Broadcast Encryption and Some Other Primitives

Lecture 24

Broadcast Encryption

Broadcast Encryption

- Encrypt to a subset of users in the system

Broadcast Encryption

- Encrypt to a subset of users in the system
  - e.g., subscribers who haven't been revoked

Broadcast Encryption

- Encrypt to a subset of users in the system
  - e.g., subscribers who haven't been revoked
- Subset <u>not known</u> at time of setup (when users get private keys)

Broadcast Encryption

- Encrypt to a subset of users in the system
  - e.g., subscribers who haven't been revoked
- Subset not known at time of setup (when users get private keys)
- Trivial solution 1: encrypt to each user separately

Broadcast Encryption

- Encrypt to a subset of users in the system
  - e.g., subscribers who haven't been revoked
- Subset not known at time of setup (when users get private keys)
- Trivial solution 1: encrypt to each user separately
  - Size of ciphertext is proportional to the number of users

Broadcast Encryption

- Encrypt to a subset of users in the system
  - e.g., subscribers who haven't been revoked
- Subset not known at time of setup (when users get private keys)
- Trivial solution 1: encrypt to each user separately
  - Size of ciphertext is proportional to the number of users
- Trivial solution 2: for each possible subset, use a different key

Broadcast Encryption

- Encrypt to a subset of users in the system
  - e.g., subscribers who haven't been revoked
- Subset not known at time of setup (when users get private keys)
- Trivial solution 1: encrypt to each user separately
  - Size of ciphertext is proportional to the number of users
- Trivial solution 2: for each possible subset, use a different key
  - Size of private key for each user is exponential

Broadcast Encryption

- Encrypt to a subset of users in the system
  - e.g., subscribers who haven't been revoked
- Subset not known at time of setup (when users get private keys)
- Trivial solution 1: encrypt to each user separately
  - Size of ciphertext is proportional to the number of users
- Trivial solution 2: for each possible subset, use a different key
  - Size of private key for each user is exponential
- Question: Can we do better?

Broadcast Encryption

- Encrypt to a subset of users in the system
  - e.g., subscribers who haven't been revoked
- Subset <u>not known</u> at time of setup (when users get private keys)
- Trivial solution 1: encrypt to each user separately
  - Size of ciphertext is proportional to the number of users
- Trivial solution 2: for each possible subset, use a different key
  - Size of private key for each user is exponential
- Question: Can we do better?
  - c.f. (Ciphertext Policy) Attribute-Based Encryption: set of recipients decided dynamically

Broadcast Encryption

Broadcast Encryption

- Typical scenario considered: set of all users large, set of revoked users small

Broadcast Encryption

- Typical scenario considered: set of all users large, set of revoked users small
  - Size of private-keys can depend on the number of users

Broadcast Encryption

- Typical scenario considered: set of all users large, set of revoked users small
  - Size of private-keys can depend on the number of users
  - Size of ciphertext can depend on the number of revoked users

Broadcast Encryption

- Typical scenario considered: set of all users large, set of revoked users small
  - Size of private-keys can depend on the number of users
  - Size of ciphertext can depend on the number of revoked users
  - Only a privileged broadcaster need to be able to encrypt

Broadcast Encryption

- Typical scenario considered: set of all users large, set of revoked users small
  - Size of private-keys can depend on the number of users
  - Size of ciphertext can depend on the number of revoked users
  - Only a privileged broadcaster need to be able to encrypt
- Security: No PPT adversary that obtains keys for all revoked users should have a non-negligible advantage in an IND-CPA (or IND-CCA) game

Broadcast Encryption

- Typical scenario considered: set of all users large, set of revoked users small
  - Size of private-keys can depend on the number of users
  - Size of ciphertext can depend on the number of revoked users
  - Only a privileged broadcaster need to be able to encrypt
- Security: No PPT adversary that obtains keys for all revoked users should have a non-negligible advantage in an IND-CPA (or IND-CCA) game
  - Set of revoked users is determined first (static corruption), or adaptively based on the public parameters, encryptions, and keys of users revoked so far

Broadcast Encryption

- Typical scenario considered: set of all users large, set of revoked users small
  - Size of private-keys can depend on the number of users
  - Size of ciphertext can depend on the number of revoked users
  - Only a privileged broadcaster need to be able to encrypt
- Security: No PPT adversary that obtains keys for all revoked users should have a non-negligible advantage in an IND-CPA (or IND-CCA) game
  - Set of revoked users is determined first (static corruption), or adaptively based on the public parameters, encryptions, and keys of users revoked so far
  - Note: revoked users collude

