Functionally Rich Signatures

Lecture 24

Functionally Rich Signatures

Functionally Rich Signatures

- Signatures with various functionality/properties

Functionally Rich Signatures

- Signatures with various functionality/properties
- Constructions come in different flavors (we'll sample each flavor):

Functionally Rich Signatures

- Signatures with various functionality/properties
- Constructions come in different flavors (we'll sample each flavor):
  - Simple and efficient ones in the Random Oracle Model

Functionally Rich Signatures

- Signatures with various functionality/properties
- Constructions come in different flavors (we'll sample each flavor):
  - Simple and efficient ones in the Random Oracle Model
  - Relatively efficient ones under specific assumptions (often relatively strong/new assumptions)

Functionally Rich Signatures

- Signatures with various functionality/properties
- Constructions come in different flavors (we'll sample each flavor):
  - Simple and efficient ones in the Random Oracle Model
  - Relatively efficient ones under specific assumptions (often relatively strong/new assumptions)
  - Using minimal/general assumptions, often simple, but not very efficient (e.g., involving NIZK for general NP statements)

Functionally Rich Signatures

- Signatures with various functionality/properties
- Constructions come in different flavors (we'll sample each flavor):
  - Simple and efficient ones in the Random Oracle Model
  - Relatively efficient ones under specific assumptions (often relatively strong/new assumptions)
  - Using minimal/general assumptions, often simple, but not very efficient (e.g., involving NIZK for general NP statements)
- Definitions sometimes have subtleties (not all of them have ideal functionality specifications)

Multi-Signatures

Multi-Signatures

- Multiple signers signing the same message

Multi-Signatures

- Multiple signers signing the same message
  - Each signer has an (SK,VK) pair

Multi-Signatures

- Multiple signers signing the same message
  - Each signer has an (SK,VK) pair
- Resulting signature must be "compact": size independent of the number of signers

Multi-Signatures

- Multiple signers signing the same message
  - Each signer has an (SK,VK) pair
- Resulting signature must be "compact": size independent of the number of signers
- Security requirement: Unforgeability (chosen message security)

Multi-Signatures

- Multiple signers signing the same message
  - Each signer has an (SK,VK) pair
- Resulting signature must be “compact”: size independent of the number of signers
- Security requirement: Unforgeability (chosen message security)
  - Adversary can collude with all but one signer

Schnorr Signature

Schnorr Signature

- A regular (i.e., non-multi) digital signature scheme secure in the Random Oracle model under the discrete log assumption

Schnorr Signature

- A regular (i.e., non-multi) digital signature scheme secure in the Random Oracle model under the discrete log assumption
  - KeyGen: Signing key is x and Verification key is $X = g^x$

Schnorr Signature

- A regular (i.e., non-multi) digital signature scheme secure in the Random Oracle model under the discrete log assumption
  - KeyGen: Signing key is x and Verification key is $X = g^x$
  - Sign(m;x): Compute $R=g^r$, $h=H(m,R)$, $s = r + hx$. Output (R,s)

Schnorr Signature

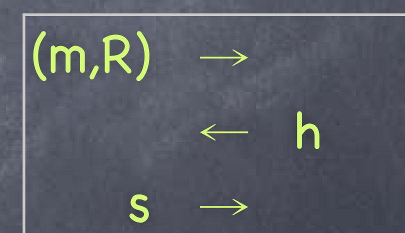
- A regular (i.e., non-multi) digital signature scheme secure in the Random Oracle model under the discrete log assumption
  - KeyGen: Signing key is x and Verification key is $X = g^x$
  - Sign(m;x): Compute $R=g^r$, h=H(m,R), s = r + hx. Output (R,s)
  - Verify(m,(h,s);X): Let h = H(m,R) and check if $g^s = RX^h$

Schnorr Signature

- A regular (i.e., non-multi) digital signature scheme secure in the Random Oracle model under the discrete log assumption
  - KeyGen: Signing key is x and Verification key is $X = g^x$
  - Sign(m;x): Compute $R=g^r$, $h=H(m,R)$, $s = r + hx$. Output (R,s)
  - Verify(m,(h,s);X): Let $h = H(m,R)$ and check if $g^s = RX^h$
- Alternate version: Signature is (h,s). To verify check if $h = H(m,g^sX^{-h})$

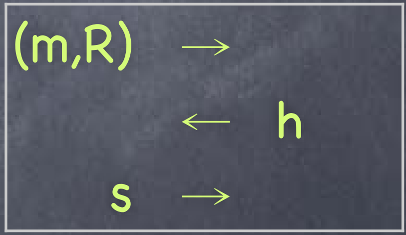
Schnorr Signature

- A regular (i.e., non-multi) digital signature scheme secure in the Random Oracle model under the discrete log assumption
  - KeyGen: Signing key is x and Verification key is $X = g^x$
  - Sign(m;x): Compute $R=g^r$, $h=H(m,R)$, $s = r + hx$. Output (R,s)
  - Verify(m,(h,s);X): Let $h = H(m,R)$ and check if $g^s = RX^h$
- Alternate version: Signature is (h,s). To verify check if $h = H(m,g^sX^{-h})$
- Security: ≈ a ("simulation-sound") HVZK PoK of x



