e-Cash

Lecture 26

Requirements

Requirements

- Involves a “Bank”, merchants and users

Requirements

- Involves a “Bank”, merchants and users
- Users have accounts in the Bank, with real money

Requirements

- Involves a “Bank”, merchants and users
- Users have accounts in the Bank, with real money
- Users should be able to withdraw e-cash and spend it later with any merchant; merchant can cash (deposit) the spent amount at the bank

Requirements

- Involves a “Bank”, merchants and users
- Users have accounts in the Bank, with real money
- Users should be able to withdraw e-cash and spend it later with any merchant; merchant can cash (deposit) the spent amount at the bank
- Even if the bank and merchant collude, they should not be able to link withdrawal with spending

Requirements

- Involves a "Bank", merchants and users
- Users have accounts in the Bank, with real money
- Users should be able to withdraw e-cash and spend it later with any merchant; merchant can cash (deposit) the spent amount at the bank
- Even if the bank and merchant collude, they should not be able to link withdrawal with spending
- Merchants/users (even colluding) should not be able to deposit e-cash that was not withdrawn

Requirements

- Involves a “Bank”, merchants and users
- Users have accounts in the Bank, with real money
- Users should be able to withdraw e-cash and spend it later with any merchant; merchant can cash (deposit) the spent amount at the bank
- Even if the bank and merchant collude, they should not be able to link withdrawal with spending
- Merchants/users (even colluding) should not be able to deposit e-cash that was not withdrawn
- Users should not be able to cheat honest merchants. In particular, users should not be able to double-spend

An approach

An approach

- Using "Blind Signatures"

An approach

- Using "Blind Signatures"
- User picks a serial number (coin), gets it signed blindly

An approach

- Using "Blind Signatures"
- User picks a serial number (coin), gets it signed blindly
- At a merchant's, the user gives the signed coin

An approach

- Using "Blind Signatures"
- User picks a serial number (coin), gets it signed blindly
- At a merchant's, the user gives the signed coin
- Merchant contacts the Bank (online) who ensures that the coin with that serial number has not been used before (i.e., no double spending) and the signature is valid. If so adds the coin to the spent-coin list

Blind Signatures

Blind Signatures

- A 2-party functionality between a User and a Signer

Blind Signatures

- A 2-party functionality between a User and a Signer
- Signer inputs a signing/verification key pair (SK,VK), User inputs a message m. User gets output (VK,Sign_{SK}(m)) (Signer gets nothing -- neither m, nor the signature).

Blind Signatures

- A 2-party functionality between a User and a Signer
- Signer inputs a signing/verification key pair (SK,VK), User inputs a message m. User gets output (VK,Sign_{SK}(m)) (Signer gets nothing -- neither m, nor the signature).
  - Signature is honestly generated (no "marked bills")

Blind Signatures

- A 2-party functionality between a User and a Signer
- Signer inputs a signing/verification key pair (SK,VK), User inputs a message m. User gets output (VK,Sign_{SK}(m)) (Signer gets nothing -- neither m, nor the signature).
    - Signature is honestly generated (no “marked bills”)
- Weaker security definition: blind, unlinkable and unforgeable

Blind Signatures

- A 2-party functionality between a User and a Signer
- Signer inputs a signing/verification key pair (SK,VK), User inputs a message m. User gets output (VK,$Sign_{SK}$(m)) (Signer gets nothing -- neither m, nor the signature).
  - Signature is honestly generated (no "marked bills")
- Weaker security definition: blind, unlinkable and unforgeable
  - Blindness: Signer cannot distinguish between m_0 and m_1

Blind Signatures

- A 2-party functionality between a User and a Signer
- Signer inputs a signing/verification key pair (SK,VK),
  User inputs a message m. User gets output (VK,Sign_{SK}(m))
  (Signer gets nothing -- neither m, nor the signature).
  - Signature is honestly generated (no "marked bills")
- Weaker security definition: blind, unlinkable and unforgeable
  - Blindness: Signer cannot distinguish between m_0 and m_1
  - Unlinkability: Signer cannot link a signature to the session in which it was created

Blind Signatures

- A 2-party functionality between a User and a Signer
- Signer inputs a signing/verification key pair (SK,VK),
  User inputs a message m. User gets output (VK,Sign_{SK}(m))
  (Signer gets nothing -- neither m, nor the signature).
  - Signature is honestly generated (no "marked bills")
- Weaker security definition: blind, unlinkable and unforgeable
  - Blindness: Signer cannot distinguish between m_0 and m_1
  - Unlinkability: Signer cannot link a signature to the session in which it was created
  - Unforgeability: After t sessions, User cannot output signatures on t+1 distinct messages

