme
3) Commitment scheme
4) Randomizable encryption scheme

Exact assumptions depend on the instantiation!

Conclusion

The first practical, truly anonymous transferable e-cash scheme

- ✔ No trusted 3rd party that can de-anonymize users
- ✔ On double spending, only the identity of the malicious user is revealed

Possible instantiation:

Groth-Sahai proof system + El Gamal encryption/commitments + ACDKNO'12 structure preserving signatures

Secure under the Decision Linear (DLIN) and Symmetric External Decision Diffie-Hellman (SXDH) assumptions

thank you!

foteini@bu.edu

Additional Slides

Anonymity for transferable e-cash is more complicated [CG'08]...

- **Full anonymity (FA)**: an attacker, impersonating the bank, cannot recognize a coin he has already observed (observe-then-receive)
- **Perfect anonymity (PA):** an attacker cannot decide whether he has already owned a coin he is receiving (impossible)

[CP'92] An unbounded adversary will always recognize his own coins if he seems them later

What about a bounded adversary A, acting as the bank?