Using Subset Covers

Using Subset Covers

- Subset-Cover approach [Naor-Naor-Lotspiech'01]

Using Subset Covers

- Subset-Cover approach [Naor-Naor-Lotspiech'01]
  - Define subsets of the universe $X_1,\ldots,X_m$

Using Subset Covers

- Subset-Cover approach [Naor-Naor-Lotspiech'01]
  - Define subsets of the universe $X_1,...,X_m$
  - For each X_j create a secret key K_j for a PRF and give it to all parties in X_j

Using Subset Covers

- Subset-Cover approach [Naor-Naor-Lotspiech'01]
  - Define subsets of the universe $X_1,\ldots,X_m$
  - For each X_j create a secret key K_j for a PRF and give it to all parties in X_j
    - PRF/Block-cipher to be used as a semantically secure (multi-message) symmetric-key encryption scheme

Using Subset Covers

- Subset-Cover approach [Naor-Naor-Lotspiech'01]
  - Define subsets of the universe $X_1,\ldots,X_m$
  - For each X_j create a secret key K_j for a PRF and give it to all parties in X_j
    - PRF/Block-cipher to be used as a semantically secure (multi-message) symmetric-key encryption scheme
  - To encrypt a message to a set S find subsets $X_{j1},\ldots,X_{jt}$ which form a <u>cover</u> of S, and encrypt the message under each key K_{ji}. All ciphertexts are broadcast.

Using Subset Covers

- Subset-Cover approach [Naor-Naor-Lotspiech'01]
  - Define subsets of the universe $X_1, \ldots, X_m$
  - For each X_j create a secret key K_j for a PRF and give it to all parties in X_j
    - PRF/Block-cipher to be used as a semantically secure (multi-message) symmetric-key encryption scheme
  - To encrypt a message to a set S find subsets $X_{j1}, \ldots, X_{jt}$ which form a <u>cover</u> of S, and encrypt the message under each key K_{ji}. All ciphertexts are broadcast.
  - Can use "hybrid encryption": encrypt a fresh key for a one-time encryption scheme (seed of a PRG), and use that key to encrypt the message

Using Subset Covers

- Subset-Cover approach [Naor-Naor-Lotspiech'01]

Using Subset Covers

- Subset-Cover approach [Naor-Naor-Lotspiech'01]
  - To encrypt a message to a set S find subsets $X_{j1},\ldots,X_{jt}$ whose union is S, and encrypt the message under each key K_{ji}

Using Subset Covers

- Subset-Cover approach [Naor-Naor-Lotspiech'01]
  - To encrypt a message to a set S find subsets $X_{j1},\ldots,X_{jt}$ whose union is S, and encrypt the message under each key K_{ji}
  - Goal: design $X_1,\ldots,X_m$ such that any set S can be obtained as the union of a few sets X_j

Using Subset Covers

- Subset-Cover approach [Naor-Naor-Lotspiech'01]
  - To encrypt a message to a set S find subsets $X_{j1},\ldots,X_{jt}$ whose union is S, and encrypt the message under each key K_{ji}
  - Goal: design $X_1,\ldots,X_m$ such that any set S can be obtained as the union of a few sets X_j
    - While keeping the total number of sets X_j not too large

Using Subset Covers

- Subset-Cover approach [Naor-Naor-Lotspiech'01]
  - To encrypt a message to a set S find subsets $X_{j1},\ldots,X_{jt}$ whose union is S, and encrypt the message under each key K_{ji}
  - Goal: design $X_1,\ldots,X_m$ such that any set S can be obtained as the union of a few sets X_j
    - While keeping the total number of sets X_j not too large
      - Each user gets keys for each X_j that it belongs to

Using Subset Covers

- Subset-Cover approach [Naor-Naor-Lotspiech'01]
  - To encrypt a message to a set S find subsets $X_{j1},\ldots,X_{jt}$ whose union is S, and encrypt the message under each key K_{ji}
  - Goal: design $X_1,\ldots,X_m$ such that any set S can be obtained as the union of a few sets X_j
    - While keeping the total number of sets X_j not too large
      - Each user gets keys for each X_j that it belongs to
    - Will settle for S such that it has at most r users revoked

Subtree Covers

Subtree Covers

- Define a balanced binary tree with leaves corresponding to the set of users {1,..,n}

Subtree Covers

- Define a balanced binary tree with leaves corresponding to the set of users {1,..,n}
- For each node u, define set X_u as the set of leaves of the subtree rooted at u

Subtree Covers

- Define a balanced binary tree with leaves corresponding to the set of users {1,..,n}
- For each node u, define set X_u as the set of leaves of the subtree rooted at u
- Can find O(r log n) sets X_u that cover any set S with at most r missing (revoked) leaves [How?]