A Blind Signature Scheme

A Blind Signature Scheme

- In the Common Reference String model: CRS includes a PK for a CPA-secure PKE scheme and the CRS for a NIZK scheme

A Blind Signature Scheme

- In the Common Reference String model: CRS includes a PK for a CPA-secure PKE scheme and the CRS for a NIZK scheme
- Signing Protocol:

A Blind Signature Scheme

- In the Common Reference String model: CRS includes a PK for a CPA-secure PKE scheme and the CRS for a NIZK scheme
- Signing Protocol:
    - User → Signer: c := Commit(m) //Commit is perfectly binding

A Blind Signature Scheme

- In the Common Reference String model: CRS includes a PK for a CPA-secure PKE scheme and the CRS for a NIZK scheme
- Signing Protocol:
  - User → Signer: $c := \mathrm{Commit}(m)$ //Commit is perfectly binding
  - Signer → User: $\sigma := \mathrm{Sign}_{SK}(c)$

A Blind Signature Scheme

- In the Common Reference String model: CRS includes a PK for a CPA-secure PKE scheme and the CRS for a NIZK scheme
- Signing Protocol:
  - User → Signer: $c := \text{Commit}(m)$ //Commit is perfectly binding
  - Signer → User: $\sigma := \text{Sign}_{SK}(c)$
  - User: Output (C,π) as the signature on m, where $C = \text{Enc}(c, \sigma)$, and π is a NIZK of correctness of C

A Blind Signature Scheme

- In the Common Reference String model: CRS includes a PK for a CPA-secure PKE scheme and the CRS for a NIZK scheme
- Signing Protocol:
  - User → Signer: $c := \text{Commit}(m)$ //Commit is perfectly binding
  - Signer → User: $\sigma := \text{Sign}_{SK}(c)$
  - User: Output (C,π) as the signature on m, where $C = \text{Enc}(c, \sigma)$, and π is a NIZK of correctness of C
    - Correctness of C: there exists $c,\sigma,r_{PKE},r_{Commit}$ such that $c=\text{Commit}(m;r_{commit})$, $C=\text{Enc}_{PK}(c,\sigma;r_{PKE})$ and $\text{Verify}_{VK}(c,\sigma)$ holds

A Blind Signature Scheme

- In the Common Reference String model: CRS includes a PK for a CPA-secure PKE scheme and the CRS for a NIZK scheme
- Signing Protocol:
  - User → Signer: $c := \text{Commit}(m)$ //Commit is perfectly binding
  - Signer → User: $\sigma := \text{Sign}_{SK}(c)$
  - User: Output (C,π) as the signature on m, where $C = \text{Enc}(c, \sigma)$, and π is a NIZK of correctness of C
    - Correctness of C: there exists $c,\sigma,r_{PKE},r_{Commit}$ such that $c=\text{Commit}(m;r_{commit})$, $C=\text{Enc}_{PK}(c,\sigma;r_{PKE})$ and $\text{Verify}_{VK}(c,\sigma)$ holds
- Blindness, because signer sees only Commit(m). Unlinkability from encryption. Unforgeability from soundness of NIZK, efficient decryption of PKE, and unforgeability of the signature scheme

A Blind Signature Scheme

- In the Common Reference String model: CRS includes a PK for a CPA-secure PKE scheme and the CRS for a NIZK scheme
- Signing Protocol:
  - User → Signer: $c := \mathrm{Commit}(m)$ //Commit is perfectly binding
  - Signer → User: $\sigma := \mathrm{Sign}_{SK}(c)$
  - User: Output (C,π) as the signature on m, where $C = \mathrm{Enc}(c, \sigma)$, and π is a NIZK of correctness of C
    - Correctness of C: there exists $c,\sigma,r_{PKE},r_{Commit}$ such that $c=\mathrm{Commit}(m;r_{commit})$, $C=\mathrm{Enc}_{PK}(c,\sigma;r_{PKE})$ and $\mathrm{Verify}_{VK}(c,\sigma)$ holds
- Blindness, because signer sees only Commit(m). Unlinkability from encryption. Unforgeability from soundness of NIZK, efficient decryption of PKE, and unforgeability of the signature scheme
- Efficient variants (under suitable assumptions) using Groth-Sahai NIZK (or NIWI) scheme and compatible primitives