Schnorr Signature

- A regular (i.e., non-multi) digital signature scheme secure in the Random Oracle model under the discrete log assumption
  - KeyGen: Signing key is x and Verification key is $X = g^x$
  - Sign(m;x): Compute $R=g^r$, $h=H(m,R)$, $s = r + hx$. Output (R,s)
  - Verify(m,(h,s);X): Let $h = H(m,R)$ and check if $g^s = RX^h$
- Alternate version: Signature is (h,s). To verify check if $h = H(m,g^sX^{-h})$
- Security: ≈ a ("simulation-sound") HVZK PoK of x



  - Idea: A forger can be used to get distinct signatures (h_1,s_1), (h_2,s_2) with same (m,R) (different h, by programming the RO), and that lets us solve for x from g^x

Schnorr Signature

- A regular (i.e., non-multi) digital signature scheme secure in the Random Oracle model under the discrete log assumption
  - KeyGen: Signing key is x and Verification key is $X = g^x$
  - Sign(m;x): Compute $R=g^r$, $h=H(m,R)$, $s = r + hx$. Output (R,s)
  - Verify(m,(h,s);X): Let $h = H(m,R)$ and check if $g^s = RX^h$
- Alternate version: Signature is (h,s). To verify check if $h = H(m,g^sX^{-h})$
- Security: ≈ a ("simulation-sound") HVZK PoK of x

(m,R)	→	
	←	h
s	→	

  - Idea: A forger can be used to get distinct signatures (h_1,s_1), (h_2,s_2) with same (m,R) (different h, by programming the RO), and that lets us solve for x from g^x
- Extended to a multi-signature scheme [BN'06] →

A Multi-Signature Scheme

A Multi-Signature Scheme

- Schnorr: Sign(m;x) = (R,s) where $R=g^r$, $s = r + hx$ for $h=H(m,R)$. Verify(m,(R,s);X) checks if $g^s = RX^h$ for $h = H(m,R)$

A Multi-Signature Scheme

- Schnorr: Sign(m;x) = (R,s) where $R=g^r$, $s = r + hx$ for $h=H(m,R)$. Verify(m,(R,s);X) checks if $g^s = RX^h$ for $h = H(m,R)$
- For multiple signers with keys $X_1,\ldots,X_n$ can create an “aggregated” signature (R,s) such that $g^s = R.X_1^{h1}\ldots X_n^{hn}$, where:

A Multi-Signature Scheme

- Schnorr: Sign(m;x) = (R,s) where $R=g^r$, s = r + hx for h=H(m,R). Verify(m,(R,s);X) checks if $g^s = RX^h$ for h = H(m,R)
- For multiple signers with keys $X_1,\ldots,X_n$ can create an "aggregated" signature (R,s) such that $g^s = R.X_1^{h1}\ldots X_n^{hn}$, where:
  - Pick R: each party picks r_i and publishes g^{ri}. Set $R = g^{r1+\ldots+rn}$

A Multi-Signature Scheme

- Schnorr: Sign(m;x) = (R,s) where $R=g^r$, s = r + hx for h=H(m,R). Verify(m,(R,s);X) checks if $g^s = RX^h$ for h = H(m,R)
- For multiple signers with keys $X_1,\ldots,X_n$ can create an "aggregated" signature (R,s) such that $g^s = R.X_1^{h1}\ldots X_n^{hn}$, where:
  - Pick R: each party picks r_i and publishes g^{ri}. Set $R = g^{r1+\ldots+rn}$
    - Ensure simultaneous announcement of g^{ri}. (Commit & reveal.)

A Multi-Signature Scheme

- Schnorr: Sign(m;x) = (R,s) where $R=g^r$, s = r + hx for h=H(m,R). Verify(m,(R,s);X) checks if $g^s = RX^h$ for h = H(m,R)
- For multiple signers with keys $X_1,\ldots,X_n$ can create an "aggregated" signature (R,s) such that $g^s = R.X_1^{h1}\ldots X_n^{hn}$, where:
  - Pick R: each party picks r_i and publishes g^{ri}. Set $R = g^{r1+\ldots+rn}$
    - Ensure simultaneous announcement of g^{ri}. (Commit & reveal.)
  - $h_i = H(m,R,X_i,L)$, where $L = \langle X_1,\ldots,X_n \rangle$

A Multi-Signature Scheme

- Schnorr: Sign(m;x) = (R,s) where $R=g^r$, $s = r + hx$ for $h=H(m,R)$. Verify(m,(R,s);X) checks if $g^s = RX^h$ for $h = H(m,R)$
- For multiple signers with keys $X_1,\ldots,X_n$ can create an "aggregated" signature (R,s) such that $g^s = R.X_1^{h1}\ldots X_n^{hn}$, where:
  - Pick R: each party picks r_i and publishes g^{ri}. Set $R = g^{r1+\ldots+rn}$
    - Ensure simultaneous announcement of g^{ri}. (Commit & reveal.)
  - $h_i = H(m,R,X_i,L)$, where $L = \langle X_1,\ldots,X_n \rangle$
  - Then, sequentially $s_i = s_{i-1} + r_i + h_i x_i$ (starting with $s_0 = 0$)