Subtree Covers

- Define a balanced binary tree with leaves corresponding to the set of users {1,..,n}
- For each node u, define set X_u as the set of leaves of the subtree rooted at u
- Can find O(r log n) sets X_u that cover any set S with at most r missing (revoked) leaves [How?]
- Each user appears in O(log n) sets

Subtree-Difference Covers

Subtree-Difference Covers

- Define a balanced binary tree with leaves corresponding to the set of users {1,..,n}

Subtree-Difference Covers

- Define a balanced binary tree with leaves corresponding to the set of users {1,..,n}
- For each pair of nodes (u,v), with v being a descendent of u, define set X_{uv} as the set of leaves of the subtree rooted at u that are not in the subtree rooted at v

Subtree-Difference Covers

- Define a balanced binary tree with leaves corresponding to the set of users {1,..,n}
- For each pair of nodes (u,v), with v being a descendent of u, define set X_{uv} as the set of leaves of the subtree rooted at u that are not in the subtree rooted at v
- Can find 2r-1 sets X_u that cover any set S with r missing (revoked) leaves [How?]

Subtree-Difference Covers

- Define a balanced binary tree with leaves corresponding to the set of users {1,..,n}
- For each pair of nodes (u,v), with v being a descendent of u, define set X_{uv} as the set of leaves of the subtree rooted at u that are not in the subtree rooted at v
- Can find 2r-1 sets X_u that cover any set S with r missing (revoked) leaves [How?]
- Each user appears in O(n) sets

Subtree-Difference Covers

- Define a balanced binary tree with leaves corresponding to the set of users {1,..,n}
- For each pair of nodes (u,v), with v being a descendent of u, define set X_{uv} as the set of leaves of the subtree rooted at u that are not in the subtree rooted at v
- Can find 2r-1 sets X_u that cover any set S with r missing (revoked) leaves [How?]
- Each user appears in O(n) sets
  - But can use PRG to derive keys so that each user hold only $O(\log^2 n)$ different keys

Subtree-Difference Covers

Subtree-Difference Covers

- Pick random meta-keys $M_{u,u}$ for each node, which is used to derive, for each v, the key K_{uv} for set X_{uv}

Subtree-Difference Covers

- Pick random meta-keys $M_{u,u}$ for each node, which is used to derive, for each v, the key K_{uv} for set X_{uv}
  - Derive keys recursively using a PRF (or a length-tripling PRG): $M_{u,v0} = F_{M_{u,v}}(0)$, $M_{u,v1} = F_{M_{u,v}}(1)$ and $K_{u,v} = F_{M_{u,v}}(2)$ (where v0 and v1 are the children of v)

Subtree-Difference Covers

- Pick random meta-keys $M_{u,u}$ for each node, which is used to derive, for each v, the key K_{uv} for set X_{uv}
  - Derive keys recursively using a PRF (or a length-tripling PRG): $M_{u,v0} = F_{M_{u,v}}(0)$, $M_{u,v1} = F_{M_{u,v}}(1)$ and $K_{u,v} = F_{M_{u,v}}(2)$ (where v0 and v1 are the children of v)
  - Deliver to a party at leaf w, for each ancestor u, log n keys: for each node v' on the path u-w, let v be the sibling of v'; give $M_{u,v}$. $O(\log^2 n)$ keys in all for each party.