Offline e-Cash

Offline e-Cash

- Previous scheme requires the merchant to contact the Bank online

Offline e-Cash

- Previous scheme requires the merchant to contact the Bank online
- Indeed, merchants can't detect/prevent double spending without contacting the Bank since they do not interact with each other

Offline e-Cash

- Previous scheme requires the merchant to contact the Bank online
- Indeed, merchants can't detect/prevent double spending without contacting the Bank since they do not interact with each other
  - (Unless hardware tokens are used)

Offline e-Cash

- Previous scheme requires the merchant to contact the Bank online
- Indeed, merchants can't detect/prevent double spending without contacting the Bank since they do not interact with each other
  - (Unless hardware tokens are used)
- Detecting double-spending only later is not enough

Offline e-Cash

- Previous scheme requires the merchant to contact the Bank online
- Indeed, merchants can't detect/prevent double spending without contacting the Bank since they do not interact with each other
  - (Unless hardware tokens are used)
- Detecting double-spending only later is not enough
- In offline e-Cash, double spending is allowed, but will be caught and <u>traced</u> to the user when a merchant deposits the coin

Offline e-Cash

- Previous scheme requires the merchant to contact the Bank online
- Indeed, merchants can't detect/prevent double spending without contacting the Bank since they do not interact with each other
  - (Unless hardware tokens are used)
- Detecting double-spending only later is not enough
- In offline e-Cash, double spending is allowed, but will be caught and traced to the user when a merchant deposits the coin
  - Idea: verification in two sessions of the spending protocol with the same coin exposes the user's identity

Offline e-Cash: A plan

Offline e-Cash: A plan

- Coin must contain information about the user's identity

Offline e-Cash: A plan

- Coin must contain information about the user's identity
- Withdrawal: get a blind signature from the Bank on (ID,s,t) where s is a serial number and t used in keeping the ID secret while spending (for up to one time). (s,t from a suitable field)

Offline e-Cash: A plan

- Coin must contain information about the user's identity
- Withdrawal: get a blind signature from the Bank on (ID,s,t) where s is a serial number and t used in keeping the ID secret while spending (for up to one time). (s,t from a suitable field)
    - Must first convince the Bank that message being signed has the correct ID (to prevent implication of a wrong user on double spending): partially blind signatures

Offline e-Cash: A plan

- Coin must contain information about the user's identity
- Withdrawal: get a blind signature from the Bank on (ID,s,t) where s is a serial number and t used in keeping the ID secret while spending (for up to one time). (s,t from a suitable field)
  - Must first convince the Bank that message being signed has the correct ID (to prevent implication of a wrong user on double spending): partially blind signatures
- Spending: reveal (s,d) where d := ID+Rt, for a random challenge R from the merchant, along with a PoK of signature on (ID',s,t') for some ID',t' s.t. ID'+Rt' = d

Offline e-Cash: A plan

- Coin must contain information about the user's identity
- Withdrawal: get a blind signature from the Bank on (ID,s,t) where s is a serial number and t used in keeping the ID secret while spending (for up to one time). (s,t from a suitable field)
  - Must first convince the Bank that message being signed has the correct ID (to prevent implication of a wrong user on double spending): partially blind signatures
- Spending: reveal (s,d) where d := ID+Rt, for a random challenge R from the merchant, along with a PoK of signature on (ID',s,t') for some ID',t' s.t. ID'+Rt' = d
  - On depositing the same coin twice, Bank can solve for ID

Offline e-Cash: A plan

- Coin must contain information about the user's identity
- Withdrawal: get a blind signature from the Bank on (ID,s,t) where s is a serial number and t used in keeping the ID secret while spending (for up to one time). (s,t from a suitable field)
  - Must first convince the Bank that message being signed has the correct ID (to prevent implication of a wrong user on double spending): partially blind signatures
- Spending: reveal (s,d) where d := ID+Rt, for a random challenge R from the merchant, along with a PoK of signature on (ID′,s,t′) for some ID′,t′ s.t. ID′+Rt′ = d
  - On depositing the same coin twice, Bank can solve for ID
  - Merchant needs to transfer the User's proof to Bank (i.e., Bank should be convinced that the merchant didn't fake)

Signatures with Proofs: CL Signatures

Signatures with Proofs: CL Signatures

- Camenisch-Lysyanskaya signatures: Uses Pedersen commitments; security under DDH and Strong RSA assumptions