A Multi-Signature Scheme

- Schnorr: Sign(m;x) = (R,s) where $R=g^r$, $s = r + hx$ for $h=H(m,R)$. Verify(m,(R,s);X) checks if $g^s = RX^h$ for $h = H(m,R)$
- For multiple signers with keys $X_1,\ldots,X_n$ can create an "aggregated" signature (R,s) such that $g^s = R.X_1^{h1}\ldots X_n^{hn}$, where:
  - Pick R: each party picks r_i and publishes g^{ri}. Set $R = g^{r1+\ldots+rn}$
    - Ensure simultaneous announcement of g^{ri}. (Commit & reveal.)
  - $h_i = H(m,R,X_i,L)$, where $L = \langle X_1,\ldots,X_n \rangle$
  - Then, sequentially $s_i = s_{i-1} + r_i + h_i x_i$ (starting with $s_0 = 0$)
  - So that final signature $s_n = r + h_1 x_1 + \ldots + h_n x_n$ where $R= g^r$

Waters Signature

Waters Signature

- A regular (non-multi) signature scheme that is secure if the Computational Diffie-Hellman assumption holds in a group with bilinear pairings (no RO)

Waters Signature

- A regular (non-multi) signature scheme that is secure if the Computational Diffie-Hellman assumption holds in a group with bilinear pairings (no RO)
  - Keys: Signing key is x and verification key is $X := e(g,g)^x$, and generators $u_0,u_1,....,u_k$ (for k bit long messages)

Waters Signature

- A regular (non-multi) signature scheme that is secure if the Computational Diffie-Hellman assumption holds in a group with bilinear pairings (no RO)
  - **Keys:** Signing key is x and verification key is $X := e(g,g)^x$, and generators $u_0,u_1,....,u_k$ (for k bit long messages)
  - **Sign(m;x) = (R,S)** where $R=g^r$ and $S = g^x H^r$, where $H = \pi(m) = u_0.u_1^{m1}...u_k^{mk}$

Waters Signature

- A regular (non-multi) signature scheme that is secure if the Computational Diffie-Hellman assumption holds in a group with bilinear pairings (no RO)
  - Keys: Signing key is x and verification key is $X := e(g,g)^x$, and generators $u_0, u_1, \ldots, u_k$ (for k bit long messages)
  - Sign(m;x) = (R,S) where $R=g^r$ and $S = g^x H^r$, where $H = \pi(m) = u_0.u_1^{m1}\ldots u_k^{mk}$
  - Verify(m,(R,S);X,u,$u_1, \ldots, u_k$): check $e(S,g) = e(R,H).X$

Waters Signature

- A regular (non-multi) signature scheme that is secure if the Computational Diffie-Hellman assumption holds in a group with bilinear pairings (no RO)
  - Keys: Signing key is x and verification key is $X := e(g,g)^x$, and generators $u_0,u_1,....,u_k$ (for k bit long messages)
  - Sign(m;x) = (R,S) where $R=g^r$ and $S = g^x H^r$, where $H = \pi(m) = u_0.u_1^{m1}...u_k^{mk}$
  - Verify(m,(R,S);X,u,u_1,....,u_k): check $e(S,g) = e(R,H).X$
- Extended to a multi-signature scheme [LOSSW'06] →

LOSSW Scheme

LOSSW Scheme

- Keys: For user i verification key is $X_i := e(g,g)^{x_i}$, and $u^i_0, u^i_1, \ldots, u^i_k$. Signing key is x_i and $y^i_0, y^i_1, \ldots, y^i_k$ where $u^i_j = g^{y_{ij}}$

LOSSW Scheme

- Keys: For user i verification key is $X_i := e(g,g)^{x_i}$, and $u^i{}_0, u^i{}_1, \ldots, u^i{}_k$. Signing key is x_i and $y^i{}_0, y^i{}_1, \ldots, y^i{}_k$ where $u^i{}_j = g^{y_{ij}}$
- Signature = (R,S), where $R = g^{r_1 + .. + r_n}$, $S = g^{x_1 + .. + x_n} (H_1 \ldots H_n)^{r_1 + .. + r_n}$ where $H_i = u^i{}_0 . (u^i{}_1)^{m_1} \ldots (u^i{}_k)^{m_k}$

LOSSW Scheme

- Keys: For user i verification key is $X_i := e(g,g)^{x_i}$, and $u^i_0, u^i_1, \ldots, u^i_k$. Signing key is x_i and $y^i_0, y^i_1, \ldots, y^i_k$ where $u^i_j = g^{y_{ij}}$
- Signature = (R,S), where $R = g^{r_1 + .. + r_n}$, $S = g^{x_1 + .. + x_n} (H_1 \ldots H_n)^{r_1 + .. + r_n}$ where $H_i = u^i_0 . (u^i_1)^{m_1} \ldots (u^i_k)^{m_k}$
- Verification of signature (R,S): check $e(S,g) = e(R,H_1)X_1 \ldots e(R,H_n)X_n$