Subtree-Difference Covers

- Pick random meta-keys $M_{u,u}$ for each node, which is used to derive, for each v, the key K_{uv} for set X_{uv}
  - Derive keys recursively using a PRF (or a length-tripling PRG): $M_{u,v0} = F_{M_{u,v}}(0)$, $M_{u,v1} = F_{M_{u,v}}(1)$ and $K_{u,v} = F_{M_{u,v}}(2)$ (where v0 and v1 are the children of v)
  - Deliver to a party at leaf w, for each ancestor u, log n keys: for each node v′ on the path u-w, let v be the sibling of v′; give $M_{u,v}$. $O(\log^2 n)$ keys in all for each party.
    - If $X_{uu'}$ covers a party at leaf w, it can derive $K_{uu'}$: Let v be the highest ancestor of u′ for which w is not a descendent (i.e., v's sibling is on the u-w path). Use $M_{u,v}$ to derive $K_{uu'}$.

Using Secret-Sharing

Using Secret-Sharing

- A secret-sharing based scheme [Naor-Pinkas'00]

Using Secret-Sharing

- A secret-sharing based scheme [Naor-Pinkas'00]
- One-time revocation scheme (using any CPA-secure encryption)

Using Secret-Sharing

- A secret-sharing based scheme [Naor-Pinkas'00]
- One-time revocation scheme (using any CPA-secure encryption)
  - Share a key K using an (r+1) out of n secret-sharing. Give the share K_i to user i

Using Secret-Sharing

- A secret-sharing based scheme [Naor-Pinkas'00]
- One-time revocation scheme (using any CPA-secure encryption)
  - Share a key K using an (r+1) out of n secret-sharing. Give the share K_i to user i
  - To revoke a set of r users (including some dummy users, if necessary), broadcast their shares, and encrypt the message using the key K

Using Secret-Sharing

- A secret-sharing based scheme [Naor-Pinkas'00]
- One-time revocation scheme (using any CPA-secure encryption)
  - Share a key K using an (r+1) out of n secret-sharing. Give the share K_i to user i
  - To revoke a set of r users (including some dummy users, if necessary), broadcast their shares, and encrypt the message using the key K
  - Only parties not in the revoked set can reconstruct K

Using Secret-Sharing

- A secret-sharing based scheme [Naor-Pinkas'00]
- One-time revocation scheme (using any CPA-secure encryption)
  - Share a key K using an (r+1) out of n secret-sharing. Give the share K_i to user i
  - To revoke a set of r users (including some dummy users, if necessary), broadcast their shares, and encrypt the message using the key K
  - Only parties not in the revoked set can reconstruct K
- Many-times revocation scheme (secure under DDH)

Using Secret-Sharing

- A secret-sharing based scheme [Naor-Pinkas'00]
- One-time revocation scheme (using any CPA-secure encryption)
  - Share a key K using an (r+1) out of n secret-sharing. Give the share K_i to user i
  - To revoke a set of r users (including some dummy users, if necessary), broadcast their shares, and encrypt the message using the key K
  - Only parties not in the revoked set can reconstruct K
- Many-times revocation scheme (secure under DDH)
  - Broadcast g^x, Mg^{Kx}, and $g^{Ki.x}$ for each i being revoked. Each non-revoked party can reconstruct g^{Kx} (but not K, or g^K)

Using Secret-Sharing

- A secret-sharing based scheme [Naor-Pinkas'00]
- One-time revocation scheme (using any CPA-secure encryption)
  - Share a key K using an (r+1) out of n secret-sharing. Give the share K_i to user i
  - To revoke a set of r users (including some dummy users, if necessary), broadcast their shares, and encrypt the message using the key K
  - Only parties not in the revoked set can reconstruct K
- Many-times revocation scheme (secure under DDH)
  - Broadcast g^x, Mg^{Kx}, and $g^{K_i.x}$ for each i being revoked. Each non-revoked party can reconstruct g^{Kx} (but not K, or g^K)
- Ciphertext size proportional to the size of the set being revoked

Using Bilinear Pairings

Using Bilinear Pairings

- A public-key scheme, with short ciphertexts, supporting arbitrary set sizes [Boneh-Gentry-Waters'05]

Using Bilinear Pairings

- A public-key scheme, with short ciphertexts, supporting arbitrary set sizes [Boneh-Gentry-Waters'05]
- Public parameters: $e(g,g)^z$, $u_1,\ldots,u_n$ for n users

Using Bilinear Pairings

- A public-key scheme, with short ciphertexts, supporting arbitrary set sizes [Boneh-Gentry-Waters'05]
- Public parameters: $e(g,g)^z$, $u_1,\ldots,u_n$ for n users
- Secret Key for user i: $R_i := g^{r_i}$, $u_j^{r_i}$ for $j \neq i$, and $K_i := g^z u_i^{r_i}$