Signatures with Proofs: CL Signatures

- Camenisch-Lysyanskaya signatures: Uses Pedersen commitments; security under DDH and Strong RSA assumptions
- Blind signature functionality:

Signatures with Proofs: CL Signatures

- Camenisch-Lysyanskaya signatures: Uses Pedersen commitments; security under DDH and Strong RSA assumptions
- Blind signature functionality:
  - Common input: Pedersen commitment to a vector $(x_1,\ldots,x_n)$ $Com(x_1,..,x_n;r) = g_1^{x1}\ldots g_n^{xn}\, h^r$ and a verification key VK

Signatures with Proofs: CL Signatures

- Camenisch-Lysyanskaya signatures: Uses Pedersen commitments; security under DDH and Strong RSA assumptions
- Blind signature functionality:
  - Common input: Pedersen commitment to a vector $(x_1,\ldots,x_n)$ $\mathrm{Com}(x_1,\ldots,x_n;r) = g_1^{x1}\ldots g_n^{xn}\, h^r$ and a verification key VK
  - User's input: $x_1,\ldots,x_n$ and r; Signer's input: signing key SK

Signatures with Proofs: CL Signatures

- Camenisch-Lysyanskaya signatures: Uses Pedersen commitments; security under DDH and Strong RSA assumptions
- Blind signature functionality:
  - Common input: Pedersen commitment to a vector $(x_1,\ldots,x_n)$ $\mathrm{Com}(x_1,\ldots,x_n;r) = g_1^{x1}\ldots g_n^{xn}\, h^r$ and a verification key VK
  - User's input: $x_1,\ldots,x_n$ and r; Signer's input: signing key SK
  - User's output: $\mathrm{Sign}_{SK}(x_1,\ldots,x_n)$ (i.e., sign on the message itself)

Signatures with Proofs: CL Signatures

- Camenisch-Lysyanskaya signatures: Uses Pedersen commitments; security under DDH and Strong RSA assumptions
- Blind signature functionality:
  - Common input: Pedersen commitment to a vector $(x_1,\ldots,x_n)$ $\mathrm{Com}(x_1,..,x_n;r) = g_1^{x1}\ldots g_n^{xn}\, h^r$ and a verification key VK
  - User's input: $x_1,\ldots,x_n$ and r; Signer's input: signing key SK
  - User's output: $\mathrm{Sign}_{SK}(x_1,\ldots,x_n)$ (i.e., sign on the message itself)
- Proof functionality:

Signatures with Proofs: CL Signatures

- Camenisch-Lysyanskaya signatures: Uses Pedersen commitments; security under DDH and Strong RSA assumptions
- Blind signature functionality:
  - Common input: Pedersen commitment to a vector $(x_1,\ldots,x_n)$ $\mathrm{Com}(x_1,\ldots,x_n;r) = g_1^{x_1}\ldots g_n^{x_n} h^r$ and a verification key VK
  - User's input: $x_1,\ldots,x_n$ and r; Signer's input: signing key SK
  - User's output: $\mathrm{Sign}_{SK}(x_1,\ldots,x_n)$ (i.e., sign on the message itself)
- Proof functionality:
  - Common input: VK and $\mathrm{Com}(x_1,\ldots,x_n;r')$

Signatures with Proofs: CL Signatures

- Camenisch-Lysyanskaya signatures: Uses Pedersen commitments; security under DDH and Strong RSA assumptions
- Blind signature functionality:
  - Common input: Pedersen commitment to a vector $(x_1,\ldots,x_n)$ $\mathrm{Com}(x_1,..,x_n;r) = g_1^{x1}\ldots g_n^{xn}\,h^r$ and a verification key VK
  - User's input: $x_1,\ldots,x_n$ and r; Signer's input: signing key SK
  - User's output: $\mathrm{Sign}_{SK}(x_1,\ldots,x_n)$ (i.e., sign on the message itself)
- Proof functionality:
  - Common input: VK and $\mathrm{Com}(x_1,..,x_n;r')$
  - User's input: $(x_1,\ldots,x_n,r')$ and a signature on $(x_1,\ldots,x_n)$

