Signatures

Lecture 24

Signatures

Signatures

- Signatures with various functionality/properties

Signatures

- Signatures with various functionality/properties
- Constructions come in different flavors:

Signatures

- Signatures with various functionality/properties
- Constructions come in different flavors:
  - Using minimal/general assumptions, often simple, but not very efficient (e.g., involving NIZK for general NP statements)

Signatures

- Signatures with various functionality/properties
- Constructions come in different flavors:
  - Using minimal/general assumptions, often simple, but not very efficient (e.g., involving NIZK for general NP statements)
  - Simple and efficient ones in the Random Oracle Model

Signatures

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

Signatures

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

A Multi-Signature Scheme

A Multi-Signature Scheme

- Following Schnorr Signature: a digital signature scheme secure in the Random Oracle model under the discrete log assumption

A Multi-Signature Scheme

- Following Schnorr Signature: a digital signature scheme secure in the Random Oracle model under the discrete log assumption
  - Signing key is x and Verification key is $X = g^x$

A Multi-Signature Scheme

- Following Schnorr Signature: a digital signature scheme secure in the Random Oracle model under the discrete log assumption
  - 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 (h,s)

A Multi-Signature Scheme

- Following Schnorr Signature: a digital signature scheme secure in the Random Oracle model under the discrete log assumption
  - 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 (h,s)
  - Verify(m,(h,s);X): check if $h = H(m,g^s X^{-h})$

A Multi-Signature Scheme

- Following Schnorr Signature: a digital signature scheme secure in the Random Oracle model under the discrete log assumption
  - 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 (h,s)
  - Verify(m,(h,s);X): check if $h = H(m,g^s X^{-h})$
- Alternately Sign(m;x) outputs (R,s). Verify(m,(R,s);X) checks if $g^s = RX^h$ for $h = H(m,R)$

A Multi-Signature Scheme

- Following Schnorr Signature: a digital signature scheme secure in the Random Oracle model under the discrete log assumption
  - 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 (h,s)
  - Verify(m,(h,s);X): check if $h = H(m,g^s X^{-h})$
- Alternately Sign(m;x) outputs (R,s). Verify(m,(R,s);X) checks if $g^s = RX^h$ for $h = H(m,R)$
- Security by showing that a forger can be used to get distinct signatures (h_1,s_1), (h_2,s_2) with same (m,R) (but different h, by programming the RO) that lets us solve for x

A Multi-Signature Scheme

- Following Schnorr Signature: a digital signature scheme secure in the Random Oracle model under the discrete log assumption
  - 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 (h,s)
  - Verify(m,(h,s);X): check if $h = H(m,g^sX^{-h})$
- Alternately Sign(m;x) outputs (R,s). Verify(m,(R,s);X) checks if $g^s = RX^h$ for $h = H(m,R)$
- Security by showing that a forger can be used to get distinct signatures (h_1,s_1), (h_2,s_2) with same (m,R) (but different h, by programming the RO) that lets us solve for 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 $h_i = H(m,R,X_i,n)$

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 $h_i = H(m,R,X_i,n)$
  - Signing done sequentially by individual signers (user i has x_i). Initially set R=1 (identity in the group) and s=0. Then:

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 $h_i = H(m,R,X_i,n)$

  - Signing done sequentially by individual signers (user i has x_i). Initially set R=1 (identity in the group) and s=0. Then:

  - AddSign(m;(R',s');x_i) = (R,s) where $R=R'.g^{r_i}$ and $s = s' + r_i + h_i x_i$

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 $h_i = H(m,R,X_i,n)$
  - Signing done sequentially by individual signers (user i has x_i). Initially set R=1 (identity in the group) and s=0. Then:
  - AddSign(m;(R′,s′);x_i) = (R,s) where $R=R'.g^{ri}$ and $s = s' + r_i + h_i x_i$
    - So that finally $R = g^r = g^{r1+\ldots+rn}$ and $s = r + h_1x_1 + \ldots + h_nx_n$

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 $h_i = H(m,R,X_i,n)$
  - Signing done sequentially by individual signers (user i has x_i). Initially set R=1 (identity in the group) and s=0. Then:
  - AddSign(m;(R',s');x_i) = (R,s) where $R=R'.g^{ri}$ and $s = s' + r_i + h_i x_i$
    - So that finally $R = g^r = g^{r1+\ldots+rn}$ and $s = r + h_1x_1 + \ldots + h_nx_n$
- Security by showing that a forger succeeds with same (m,R) when given two distinct answers for h_1 (by programming the RO)