Using Bilinear Pairings

- A public-key scheme, with short ciphertexts, supporting arbitrary set sizes [Boneh-Gentry-Waters'05]
- Public parameters: $e(g,g)^z$, $u_1,\ldots,u_n$ for n users
- Secret Key for user i: $R_i := g^{r_i}$, $u_j^{r_i}$ for $j \neq i$, and $K_i := g^z u_i^{r_i}$
- $\text{Encrypt}_{PK,S}(M;x) := (g^x,\ M\ e(g,g)^{zx},\ H(S)^x\ )$ where S is the set of users allowed to decrypt, x is randomly chosen, and $H(S) := \prod_{j \in S} u_j$

Using Bilinear Pairings

- A public-key scheme, with short ciphertexts, supporting arbitrary set sizes [Boneh-Gentry-Waters'05]
- Public parameters: $e(g,g)^z$, $u_1,\ldots,u_n$ for n users
- Secret Key for user i: $R_i := g^{r_i}$, $u_j^{r_i}$ for $j \neq i$, and $K_i := g^z u_i^{r_i}$
- $\text{Encrypt}_{PK,S}(M;x) := (g^x,\ M\ e(g,g)^{zx},\ H(S)^x\ )$ where S is the set of users allowed to decrypt, x is randomly chosen, and $H(S) := \Pi_{j \in S}\ u_j$
- Decryption (by $i \in S$): From $e(g^x, \Pi_{j \in S \setminus \{i\}}\ u_j^{r_i}\ ) \ /\ e(R_i, H(S)^x) = e(g,u_i)^{-r_i \cdot x}$ and $e(g^x,K_i) = e(g,g)^{zx}\ e(g,u_i)^{r_i \cdot x}$, get $e(g,g)^{zx}$ and hence M

Using Bilinear Pairings

- A public-key scheme, with short ciphertexts, supporting arbitrary set sizes [Boneh-Gentry-Waters'05]
- Public parameters: $e(g,g)^z$, $u_1,\ldots,u_n$ for n users
- Secret Key for user i: $R_i := g^{r_i}$, $u_j^{r_i}$ for $j \neq i$, and $K_i := g^z u_i^{r_i}$
- $\text{Encrypt}_{PK,S}(M;x) := (g^x,\ M\ e(g,g)^{zx},\ H(S)^x\ )$ where S is the set of users allowed to decrypt, x is randomly chosen, and $H(S) := \prod_{j\in S} u_j$
- Decryption (by $i \in S$): From $e(g^x, \prod_{j\in S\setminus\{i\}} u_j^{r_i}\ ) \ /\ e(R_i,H(S)^x) = e(g,u_i)^{-r_i \cdot x}$ and $e(g^x,K_i) = e(g,g)^{zx}\ e(g,u_i)^{r_i \cdot x}$, get $e(g,g)^{zx}$ and hence M
  - Security relies on an indistinguishability assumption involving O(n) group elements (cf. DDH has 3 group elements)

Traitor Tracing

Traitor Tracing

- A legitimate user (paid subscriber) may sell pirated devices/ software for decryption

Traitor Tracing

- A legitimate user (paid subscriber) may sell pirated devices/ software for decryption
  - To detect such a user

Traitor Tracing

- A legitimate user (paid subscriber) may sell pirated devices/software for decryption
  - To detect such a user
    - Using black-box access to the pirated device/code

Traitor Tracing

- A legitimate user (paid subscriber) may sell pirated devices/software for decryption
  - To detect such a user
    - Using black-box access to the pirated device/code
    - Device may output only if message "interesting" (hence cannot trace if the device is interested only in a hard to guess subset of the message space)

Traitor Tracing

- A legitimate user (paid subscriber) may sell pirated devices/software for decryption
  - To detect such a user
    - Using black-box access to the pirated device/code
    - Device may output only if message "interesting" (hence cannot trace if the device is interested only in a hard to guess subset of the message space)
- Will assume stateless decoder

Traitor Tracing

- A legitimate user (paid subscriber) may sell pirated devices/ software for decryption
  - To detect such a user
    - Using black-box access to the pirated device/code
    - Device may output only if message "interesting" (hence cannot trace if the device is interested only in a hard to guess subset of the message space)
- Will assume stateless decoder
  - Can use "robust watermarks" to handle stateful decoders