LOSSW Scheme

- Keys: For user i verification key is $X_i := e(g,g)^{x_i}$, and $u^i{}_0, u^i{}_1, \ldots, u^i{}_k$. Signing key is x_i and $y^i{}_0, y^i{}_1, \ldots, y^i{}_k$ where $u^i{}_j = g^{y_{ij}}$
- Signature = (R,S), where $R = g^{r_1 + .. + r_n}$, $S = g^{x_1 + .. + x_n} (H_1 \ldots H_n)^{r_1 + .. + r_n}$ where $H_i = u^i{}_0 . (u^i{}_1)^{m_1} \ldots (u^i{}_k)^{m_k}$
- Verification of signature (R,S): check $e(S,g) = e(R,H_1) X_1 \ldots e(R,H_n) X_n$
- Signing done sequentially by individual signers. Initially set R=1 and S = 1 (identity in the group). Then:

LOSSW Scheme

- Keys: For user i verification key is $X_i := e(g,g)^{x_i}$, and $u^i_0, u^i_1, \ldots, u^i_k$. Signing key is x_i and $y^i_0, y^i_1, \ldots, y^i_k$ where $u^i_j = g^{y_{ij}}$
- Signature = (R,S), where $R = g^{r_1 + .. + r_n}$, $S = g^{x_1 + .. + x_n} (H_1 \ldots H_n)^{r_1 + .. + r_n}$ where $H_i = u^i_0 . (u^i_1)^{m_1} \ldots (u^i_k)^{m_k}$
- Verification of signature (R,S): check $e(S,g) = e(R,H_1)X_1 \ldots e(R,H_n)X_n$
- Signing done sequentially by individual signers. Initially set R=1 and S = 1 (identity in the group). Then:
  - AddSign$(m,(R',S'); x_i, y^i_0, y^i_1, \ldots, y^i_k)$ = ReRand(R'',S''), where $R''=R'$ and $S'' = S'.g^{x_i}.(R')^{h_i}$ where h_i s.t. $g^{h_i} = H_i$

LOSSW Scheme

- Keys: For user i verification key is $X_i := e(g,g)^{xi}$, and $u^i_0, u^i_1, \ldots, u^i_k$. Signing key is x_i and $y^i_0, y^i_1, \ldots, y^i_k$ where $u^i_j = g^{yij}$
- Signature = (R,S), where $R = g^{r1+..+rn}$, $S = g^{x1+..+xn} (H_1 \ldots H_n)^{r1+..+rn}$ where $H_i = u^i_0.(u^i_1)^{m1} \ldots (u^i_k)^{mk}$
- Verification of signature (R,S): check $e(S,g) = e(R,H_1)X_1 \ldots e(R,H_n)X_n$
- Signing done sequentially by individual signers. Initially set R=1 and S = 1 (identity in the group). Then:
  - AddSign$(m,(R',S'); x_i, y^i_0, y^i_1, \ldots, y^i_k)$ = ReRand(R'',S''), where $R'' = R'$ and $S'' = S'.g^{xi}.(R')^{hi}$ where h_i s.t. $g^{hi} = H_i$
  - ReRand(R'',S'') = (R,S), where $R = R''g^t$ and $S = S'' (H_1..H_i)^t$

Aggregate Signatures

Aggregate Signatures

- Generalization of multi-signatures where multiple signers may have different messages

Aggregate Signatures

- Generalization of multi-signatures where multiple signers may have different messages
- Sequential aggregation: each signer gets the aggregated signature so far and adds her signature into it (using own signing key)

Aggregate Signatures

- Generalization of multi-signatures where multiple signers may have different messages
- Sequential aggregation: each signer gets the aggregated signature so far and adds her signature into it (using own signing key)
  - e.g., LOSSW'06: If $(m^1,\ldots,m^n)$ where $m^i=(m^i{}_1,\ldots,m^i{}_k)$, then let $H_i = u^i{}_0.(u^i{}_1)^{m^i{}_1}\ldots(u^i{}_k)^{m^i{}_k}$

Aggregate Signatures

- Generalization of multi-signatures where multiple signers may have different messages
- Sequential aggregation: each signer gets the aggregated signature so far and adds her signature into it (using own signing key)
    - e.g., LOSSW'06: If $(m^1,\ldots,m^n)$ where $m^i=(m^i_1,\ldots,m^i_k)$, then let $H_i = u^i_0.(u^i_1)^{m^i_1}\ldots(u^i_k)^{m^i_k}$
- General aggregation: signatures can be created independently and then aggregated in arbitrary order without knowing the secret keys

Batch Verification

Batch Verification

- To speed up verification of a collection of signatures

Batch Verification

- To speed up verification of a collection of signatures
  - Batching done by the verifier

Batch Verification

- To speed up verification of a collection of signatures
  - Batching done by the verifier
  - Incomparable to aggregate signatures

Batch Verification

- To speed up verification of a collection of signatures
  - Batching done by the verifier
  - Incomparable to aggregate signatures
    - Batch verifiable signature scheme reduces verification time, but does not reduce the total size of signatures that verifier gets. No co-ordination among signers.