Signatures with Proofs: CL Signatures

- Camenisch-Lysyanskaya signatures: Uses Pedersen commitments; security under DDH and Strong RSA assumptions
- Blind signature functionality:
  - Common input: Pedersen commitment to a vector $(x_1,\ldots,x_n)$ $\mathrm{Com}(x_1,..,x_n;r) = g_1^{x1}\ldots g_n^{xn}\, h^r$ and a verification key VK
  - User's input: $x_1,\ldots,x_n$ and r; Signer's input: signing key SK
  - User's output: $\mathrm{Sign}_{SK}(x_1,\ldots,x_n)$ (i.e., sign on the message itself)
- Proof functionality:
  - Common input: VK and $\mathrm{Com}(x_1,..,x_n;r')$
  - User's input: $(x_1,\ldots,x_n,r')$ and a signature on $(x_1,\ldots,x_n)$
  - Verifier gets verification that signature and commitment are valid and on same message

Signatures with Proofs: CL Signatures

- Camenisch-Lysyanskaya signatures: Uses Pedersen commitments; security under DDH and Strong RSA assumptions
- Blind signature functionality:
  - Common input: Pedersen commitment to a vector $(x_1,\ldots,x_n)$ $\mathrm{Com}(x_1,..,x_n;r) = g_1^{x_1}\ldots g_n^{x_n} h^r$ and a verification key VK
  - User's input: $x_1,\ldots,x_n$ and r; Signer's input: signing key SK
  - User's output: $\mathrm{Sign}_{SK}(x_1,\ldots,x_n)$ (i.e., sign on the message itself)
- Proof functionality:
  - Common input: VK and $\mathrm{Com}(x_1,..,x_n;r')$
  - User's input: $(x_1,\ldots,x_n,r')$ and a signature on $(x_1,\ldots,x_n)$
  - Verifier gets verification that signature and commitment are valid and on same message
- Verification is interactive (but can be made transferable using Fiat-Shamir heuristics in the RO model)

Signatures with Proofs: P-Signatures

Signatures with Proofs: P-Signatures

- Like CL Signatures, but with non-interactive proofs

Signatures with Proofs: P-Signatures

- Like CL Signatures, but with <u>non-interactive</u> proofs
  - \- Blind Signature; signer takes a commitment to message
    \- Proof of Knowledge of signature on a value
    \- Proof of equivalence of two committed values

Signatures with Proofs: P-Signatures

- Like CL Signatures, but with non-interactive proofs
  - \- Blind Signature; signer takes a commitment to message
    \- Proof of Knowledge of signature on a value
    \- Proof of equivalence of two committed values
  - Setup involves a (trusted) CRS

Signatures with Proofs: P-Signatures

- Like CL Signatures, but with non-interactive proofs
  - \- Blind Signature; signer takes a commitment to message
    \- Proof of Knowledge of signature on a value
    \- Proof of equivalence of two committed values
  - Setup involves a (trusted) CRS
- Constructions known in groups with bilinear pairings

Signatures with Proofs: P-Signatures

- Like CL Signatures, but with <u>non-interactive</u> proofs
  - \- Blind Signature; signer takes a commitment to message
    \- Proof of Knowledge of signature on a value
    \- Proof of equivalence of two committed values
  - Setup involves a (trusted) CRS
- Constructions known in groups with bilinear pairings
  - Proofs using Groth-Sahai NIZK/NIWI schemes

Signatures with Proofs: P-Signatures

- Like CL Signatures, but with non-interactive proofs
  - - Blind Signature; signer takes a commitment to message
    - Proof of Knowledge of signature on a value
    - Proof of equivalence of two committed values
  - Setup involves a (trusted) CRS
- Constructions known in groups with bilinear pairings
  - Proofs using Groth-Sahai NIZK/NIWI schemes
  - Uses signatures and commitments s.t. the statements to be proven are covered by GS NIZKs

Signatures with Proofs: P-Signatures

- Like CL Signatures, but with <u>non-interactive</u> proofs
  - - Blind Signature; signer takes a commitment to message
    - Proof of Knowledge of signature on a value
    - Proof of equivalence of two committed values
  - Setup involves a (trusted) CRS
- Constructions known in groups with bilinear pairings
  - Proofs using Groth-Sahai NIZK/NIWI schemes
  - Uses signatures and commitments s.t. the statements to be proven are covered by GS NIZKs
    - e.g. (Weak) Boneh-Boyen signature: $\text{Sign}_{SK}(x) = g^{1/(SK+x)}$

Efficiency Issues

Efficiency Issues

- So far, withdrawal involves one signature per coin

Efficiency Issues

- So far, withdrawal involves one signature per coin
- Use large denominations?