Traitor Tracing

- A legitimate user (paid subscriber) may sell pirated devices/software for decryption
  - To detect such a user
    - Using black-box access to the pirated device/code
    - Device may output only if message "interesting" (hence cannot trace if the device is interested only in a hard to guess subset of the message space)
- Will assume stateless decoder
  - Can use "robust watermarks" to handle stateful decoders
- Useful for broadcast encryption, but also considered independently

Traitor Tracing

Traitor Tracing

- A proof-of-concept scheme (with a long ciphertext)

Traitor Tracing

- A proof-of-concept scheme (with a long ciphertext)
  - Encrypt(M) = ($E_{PK1}(M),\ldots,E_{PKn}(M)$)

Traitor Tracing

- A proof-of-concept scheme (with a long ciphertext)
  - Encrypt(M) = ($E_{PK1}(M),\ldots,E_{PKn}(M)$)
  - Trace^D: Feed D encryptions of the form ($E_{PK1}(0),\ldots,E_{PKi-1}(0)$, $E_{PKi}(M)$, … $E_{PKn}(M)$). Let p_i be the probability of D outputting M

Traitor Tracing

- A proof-of-concept scheme (with a long ciphertext)
  - Encrypt(M) = ($E_{PK1}(M),\ldots,E_{PKn}(M)$)
  - $Trace^D$: Feed D encryptions of the form ($E_{PK1}(0),\ldots,E_{PKi-1}(0), E_{PKi}(M), \ldots E_{PKn}(M)$). Let p_i be the probability of D outputting M
    - Determine p_i empirically: relies on sampling "interesting" M

Traitor Tracing

- A proof-of-concept scheme (with a long ciphertext)
  - Encrypt(M) = ($E_{PK1}(M),\ldots,E_{PKn}(M)$)
  - Trace^D: Feed D encryptions of the form ($E_{PK1}(0),\ldots,E_{PKi-1}(0)$, $E_{PKi}(M)$, … $E_{PKn}(M)$). Let p_i be the probability of D outputting M
    - Determine p_i empirically: relies on sampling "interesting" M
    - If $p_i - p_{i-1}$ is large for some i, implicate PK_i

Traitor Tracing

- A proof-of-concept scheme (with a long ciphertext)
  - Encrypt(M) = ($E_{PK1}(M),\ldots,E_{PKn}(M)$)
  - Trace^D: Feed D encryptions of the form ($E_{PK1}(0),\ldots,E_{PKi-1}(0), E_{PKi}(M), \ldots E_{PKn}(M)$). Let p_i be the probability of D outputting M
    - Determine p_i empirically: relies on sampling "interesting" M
    - If $p_i - p_{i-1}$ is large for some i, implicate PK_i
    - Note: D may have multiple keys, and may check consistency of decryptions before outputting a message

Traitor Tracing

- A proof-of-concept scheme (with a long ciphertext)
  - Encrypt(M) = ($E_{PK1}(M),\ldots,E_{PKn}(M)$)
  - $Trace^D$: Feed D encryptions of the form ($E_{PK1}(0),\ldots,E_{PKi-1}(0)$, $E_{PKi}(M)$, ... $E_{PKn}(M)$). Let p_i be the probability of D outputting M
    - Determine p_i empirically: relies on sampling "interesting" M
    - If $p_i - p_{i-1}$ is large for some i, implicate PK_i
    - Note: D may have multiple keys, and may check consistency of decryptions before outputting a message
  - Use with subset cover based broadcast encryption? Can be used for "subset tracing", but not satisfactory if D decrypts only when, say, the subset that will be traced is large

Traitor Tracing

Traitor Tracing

- Traitor tracing from “Set-hiding Broadcast Encryption” for intervals

Traitor Tracing

- Traitor tracing from “Set-hiding Broadcast Encryption” for intervals
  - For intervals: Allows broadcasting to sets of the form {i,i+1,...,n}

Traitor Tracing

- Traitor tracing from "Set-hiding Broadcast Encryption" for intervals
  - For intervals: Allows broadcasting to sets of the form {i,i+1,...,n}
  - Set to which the encryption is addressed is hidden (i.e., i is hidden), except as revealed by decrypting using the keys possessed by the adversary