Batch Verification

- To speed up verification of a collection of signatures
  - Batching done by the verifier
  - Incomparable to aggregate signatures
    - Batch verifiable signature scheme reduces verification time, but does not reduce the total size of signatures that verifier gets. No co-ordination among signers.
    - Aggregate signatures saves on bandwidth and verification time, but needs coordination among signers and does not allow un-aggregating the signatures

Batch Verification

Batch Verification

- Idea: to verify several equations of the form $Z_i = g^{z_i}$, pick random weights w_i and check $\prod_i Z_i^{w_i} = g^{\sum z_i.w_i}$

Batch Verification

- Idea: to verify several equations of the form $Z_i = g^{z_i}$, pick random weights w_i and check $\prod_i Z_i^{w_i} = g^{\sum z_i . w_i}$
  - If one (or more) equation is wrong, probability of verifying is at most $1/q$, where q is the size of the domain of w_i

Batch Verification

- Idea: to verify several equations of the form $Z_i = g^{z_i}$, pick random weights w_i and check $\prod_i Z_i^{w_i} = g^{\sum z_i \cdot w_i}$
  - If one (or more) equation is wrong, probability of verifying is at most 1/q, where q is the size of the domain of w_i
  - Efficiency by using a small domain for w_i. e.g., use $w_i \in \{0,1\}$, and repeat k times (independent of number of signatures)

Batch Verification

- Idea: to verify several equations of the form $Z_i = g^{z_i}$, pick random weights w_i and check $\prod_i Z_i^{w_i} = g^{\sum z_i.w_i}$
  - If one (or more) equation is wrong, probability of verifying is at most 1/q, where q is the size of the domain of w_i
  - Efficiency by using a small domain for w_i. e.g., use $w_i \in \{0,1\}$, and repeat k times (independent of number of signatures)
- Similarly for pairing equations, but with further optimizations

Batch Verification

- Idea: to verify several equations of the form $Z_i = g^{z_i}$, pick random weights w_i and check $\prod_i Z_i^{w_i} = g^{\sum z_i . w_i}$
  - If one (or more) equation is wrong, probability of verifying is at most 1/q, where q is the size of the domain of w_i
  - Efficiency by using a small domain for w_i. e.g., use $w_i \in \{0,1\}$, and repeat k times (independent of number of signatures)
- Similarly for pairing equations, but with further optimizations
  - e.g. Waters' signature: $e(S,g)=e(R,H).X$ (g same for all signers)

Batch Verification

- Idea: to verify several equations of the form $Z_i = g^{z_i}$, pick random weights w_i and check $\Pi_i Z_i^{w_i} = g^{\Sigma z_i.w_i}$
  - If one (or more) equation is wrong, probability of verifying is at most 1/q, where q is the size of the domain of w_i
  - Efficiency by using a small domain for w_i. e.g., use $w_i \in \{0,1\}$, and repeat k times (independent of number of signatures)
- Similarly for pairing equations, but with further optimizations
  - e.g. Waters' signature: $e(S,g)=e(R,H).X$ (g same for all signers)
    - Can save on number of pairing operations using $\Pi_i e(S_i,g)^{w_i} = \Pi_i e(S_i^{w_i},g) = e(\Pi_i S_i^{w_i},g)$

Group Signatures

Group Signatures

- To sign a message “anonymously” [Chaum-vanHeyst’91]

Group Signatures

- To sign a message "anonymously" [Chaum-vanHeyst'91]
  - Signature shows that message was signed by <u>some</u> member of a group

Group Signatures

- To sign a message "anonymously" [Chaum-vanHeyst'91]
  - Signature shows that message was signed by <u>some</u> member of a group
  - But a group manager can "<u>trace</u>" the signer

Group Signatures

- To sign a message "anonymously" [Chaum-vanHeyst'91]
  - Signature shows that message was signed by some member of a group
  - But a group manager can "trace" the signer
  - However, the group manager or other group members "cannot frame" a member

Group Signatures

Group Signatures

- **Full-Anonymity**: Adversary gives (m, ID_0, ID_1) and gets back $Sign(m; ID_b)$ for a random bit b. Advantage of the adversary in finding b should be negligible.

Group Signatures

- **Full-Anonymity**: Adversary gives (m,ID_0,ID_1) and gets back Sign(m;ID_b) for a random bit b. Advantage of the adversary in finding b should be negligible.
  - Even if adversary knows secret keys of all group-members, and has oracle access to the "tracing algorithm" (but not allowed to query it on the challenge)

Group Signatures

- **Full-Anonymity**: Adversary gives (m, ID_0, ID_1) and gets back $Sign(m; ID_b)$ for a random bit b. Advantage of the adversary in finding b should be negligible.
  - Even if adversary knows secret keys of all group-members, and has oracle access to the "tracing algorithm" (but not allowed to query it on the challenge)
  - Implies unlinkability (can't link signatures from same user)

Group Signatures

- **Full-Anonymity**: Adversary gives (m, ID_0, ID_1) and gets back $Sign(m; ID_b)$ for a random bit b. Advantage of the adversary in finding b should be negligible.
  - Even if adversary knows secret keys of all group-members, and has oracle access to the "tracing algorithm" (but not allowed to query it on the challenge)
  - Implies unlinkability (can't link signatures from same user)
- **Full-Traceability**: If a set of group members collude and create a valid signature, the tracing algorithm will trace at least one member of the set. This holds even if the group manager is passively corrupt.