Efficiency Issues

- So far, withdrawal involves one signature per coin
- Use large denominations?
  - Should allow spending in small denominations

Efficiency Issues

- So far, withdrawal involves one signature per coin
- Use large denominations?
  - Should allow spending in small denominations
  - Divisible e-cash

Efficiency Issues

- So far, withdrawal involves one signature per coin
- Use large denominations?
  - Should allow spending in small denominations
  - Divisible e-cash
    - Should allow spending multiple times from the same large denomination coin. But to detect over-spending, allows linking together spendings from the same coin

Efficiency Issues

- So far, withdrawal involves one signature per coin
- Use large denominations?
  - Should allow spending in small denominations
  - Divisible e-cash
    - Should allow spending multiple times from the same large denomination coin. But to detect over-spending, allows linking together spendings from the same coin
    - Trees with small denomination coins at the leaves; can spend any node (root of a subtree); spending a node and a descendent will reveal ID

Efficiency Issues

- So far, withdrawal involves one signature per coin
- Use large denominations?
  - Should allow spending in small denominations
  - Divisible e-cash
    - Should allow spending multiple times from the same large denomination coin. But to detect over-spending, allows linking together spendings from the same coin
    - Trees with small denomination coins at the leaves; can spend any node (root of a subtree); spending a node and a descendent will reveal ID
  - Compact e-Cash: Remove linking multiple spending

Compact e-Cash

Compact e-Cash

- Recall previous (non-compact) scheme: get signature on (ID,s,t) during withdrawal and reveal (s,d) where d := ID+Rt for a challenge R, when spending the coin

Compact e-Cash

- Recall previous (non-compact) scheme: get signature on (ID,s,t) during withdrawal and reveal (s,d) where d := ID+Rt for a challenge R, when spending the coin
  - Instead, let s, t be seeds of a PRF

Compact e-Cash

- Recall previous (non-compact) scheme: get signature on (ID,s,t) during withdrawal and reveal (s,d) where d := ID+Rt for a challenge R, when spending the coin
  - Instead, let s, t be seeds of a PRF
  - On spending for the i^{th} time, reveal (S,D) where $S = PRF_s(i)$ and $D = ID + R\,T$, where $T = PRF_t(i)$

Compact e-Cash

- Recall previous (non-compact) scheme: get signature on (ID,s,t) during withdrawal and reveal (s,d) where d := ID+Rt for a challenge R, when spending the coin
  - Instead, let s, t be seeds of a PRF
  - On spending for the i^{th} time, reveal (S,D) where $S = PRF_s(i)$ and $D = ID + R\,T$, where $T = PRF_t(i)$
    - Prove that ID,s,t,i,signature exist as claimed. Optionally, that i is in the range [1,L] for some upper-bound L

Compact e-Cash

- Recall previous (non-compact) scheme: get signature on (ID,s,t) during withdrawal and reveal (s,d) where d := ID+Rt for a challenge R, when spending the coin
  - Instead, let s, t be seeds of a PRF
  - On spending for the i^{th} time, reveal (S,D) where $S = PRF_s(i)$ and $D = ID + R\,T$, where $T = PRF_t(i)$
    - Prove that ID,s,t,i,signature exist as claimed. Optionally, that i is in the range [1,L] for some upper-bound L
    - s secret, so can't link multiple spendings of the same coin

Compact e-Cash

- Recall previous (non-compact) scheme: get signature on (ID,s,t) during withdrawal and reveal (s,d) where d := ID+Rt for a challenge R, when spending the coin
  - Instead, let s, t be seeds of a PRF
  - On spending for the i^{th} time, reveal (S,D) where $S = PRF_s(i)$ and $D = ID + R\,T$, where $T = PRF_t(i)$
    - Prove that ID,s,t,i,signature exist as claimed. Optionally, that i is in the range [1,L] for some upper-bound L
    - s secret, so can't link multiple spendings of the same coin
    - Spending is still one coin at a time

Compact e-Cash

- Recall previous (non-compact) scheme: get signature on (ID,s,t) during withdrawal and reveal (s,d) where d := ID+Rt for a challenge R, when spending the coin
  - Instead, let s, t be seeds of a PRF
  - On spending for the i^{th} time, reveal (S,D) where $S = PRF_s(i)$ and $D = ID + R\,T$, where $T = PRF_t(i)$
    - Prove that ID,s,t,i,signature exist as claimed. Optionally, that i is in the range [1,L] for some upper-bound L
    - s secret, so can't link multiple spendings of the same coin
    - Spending is still one coin at a time
  - Need a PRF that supports efficient proofs