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

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
- General aggregation: signatures can be created independently and then aggregated in arbitrary order

A Sequential Aggregate Signature Scheme

A Sequential Aggregate Signature Scheme

- Water's Signature: Secure if the Computational Diffie-Hellman assumption holds in a group with bilinear pairings (no RO)

A Sequential Aggregate Signature Scheme

- Water's Signature: Secure if the Computational Diffie-Hellman assumption holds in a group with bilinear pairings (no RO)
  - 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)

A Sequential Aggregate Signature Scheme

- Water's Signature: Secure if the Computational Diffie-Hellman assumption holds in a group with bilinear pairings (no RO)
  - 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 $H = u_0 . u_1^{m1} \ldots u_k^{mk}$

A Sequential Aggregate Signature Scheme

- Water's Signature: Secure if the Computational Diffie-Hellman assumption holds in a group with bilinear pairings (no RO)
  - 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 $H = u_0.u_1^{m1}...u_k^{mk}$
  - $Verify(m,(R,S);X,u,u_1,....,u_k)$: check $e(S,g) = e(R,H).X$

A Sequential Aggregate Signature Scheme

- Water's Signature: Secure if the Computational Diffie-Hellman assumption holds in a group with bilinear pairings (no RO)
  - 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 $H = u_0.u_1^{m1}...u_k^{mk}$
  - $Verify(m,(R,S);X,u,u_1,....,u_k)$: check $e(S,g) = e(R,H).X$
- Extended to a sequential aggregate scheme [LOSSW'06]

A Sequential Aggregate Signature Scheme

A Sequential Aggregate Signature Scheme

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

A Sequential Aggregate Signature Scheme

- 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 will be (R,S) where $R = g^{r_1 + .. + r_n}$, $S = g^{x_1 + .. x_n} (H_1 \ldots H_n)^{r_1 + .. + r_n}$

A Sequential Aggregate Signature Scheme

- 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 will be (R,S) where $R = g^{r1+..+rn}$, $S = g^{x1+..xn} (H_1 \ldots H_n)^{r1+..+rn}$
- Verification of signature (R,S) for messages $(m^1, \ldots, m^n)$: check if $e(S,g) = e(R,H_1)X_1 \ldots e(R,H_n)X_n$ where $H_i = u^i{}_0.(u^i{}_1)^{m1} \ldots (u^i{}_k)^{mk}$

A Sequential Aggregate Signature Scheme

- 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 will be (R,S) where $R = g^{r1+..+rn}$, $S = g^{x1+..xn} (H_1 \ldots H_n)^{r1+..+rn}$
- Verification of signature (R,S) for messages $(m^1, \ldots, m^n)$: check if $e(S,g) = e(R,H_1)X_1 \ldots e(R,H_n)X_n$ where $H_i = u^i{}_0 . (u^i{}_1)^{m1} \ldots (u^i{}_k)^{mk}$
- Signing done sequentially by individual signers. Initially set R=1 and S = 1 (identity in the group). Then:

A Sequential Aggregate Signature Scheme

- 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 will be (R,S) where $R = g^{r_1 + \ldots + r_n}$, $S = g^{x_1 + \ldots x_n} (H_1 \ldots H_n)^{r_1 + \ldots + r_n}$
- Verification of signature (R,S) for messages $(m^1, \ldots, m^n)$: check if $e(S,g) = e(R,H_1)X_1 \ldots e(R,H_n)X_n$ where $H_i = u^i_0.(u^i_1)^{m_1} \ldots (u^i_k)^{m_k}$
- Signing done sequentially by individual signers. Initially set R=1 and S = 1 (identity in the group). Then:
    - $\text{AddSign}(m^i,(R',S'); x_i, y^i_0, y^i_1, \ldots, y^i_k) = \text{ReRand}(R'',S'')$, where $R''=R'$ and $S'' = S'.g^{x_i}.(R')^{h_i}$ where h_i s.t. $g^{h_i} = H_i$