Group Signatures

- **Full-Anonymity**: Adversary gives (m, ID_0, ID_1) and gets back $Sign(m; ID_b)$ for a random bit b. Advantage of the adversary in finding b should be negligible.
  - Even if adversary knows secret keys of all group-members, and has oracle access to the "tracing algorithm" (but not allowed to query it on the challenge)
  - Implies unlinkability (can't link signatures from same user)
- **Full-Traceability**: If a set of group members collude and create a valid signature, the tracing algorithm will trace at least one member of the set. This holds even if the group manager is passively corrupt.
  - Implies unforgeability (i.e., with no group members colluding with it, adversary cannot produce a valid signature) and framing-resistance (even colluding with the group manager)

Group Signatures

Group Signatures

- A general construction: using a digital signature scheme, a CCA secure encryption scheme, and a "simulation-sound" NIZK [BMW'03]

Group Signatures

- A general construction: using a digital signature scheme, a CCA secure encryption scheme, and a "simulation-sound" NIZK [BMW'03]
- Each member's signing key $SK^*_i = (SK_i, VK_i, ID_i, \sigma)$ where (SK_i, VK_i) are signing/verification keys, and σ is a signature (w.r.t. VK_{group}) from the group-manager on (VK_i, ID_i)

Group Signatures

- A general construction: using a digital signature scheme, a CCA secure encryption scheme, and a "simulation-sound" NIZK [BMW'03]
- Each member's signing key $SK^*_i = (SK_i, VK_i, ID_i, \sigma)$ where (SK_i, VK_i) are signing/verification keys, and σ is a signature (w.r.t. VK_{group}) from the group-manager on (VK_i, ID_i)
- Group signature's verification key = $(VK_{group}, PK_{group}, CRS_{group})$

Group Signatures

- A general construction: using a digital signature scheme, a CCA secure encryption scheme, and a "simulation-sound" NIZK [BMW'03]
- Each member's signing key $SK^*_i = (SK_i, VK_i, ID_i, \sigma)$ where (SK_i, VK_i) are signing/verification keys, and σ is a signature (w.r.t. VK_{group}) from the group-manager on (VK_i, ID_i)
- Group signature's verification key = $(VK_{group}, PK_{group}, CRS_{group})$
- Signature is (C, π), where:

Group Signatures

- A general construction: using a digital signature scheme, a CCA secure encryption scheme, and a "simulation-sound" NIZK [BMW'03]
- Each member's signing key $SK^*_i = (SK_i, VK_i, ID_i, \sigma)$ where (SK_i, VK_i) are signing/verification keys, and σ is a signature (w.r.t. VK_{group}) from the group-manager on (VK_i, ID_i)
- Group signature's verification key = $(VK_{group}, PK_{group}, CRS_{group})$
- Signature is (C, π), where:
  - $s = Sign(message; SK_i)$

Group Signatures

- A general construction: using a digital signature scheme, a CCA secure encryption scheme, and a "simulation-sound" NIZK [BMW'03]
- Each member's signing key $SK^*_i = (SK_i,VK_i,ID_i,\sigma)$ where (SK_i,VK_i) are signing/verification keys, and σ is a signature (w.r.t. VK_{group}) from the group-manager on (VK_i,ID_i)
- Group signature's verification key = $(VK_{group}, PK_{group}, CRS_{group})$
- Signature is (C,π), where:
  - $s = \text{Sign}(\text{message}; SK_i)$
  - $C = \text{Encrypt}_{PK_{group}}(s,SK^*_i)$

Group Signatures

- A general construction: using a digital signature scheme, a CCA secure encryption scheme, and a "simulation-sound" NIZK [BMW'03]
- Each member's signing key $SK^*_i = (SK_i, VK_i, ID_i, \sigma)$ where (SK_i, VK_i) are signing/verification keys, and σ is a signature (w.r.t. VK_{group}) from the group-manager on (VK_i, ID_i)
- Group signature's verification key = $(VK_{group}, PK_{group}, CRS_{group})$
- Signature is (C, π), where:
  - $s = Sign(message; SK_i)$
  - $C = Encrypt_{PK_{group}}(s, SK^*_i)$
  - π = a proof (w.r.t CRS_{group}) that C is correct

Group Signatures

- A general construction: using a digital signature scheme, a CCA secure encryption scheme, and a "simulation-sound" NIZK [BMW'03]
- Each member's signing key $SK^*_i = (SK_i, VK_i, ID_i, \sigma)$ where (SK_i, VK_i) are signing/verification keys, and σ is a signature (w.r.t. VK_{group}) from the group-manager on (VK_i, ID_i)
- Group signature's verification key = $(VK_{group}, PK_{group}, CRS_{group})$
- Signature is (C, π), where:
  - $s = \mathrm{Sign}(\mathrm{message}; SK_i)$
  - $C = \mathrm{Encrypt}_{PK_{group}}(s, SK^*_i)$
  - π = a proof (w.r.t CRS_{group}) that C is correct
- Tracing algorithm decrypts C to find SK^*_i and hence ID_i