A PRF for compact e-Cash

A PRF for compact e-Cash

- $F_{g,s}(x) = g^{1/(s+x+1)}$ where s is the seed (g can be public) [DY05]

A PRF for compact e-Cash

- $F_{g,s}(x) = g^{1/(s+x+1)}$ where s is the seed (g can be public) [DY05]
- Secure under q-DDH Inversion (DDHI) Assumption

A PRF for compact e-Cash

- $F_{g,s}(x) = g^{1/(s+x+1)}$ where s is the seed (g can be public) [DY05]
- Secure under q-DDH Inversion (DDHI) Assumption
  - Given $(g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^q})$ for random g and x, $g^{1/x}$ is pseudorandom (i.e., indistinguishable from g^r)

A PRF for compact e-Cash

- $F_{g,s}(x) = g^{1/(s+x+1)}$ where s is the seed (g can be public) [DY05]
- Secure under q-DDH Inversion (DDHI) Assumption
  - Given $(g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^q})$ for random g and x, $g^{1/x}$ is pseudorandom (i.e., indistinguishable from g^r)
  - cf. q-SDH: hard to find $(y, g^{1/x+y})$

A PRF for compact e-Cash

- $F_{g,s}(x) = g^{1/(s+x+1)}$ where s is the seed (g can be public) [DY05]
- Secure under q-DDH Inversion (DDHI) Assumption
  - Given $(g, g^{x}, g^{x^2}, g^{x^3}, \ldots, g^{x^q})$ for random g and x, $g^{1/x}$ is pseudorandom (i.e., indistinguishable from g^r)
  - cf. q-SDH: hard to find $(y, g^{1/x+y})$
- Efficient (but interactive) HVZK proofs known for requisite statements. Used to get compact e-cash in the Random Oracle Model [CHL06]

A PRF for compact e-Cash

- $F_{g,s}(x) = g^{1/(s+x+1)}$ where s is the seed (g can be public) [DY05]
- Secure under q-DDH Inversion (DDHI) Assumption
  - Given $(g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^q})$ for random g and x, $g^{1/x}$ is pseudorandom (i.e., indistinguishable from g^r)
  - cf. q-SDH: hard to find $(y, g^{1/x+y})$
- Efficient (but interactive) HVZK proofs known for requisite statements. Used to get compact e-cash in the Random Oracle Model [CHL06]
- Alternately, working in groups with bilinear pairings, can use Groth-Sahai NIZK (under appropriate assumptions)

e-Cash today

e-Cash today

- Originally proposed by Chaum in 1982

e-Cash today

- Originally proposed by Chaum in 1982
- Not commercially deployed

e-Cash today

- Originally proposed by Chaum in 1982
- Not commercially deployed
  - Some attempts in mid 90's failed commercially

e-Cash today

- Originally proposed by Chaum in 1982
- Not commercially deployed
  - Some attempts in mid 90's failed commercially
  - Requires investment from financial institutions, merchants and bankers

e-Cash today

- Originally proposed by Chaum in 1982
- Not commercially deployed
  - Some attempts in mid 90's failed commercially
  - Requires investment from financial institutions, merchants and bankers
  - Non-anonymous electronic payment methods (credit-cards, pay-pal etc.) are still widely trusted

e-Cash today

- Originally proposed by Chaum in 1982
- Not commercially deployed
  - Some attempts in mid 90's failed commercially
  - Requires investment from financial institutions, merchants and bankers
  - Non-anonymous electronic payment methods (credit-cards, pay-pal etc.) are still widely trusted
- Active research continues

e-Cash today

- Originally proposed by Chaum in 1982
- Not commercially deployed
  - Some attempts in mid 90's failed commercially
  - Requires investment from financial institutions, merchants and bankers
  - Non-anonymous electronic payment methods (credit-cards, pay-pal etc.) are still widely trusted
- Active research continues
  - e.g. schemes not depending on Random Oracles, but on relatively untested assumptions

e-Cash today

- Originally proposed by Chaum in 1982
- Not commercially deployed
  - Some attempts in mid 90's failed commercially
  - Requires investment from financial institutions, merchants and bankers
  - Non-anonymous electronic payment methods (credit-cards, pay-pal etc.) are still widely trusted
- Active research continues
  - e.g. schemes not depending on Random Oracles, but on relatively untested assumptions
- Security/Efficiency/Usability issues: need to cancel stolen electronic wallet; need to recharge electronic wallet (cellphone?) online, but protect it from malware; efficient multiple denomination coins; allow transferability; tracing may not deter double-spending