Traitor Tracing

- Traitor tracing from "Set-hiding Broadcast Encryption" for intervals
  - For intervals: Allows broadcasting to sets of the form {i,i+1,...,n}
  - Set to which the encryption is addressed is hidden (i.e., i is hidden), except as revealed by decrypting using the keys possessed by the adversary
    - In particular, encryption to {i,..,n} and {i+1,...,n} distinguishable only if adversary gets key for user i

Traitor Tracing

- Traitor tracing from "Set-hiding Broadcast Encryption" for intervals
  - For intervals: Allows broadcasting to sets of the form {i,i+1,...,n}
  - Set to which the encryption is addressed is hidden (i.e., i is hidden), except as revealed by decrypting using the keys possessed by the adversary
    - In particular, encryption to {i,..,n} and {i+1,...,n} distinguishable only if adversary gets key for user i
- In the traitor-tracing scheme, encryption will use the broadcast encryption with i=1 (i.e., for the entire set of users) and tracing algorithm will use encryptions to all intervals

Traitor Tracing

- Traitor tracing from "Set-hiding Broadcast Encryption" for intervals
  - For intervals: Allows broadcasting to sets of the form {i,i+1,...,n}
  - Set to which the encryption is addressed is hidden (i.e., i is hidden), except as revealed by decrypting using the keys possessed by the adversary
    - In particular, encryption to {i,..,n} and {i+1,...,n} distinguishable only if adversary gets key for user i
- In the traitor-tracing scheme, encryption will use the broadcast encryption with i=1 (i.e., for the entire set of users) and tracing algorithm will use encryptions to all intervals
- Scheme with $O(\sqrt{n})$ ciphertext, using bilinear pairing [BSW'06]

Group Key Assignment

Group Key Assignment

- A.k.a key distribution for dynamic conferences

Group Key Assignment

- A.k.a key distribution for dynamic conferences
- A center distributes private information to each party (and possibly publishes additional public information)

Group Key Assignment

- A.k.a key distribution for dynamic conferences
- A center distributes private information to each party (and possibly publishes additional public information)
- Each party should be able to derive the key for any group containing it, using its private information and public information alone

Group Key Assignment

- A.k.a key distribution for dynamic conferences
- A center distributes private information to each party (and possibly publishes additional public information)
- Each party should be able to derive the key for any group containing it, using its private information and public information alone
- Security requirement: a set of colluding parties outside a group should not be able to distinguish the key for the group from a random key

Group Key Assignment

- A.k.a key distribution for dynamic conferences
- A center distributes private information to each party (and possibly publishes additional public information)
- Each party should be able to derive the key for any group containing it, using its private information and public information alone
- Security requirement: a set of colluding parties outside a group should not be able to distinguish the key for the group from a random key
  - May impose an upperbound on the number of colluding parties

Group Key Assignment

Group Key Assignment

- A perfectly secure scheme [Blundo et al. '92]

Group Key Assignment

- A perfectly secure scheme [Blundo et al. '92]
- Symmetric polynomial: $P(x_1,...,x_t) = P(x_{\pi(1)},...,x_{\pi(t)})$ for any permutation π

Group Key Assignment

- A perfectly secure scheme [Blundo et al. '92]
- Symmetric polynomial: $P(x_1,\ldots,x_t) = P(x_{\pi(1)},\ldots,x_{\pi(t)})$ for any permutation π
  - i.e. $a_{d1\ldots dt} = a_{\pi(d1)\ldots\pi(dt)}$ for all π, where $a_{d1\ldots dt}$ is the coefficient of $x_1^{d1}x_2^{d2}\ldots x_t^{dt}$

Group Key Assignment

- A perfectly secure scheme [Blundo et al. '92]
- Symmetric polynomial: $P(x_1,\ldots,x_t) = P(x_{\pi(1)},\ldots,x_{\pi(t)})$ for any permutation π
  - i.e. $a_{d1\ldots dt} = a_{\pi(d1)\ldots\pi(dt)}$ for all π, where $a_{d1\ldots dt}$ is the coefficient of $x_1^{d1}x_2^{d2}\ldots x_t^{dt}$
- Key for the group $(j_1,\ldots,j_t)$ will be $P(j_1,\ldots,j_t)$. Each user j will have the (t-1)-variate polynomial $f_i(x_1,\ldots,x_{t-1})$ defined as $P(x_1,\ldots,x_{t-1}, j)$