Ring Signatures

Ring Signatures

- For "leaking secrets"

Ring Signatures

- For “leaking secrets”
  - Similar to group signatures, but with unwitting collaborators

Ring Signatures

- For "leaking secrets"
  - Similar to group signatures, but with unwitting collaborators
    - i.e. the "ring" is not a priori fixed

Ring Signatures

- For “leaking secrets”
  - Similar to group signatures, but with unwitting collaborators
    - i.e. the “ring” is not a priori fixed
  - And no manager who can trace the signer

Ring Signatures

Ring Signatures

- Recall T-OWP/RO based signature

Ring Signatures

- Recall T-OWP/RO based signature
  - (SK,VK) = (F^{-1},F)

Ring Signatures

- Recall T-OWP/RO based signature
  - $(SK,VK) = (F^{-1},F)$
  - $Sign(m;F^{-1}) = F^{-1}(H(m))$

Ring Signatures

- Recall T-OWP/RO based signature
  - $(SK,VK) = (F^{-1},F)$
  - $Sign(m;F^{-1}) = F^{-1}(H(m))$
  - $Verify(S;F)$: check if $H(m) = F(S)$

Ring Signatures

- Recall T-OWP/RO based signature
  - (SK,VK) = (F^{-1},F)
  - Sign(m;F^{-1}) = $F^{-1}(H(m))$
  - Verify(S;F): check if H(m) = F(S)
- Extended to a ring signature [RST'01]

Ring Signatures

- Recall T-OWP/RO based signature
  - $(SK,VK) = (F^{-1},F)$
  - $Sign(m;F^{-1}) = F^{-1}(H(m))$
  - Verify(S;F): check if $H(m) = F(S)$
- Extended to a ring signature [RST'01]
- Verify(m, $(S_1,\ldots,S_n)$; $(F_1,\ldots,F_n)$) : check $H(m) = F_1(S_1) + \ldots + F_n(S_n)$

Ring Signatures

- Recall T-OWP/RO based signature
  - $(SK,VK) = (F^{-1},F)$
  - $Sign(m;F^{-1}) = F^{-1}(H(m))$
  - Verify(S;F): check if $H(m) = F(S)$
- Extended to a ring signature [RST'01]
- Verify(m, $(S_1,\ldots,S_n)$; $(F_1,\ldots,F_n)$) : check $H(m) = F_1(S_1) + \ldots + F_n(S_n)$
- Sign (m; $F_1^{-1},F_2,\ldots,F_n$) = $(S_1,\ldots,S_n)$ where $S_2,\ldots,S_n$ are random and $S_1 = F_1^{-1} (H(m) - F_2(S_2) - \ldots - F_n(S_n))$

Ring Signatures

- Recall T-OWP/RO based signature
  - $(SK,VK) = (F^{-1},F)$
  - $Sign(m;F^{-1}) = F^{-1}(H(m))$
  - $Verify(S;F)$: check if $H(m) = F(S)$
- Extended to a ring signature [RST'01]
- $Verify(m, (S_1,\ldots,S_n); (F_1,\ldots,F_n))$: check $H(m) = F_1(S_1) + \ldots + F_n(S_n)$
- $Sign(m; F_1^{-1},F_2,\ldots,F_n) = (S_1,\ldots,S_n)$ where $S_2,\ldots,S_n$ are random and $S_1 = F_1^{-1}(H(m) - F_2(S_2) - \ldots - F_n(S_n))$
- Unwitting collaborators: F_i's could be the verification keys for a standard signature scheme

Mesh Signatures

Mesh Signatures

- Ring signature allows statements of the form
  (P_1 signed m) or (P_2 signed m) or .... or (P_n signed m)

Mesh Signatures

- Ring signature allows statements of the form
  (P_1 signed m) or (P_2 signed m) or .... or (P_n signed m)
- Mesh signatures extend this to more complex statements

Mesh Signatures

- Ring signature allows statements of the form
  (P_1 signed m) or (P_2 signed m) or .... or (P_n signed m)
- Mesh signatures extend this to more complex statements
  - e.g., (P_1 signed m_1) or ((P_2 signed m_2) and (P_3 signed m_3))

Mesh Signatures

- Ring signature allows statements of the form
  (P_1 signed m) or (P_2 signed m) or .... or (P_n signed m)
- Mesh signatures extend this to more complex statements
  - e.g., (P_1 signed m_1) or ((P_2 signed m_2) and (P_3 signed m_3))
  - e.g., some two out of the three statements (P_1 signed m_1), (P_2 signed m_2), (P_3 signed m_3) hold

Mesh Signatures

- Ring signature allows statements of the form
  (P_1 signed m) or (P_2 signed m) or .... or (P_n signed m)
- Mesh signatures extend this to more complex statements
  - e.g., (P_1 signed m_1) or ((P_2 signed m_2) and (P_3 signed m_3))
  - e.g., some two out of the three statements (P_1 signed m_1), (P_2 signed m_2), (P_3 signed m_3) hold
- Signature is produced by the relevant parties collaborating