Anonymous Credentials

Anonymous Credentials

- Introduced by Chaum in 1985

Anonymous Credentials

- Introduced by Chaum in 1985
- Similar to e-cash, but must allow multiple uses (double-spending not an issue)

Anonymous Credentials

- Introduced by Chaum in 1985
- Similar to e-cash, but must allow multiple uses (double-spending not an issue)
- Alice should be able to prove to Bob that she has a credential from Carol (cf. Alice withdraws a coin from Carol and spends it with Bob)

Anonymous Credentials

- Introduced by Chaum in 1985
- Similar to e-cash, but must allow multiple uses (double-spending not an issue)
- Alice should be able to prove to Bob that she has a credential from Carol (cf. Alice withdraws a coin from Carol and spends it with Bob)
  - Bob and Carol cannot link the persons who proved credentials to the persons who obtained credentials

Anonymous Credentials

- Introduced by Chaum in 1985
- Similar to e-cash, but must allow multiple uses (double-spending not an issue)
- Alice should be able to prove to Bob that she has a credential from Carol (cf. Alice withdraws a coin from Carol and spends it with Bob)
  - Bob and Carol cannot link the persons who proved credentials to the persons who obtained credentials
  - And they cannot link together multiple proofs coming from the same user

Anonymous Credentials from P-Signatures

Anonymous Credentials from P-Signatures

- User Alice has a public-key, PK_A and a secret key SK_A

Anonymous Credentials from P-Signatures

- User Alice has a public-key, PK_A and a secret key SK_A
- Alice needs pseudonyms with Bob and Carol, say A_B and A_C

Anonymous Credentials from P-Signatures

- User Alice has a public-key, PK_A and a secret key SK_A
- Alice needs pseudonyms with Bob and Carol, say A_B and A_C
  - A_B and A_C will be (independent) commitments to SK_A (using the commitment supported by the P-Signature)

Anonymous Credentials from P-Signatures

- User Alice has a public-key, PK_A and a secret key SK_A
- Alice needs pseudonyms with Bob and Carol, say A_B and A_C
  - A_B and A_C will be (independent) commitments to SK_A (using the commitment supported by the P-Signature)
- Obtaining credential: Carol signs SK_A using the P-Signature scheme using A_C (without learning SK_A). If Carol is a root authority, she requires a proof that A_C is a valid commitment of SK_A that corresponds to PK_A (not anonymous). Else Carol verifies that A_C has a credential from the root authority (as below)

Anonymous Credentials from P-Signatures

- User Alice has a public-key, PK_A and a secret key SK_A
- Alice needs pseudonyms with Bob and Carol, say A_B and A_C
  - A_B and A_C will be (independent) commitments to SK_A (using the commitment supported by the P-Signature)
- <u>Obtaining credential</u>: Carol signs SK_A using the P-Signature scheme using A_C (without learning SK_A). If Carol is a root authority, she requires a proof that A_C is a valid commitment of SK_A that corresponds to PK_A (not anonymous). Else Carol verifies that A_C has a credential from the root authority (as below)
- <u>Proving</u>: Alice wants to prove to Carol that owner of A_C has a credential from Bob. She commits SK_A again to get A' and shows that she has a signature from Bob on the message in A'. She also proves that A' and A_C have the same message

Today

Today

- e-Cash

Today

- e-Cash
  - Anonymous, offline validation and compact

Today

- e-Cash
    - Anonymous, offline validation and compact
- Relies on signatures, PRFs and NIZK

Today

- e-Cash
  - Anonymous, offline validation and compact
- Relies on signatures, PRFs and NIZK
  - Signatures with associated protocols (P-signatures, CL signatures, (partially) Blind signatures)

Today

- e-Cash
  - Anonymous, offline validation and compact
- Relies on signatures, PRFs and NIZK
  - Signatures with associated protocols (P-signatures, CL signatures, (partially) Blind signatures)
  - Efficient schemes using appropriate signatures that allow efficient NIZK schemes (e.g. Groth-Sahai)

Today

- e-Cash
  - Anonymous, offline validation and compact
- Relies on signatures, PRFs and NIZK
  - Signatures with associated protocols (P-signatures, CL signatures, (partially) Blind signatures)
  - Efficient schemes using appropriate signatures that allow efficient NIZK schemes (e.g. Groth-Sahai)
- Anonymous credentials