A Sequential Aggregate Signature Scheme

- 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 will be (R,S) where $R = g^{r_1+..+r_n}$, $S = g^{x_1+..x_n} (H_1 \ldots H_n)^{r_1+..+r_n}$
- Verification of signature (R,S) for messages $(m^1, \ldots, m^n)$: check if $e(S,g) = e(R,H_1)X_1 \ldots e(R,H_n)X_n$ where $H_i = u^i_0.(u^i_1)^{m_1} \ldots (u^i_k)^{m_k}$
- Signing done sequentially by individual signers. Initially set R=1 and S = 1 (identity in the group). Then:
  - $\text{AddSign}(m^i,(R',S'); x_i, y^i_0, y^i_1, \ldots, y^i_k) = \text{ReRand}(R'',S'')$, where $R''=R'$ and $S'' = S'.g^{x_i}.(R')^{h_i}$ where h_i s.t. $g^{h_i} = H_i$
  - $\text{ReRand}(R'',S'') = (R,S)$, where $R = R''g^t$ and $S = S'' (H_1..H_i)^t$

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. Retains individual signatures (that can be forwarded/re-batched).

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. Retains individual signatures (that can be forwarded/re-batched).
    - Aggregate signatures saves on bandwidth and verification time, but 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 $\Pi_i \, Z_i^{w_i} = g^{\Sigma \, z_i.w_i}$

Batch Verification

- Idea: to verify several equations of the form $Z_i = g^{z_i}$, pick random weights w_i and check $\Pi_i Z_i^{w_i} = g^{\Sigma z_i . w_i}$
  - If one (or more) equation is wrong, probability of verifying is $1/q$, where q is the size of the domain of w_i. Efficiency by using a small domain (say $\{0,1\}$) for w_i.

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 1/q, where q is the size of the domain of w_i. Efficiency by using a small domain (say {0,1}) for w_i.
  - Can 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 $\Pi_i Z_i^{w_i} = g^{\Sigma z_i.w_i}$
  - If one (or more) equation is wrong, probability of verifying is $1/q$, where q is the size of the domain of w_i. Efficiency by using a small domain (say {0,1}) for w_i.
  - Can 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 $\Pi_i Z_i^{w_i} = g^{\Sigma z_i.w_i}$
  - If one (or more) equation is wrong, probability of verifying is 1/q, where q is the size of the domain of w_i. Efficiency by using a small domain (say {0,1}) for w_i.
  - Can 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 1/q, where q is the size of the domain of w_i. Efficiency by using a small domain (say {0,1}) for w_i.
  - Can 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" [CvH'91]

Group Signatures

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

Group Signatures

- To sign a message "anonymously" [CvH'91]
  - Signature shows that message was signed by some member of a group
  - But a group manager can "trace" the signer

Group Signatures

- To sign a message "anonymously" [CvH'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.
  - 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.
  - 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.
  - 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.
  - 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 consists of a key-pair (VK_i, SK_i) and a certificate from the group-manager for VK_i (optionally binding it to 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 consists of a key-pair (VK_i,SK_i) and a certificate from the group-manager for VK_i (optionally binding it to ID_i)
- Signature is (C,π), C being an encryption of (s,SK^*_i) where s is a signature on the message using SK_i and π being a proof (w.r.t a CRS in the group's public-key) that C is correct (and in particular, contains a correct s and SK*)

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 consists of a key-pair (VK_i, SK_i) and a certificate from the group-manager for VK_i (optionally binding it to ID_i)
- Signature is (C,π), C being an encryption of (s, SK^*_i) where s is a signature on the message using SK_i and π being a proof (w.r.t a CRS in the group's public-key) that C is correct (and in particular, contains a correct s and SK*)
- 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$(m, (S_1,\dots,S_n); (F_1,\dots,F_n))$: check $H(m) = F_1(S_1) + \dots + F_n(S_n)$

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, even by colluding
  - 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, even by colluding
  - 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

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
- Note: Still allows multiple (mutually distrusting) verifiers to be convinced if they run a secure MPC protocol to implement a virtual verifier

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
- Next up: digital cash

Today

- Signatures
  - Multi-signatures
  - Aggregate Signatures
  - Signatures with Batch verification
  - Group signatures
  - Ring and Mesh signatures
  - Attribute-Based signatures
  - Undeniable signatures
  - Designated verifier signatures
- Next up: digital cash
  - Using Blind signatures and P-signatures