Mesh Signatures

- Ring signature allows statements of the form
  (P_1 signed m) or (P_2 signed m) or .... or (P_n signed m)
- Mesh signatures extend this to more complex statements
  - e.g., (P_1 signed m_1) or ((P_2 signed m_2) and (P_3 signed m_3))
  - e.g., some two out of the three statements (P_1 signed m_1), (P_2 signed m_2), (P_3 signed m_3) hold
- Signature is produced by the relevant parties collaborating
- Security requirements: Unforgeability and Hiding

Attribute-Based Signatures

Attribute-Based Signatures

- "Claim-and-endorse": Claim to have attributes satisfying a certain policy, and sign a message

Attribute-Based Signatures

- "Claim-and-endorse": Claim to have attributes satisfying a certain policy, and sign a message
  - Soundness: can't forge, <u>even by colluding</u>

Attribute-Based Signatures

- "Claim-and-endorse": Claim to have attributes satisfying a certain policy, and sign a message
  - Soundness: can't forge, <u>even by colluding</u>
  - Hiding: Verification without learning how the policy was satisfied

Attribute-Based Signatures

- "Claim-and-endorse": Claim to have attributes satisfying a certain policy, and sign a message
  - Soundness: can't forge, <u>even by colluding</u>
  - Hiding: Verification without learning how the policy was satisfied
    - Also unlinkable: cannot link multiple signatures as originating from the same signer

Attribute-Based Signatures

- "Claim-and-endorse": Claim to have attributes satisfying a certain policy, and sign a message
  - Soundness: can't forge, <u>even by colluding</u>
  - Hiding: Verification without learning how the policy was satisfied
    - Also unlinkable: cannot link multiple signatures as originating from the same signer
- c.f. Mesh signatures: here, instead of multiple parties signing a message, a single party with multiple attributes

Undeniable Signatures

Undeniable Signatures

- Suppose Signer wants to control when/how often the signature can be verified, but signature is a commitment to a message

Undeniable Signatures

- Suppose Signer wants to control when/how often the signature can be verified, but signature is a commitment to a message
  - Verification is via an interactive protocol

Undeniable Signatures

- Suppose Signer wants to control when/how often the signature can be verified, but signature is a commitment to a message
  - Verification is via an interactive protocol
  - It lets the signer verifiably accept or deny endorsing the message

Undeniable Signatures

- Suppose Signer wants to control when/how often the signature can be verified, but signature is a commitment to a message
  - Verification is via an interactive protocol
  - It lets the signer verifiably accept or deny endorsing the message
  - Signer refusing to deny can be taken as accepting

Undeniable Signatures

- Suppose Signer wants to control when/how often the signature can be verified, but signature is a commitment to a message
  - Verification is via an interactive protocol
  - It lets the signer verifiably accept or deny endorsing the message
  - Signer refusing to deny can be taken as accepting
- Zero-knowledge verification: A verifier cannot transfer a signature that it verified

Designated Verifier Signatures

Designated Verifier Signatures

- Signature addressed to a single designated verifier

Designated Verifier Signatures

- Signature addressed to a single designated verifier
  - Verifier cannot convince others of the validity of the signature

Designated Verifier Signatures

- Signature addressed to a single designated verifier
  - Verifier cannot convince others of the validity of the signature
  - e.g. a ring signature with a ring of size 2, containing the signer and the designated verifier

Today

Today

- Signatures

Today

- Signatures
  - Multi-signatures

Today

- Signatures
  - Multi-signatures
  - Aggregate Signatures

Today

- Signatures
  - Multi-signatures
  - Aggregate Signatures
  - Signatures with Batch verification

Today

- Signatures
  - Multi-signatures
  - Aggregate Signatures
  - Signatures with Batch verification
  - Group signatures

Today

- Signatures
  - Multi-signatures
  - Aggregate Signatures
  - Signatures with Batch verification
  - Group signatures
  - Ring and Mesh signatures

Today

- Signatures
  - Multi-signatures
  - Aggregate Signatures
  - Signatures with Batch verification
  - Group signatures
  - Ring and Mesh signatures
  - Attribute-Based signatures

Today

- Signatures
  - Multi-signatures
  - Aggregate Signatures
  - Signatures with Batch verification
  - Group signatures
  - Ring and Mesh signatures
  - Attribute-Based signatures
  - Undeniable signatures

Today

- Signatures
  - Multi-signatures
  - Aggregate Signatures
  - Signatures with Batch verification
  - Group signatures
  - Ring and Mesh signatures
  - Attribute-Based signatures
  - Undeniable signatures
  - Designated verifier signatures

Today

- Signatures
  - Multi-signatures
  - Aggregate Signatures
  - Signatures with Batch verification
  - Group signatures
  - Ring and Mesh signatures
  - Attribute-Based signatures
  - Undeniable signatures
  - Designated verifier signatures
- Not discussed: Blind signatures, P-signatures, anonymous credentials...