Group Key Assignment

- A perfectly secure scheme [Blundo et al. '92]
- Symmetric polynomial: $P(x_1,\ldots,x_t) = P(x_{\pi(1)},\ldots,x_{\pi(t)})$ for any permutation π
  - i.e. $a_{d1\ldots dt} = a_{\pi(d1)\ldots\pi(dt)}$ for all π, where $a_{d1\ldots dt}$ is the coefficient of $x_1^{d1}x_2^{d2}\ldots x_t^{dt}$
- Key for the group $(j_1,\ldots,j_t)$ will be $P(j_1,\ldots,j_t)$. Each user j will have the (t-1)-variate polynomial $f_i(x_1,\ldots,x_{t-1})$ defined as $P(x_1,\ldots,x_{t-1}, j)$
  - If P is a random symmetric polynomial of degree k in each variable, then the scheme is k-secure (i.e., for up to k users outside the group, the group key is perfectly random)

Group Key Agreement

Group Key Agreement

- Recall 3-party extension of Diffie-Hellman key exchange [Joux'00]

Group Key Agreement

- Recall 3-party extension of Diffie-Hellman key exchange [Joux'00]
  - Single round (of broadcasts), using bilinear pairings, under DBDH

Group Key Agreement

- Recall 3-party extension of Diffie-Hellman key exchange [Joux'00]
  - Single round (of broadcasts), using bilinear pairings, under DBDH
- How about larger groups?

Group Key Agreement

- Recall 3-party extension of Diffie-Hellman key exchange [Joux'00]
  - Single round (of broadcasts), using bilinear pairings, under DBDH
- How about larger groups?
- 2-round, based on DDH [Burmester-Desmedt'94]

Group Key Agreement

- Recall 3-party extension of Diffie-Hellman key exchange [Joux'00]
  - Single round (of broadcasts), using bilinear pairings, under DBDH
- How about larger groups?
- 2-round, based on DDH [Burmester-Desmedt'94]
  - Each player i chooses r_i and broadcasts $z_i = g^{r_i}$

Group Key Agreement

- Recall 3-party extension of Diffie-Hellman key exchange [Joux'00]
  - Single round (of broadcasts), using bilinear pairings, under DBDH
- How about larger groups?
- 2-round, based on DDH [Burmester-Desmedt'94]
  - Each player i chooses r_i and broadcasts $z_i = g^{r_i}$
  - Each player i broadcasts $X_i = (z_{i+1}/z_{i-1})^{r_i}$

Group Key Agreement

- Recall 3-party extension of Diffie-Hellman key exchange [Joux'00]
  - Single round (of broadcasts), using bilinear pairings, under DBDH
- How about larger groups?
- 2-round, based on DDH [Burmester-Desmedt'94]
  - Each player i chooses r_i and broadcasts $z_i = g^{r_i}$
  - Each player i broadcasts $X_i = (z_{i+1}/z_{i-1})^{r_i}$
  - Key $K_i = z_{i-1}^{n.r_i} . X_i^{n-1} . X_{i+1}^{n-2} \ldots X_{i-3}^2 . X_{i-2} = g^{r_1.r_2 + r_2.r_3 + \ldots + r_n.r_1}$

Group Key Agreement

- Recall 3-party extension of Diffie-Hellman key exchange [Joux'00]
  - Single round (of broadcasts), using bilinear pairings, under DBDH
- How about larger groups?
- 2-round, based on DDH [Burmester-Desmedt'94]
  - Each player i chooses r_i and broadcasts $z_i = g^{ri}$
  - Each player i broadcasts $X_i = (z_{i+1}/z_{i-1})^{ri}$
  - Key $K_i = z_{i-1}^{n.ri} . X_i^{n-1} . X_{i+1}^{n-2} \ldots X_{i-3}^{2} . X_{i-2} = g^{r1.r2 + r2.r3 + \ldots + rn.r1}$
- Can convert to authenticated group key agreement [KY'03]

Today

Today

- Broadcast encryption

Today

- Broadcast encryption
- Traitor Tracing

Today

- Broadcast encryption
- Traitor Tracing
- Group Key Assignment (a.k.a key distribution for dynamic conferences)

Today

- Broadcast encryption
- Traitor Tracing
- Group Key Assignment (a.k.a key distribution for dynamic conferences)
- Group Key Agreement (a.k.a group key exchange)