UNIVERSITY OF TWENTE.

SECURE DATA MANAGEMENT

Multiple Writer – Multiple Reader Example

Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-user System

Source:
Hwang, Y., & Lee, P. (2007). Public key encryption with conjunctive keyword search and its extension to a multi-user system. *Pairing-Based Cryptography–Pairing 2007*, 2-22.

OUTLINE

- Context of mPECK scheme
- Generic PECK scheme and its adversarial models
- Limitation of PECK
- Generic mPECK scheme and its adversarial model
- Concrete construction of mPECK using ElGamal Multi-receiver Encryption Scheme

Context of mPECK SCHEME

mPECK: **m**ulti-user **P**ublic-**K**ey **E**ncryption **C**onjunctive **K**eyword Search

- Data of multiple users is stored encrypted on a minimally trusted server
- Multiple users can make queries for conjunctive keyword searches
  - e.g. ‘documents containing keyword1 AND keyword2 AND keyword3’
- Simplifications
  - Same keyword never appears in two different keyword fields
    - Example of field names in emails: ‘to’, ‘from’, ‘subject’, ‘time’, etc.
    - Embed field names within keywords (e.g. concatenate: ‘from.Alice’, ‘subject.report’)
  - Every keyword field is defined for every document
  - ‘to.Null’ for a field that doesn’t have a valid keyword
  - Don’t mark field names in keywords!

PECK SECURITY MODEL: DLDH ASSUMPTION

DECISION LINEAR DIFFIE-HELLMAN ASSUMPTION

*PPT: **P**robabilistic **P**olynomial **T**ime

- Let $g_1, g_2, g_3 \in G_1$ and $x, y, z \in \mathbb{Z}_q$

  Given $(g_1, g_2, g_3, g_1^x, g_2^y, g_3^z)$ decide whether $z =^? x + y$.

- The advantage of an algorithm $\mathcal{A}$ in solving DLDH in G_1 is:

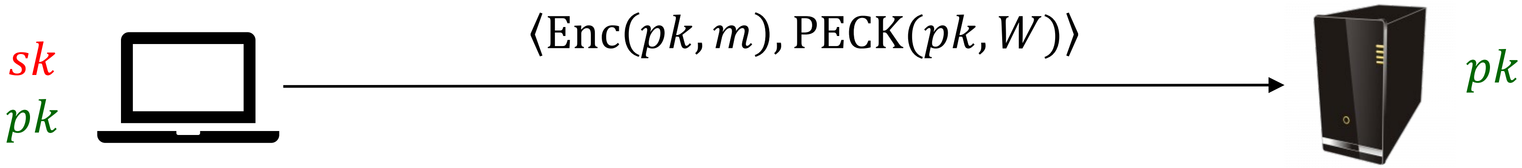
  $Adv(DLDH_{\mathcal{A}}) = \left|\Pr\left[\mathcal{A}\left(g_1, g_2, g_3, g_1^x, g_2^y, g_3^{x+y}\right) = 1\right] - \Pr\left[\mathcal{A}\left(g_1, g_2, g_3, g_1^x, g_2^y, g_4\right) = 1\right]\right|$

  where $g_1, g_2, g_3, g_4 \in_R G_1$ and $x, y, z \in_R \mathbb{Z}_q$

- The DLDH assumption holds if no PPT algorithm has a non-negligible advantage in solving the DLDH problem in G_1.

GENERIC PECK SCHEME

Overview - Upload



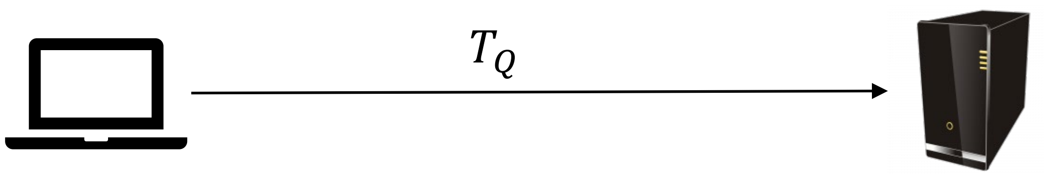
Stores $\langle \mathrm{Enc}(pk, m), \mathrm{PECK}(pk, W) \rangle$

$W = \{w_1, \dots, w_\ell\}$ set of all possible searchable keywords

GENERIC PECK SCHEME

Overview - Query

sk
pk



T_Q

pk
$\langle \text{Enc}(pk, m), \text{PECK}(pk, W) \rangle$

$\text{Test}\left(pk, \text{PECK}(pk, W), T_Q\right) \rightarrow 1$ or 0
If 1 sends $\text{Enc}(pk, m)$

T_Q query trapdoor
$Q = \left\{I_1, \dots, I_N, w_{I_1}, \dots, w_{I_N}\right\}$ conjunctive keyword query, $\left\{w_{I_1}, \dots, w_{I_N}\right\} \subseteq W$

GENERIC PECK SCHEME (1)

PECK scheme consists of four polynomial time algorithms:

- **KeyGen**
  - Input
    - 1^k: security parameter
  - Output
    - $params$: system's parameters
    - (pk, sk): public/private keypair

- **PECK:** "Run by the **sender (data owner)** to encrypt a keyword set"
  - Input
    - pk: public key
    - $W = \{w_1, \dots, w_\ell\}$: keyword set
  - Output
    - S: a searchable keyword encryption of W (under public key pk)

GENERIC PECK SCHEME (2)

PECK scheme consists of four polynomial time algorithms:

- **Trapdoor:** "Run by the **sender** to enable the **server** to retrieve the keywords of S "
  - Input
    - sk: secret key
    - Query: $Q = \{I_1, \ldots, I_m, w_{I_1}, \ldots, w_{I_m}\}$ for $m \leq \ell$ where I_i is an index of a location of w_{I_i}
  - Output
    - T_Q: a trapdoor for the conjunctive search of the given keyword query Q

- **Test:** "Run by the **server** to search the documents with the keywords of a trapdoor T_Q"
  - Input
    - pk, S, T_Q
  - Output
    - '1' if S includes Q, and '0' otherwise

ADVERSARIAL MODEL FOR PECK: CHOSEN KEYWORD ATTACK

Semantic security against two variants of chosen keyword attacks

- IND-**CC**-CKA (indistinguishability of **ciphertext** from **ciphertext**)
- IND-**CR**-CKA (indistinguishability of **ciphertext** from **random**)

IND-CC-CKA GAME

Challenger | **Adversary**

Setup

$(params, pk, sk) \leftarrow KeyGen(1^k)$ $\xrightarrow{params, pk}$

Probing 1

- Chooses keyword sets $Q_1, \dots, Q_d$
- Queries $\mathcal{O}^{\text{Trapdoor}}$ and gets trapdoors:

$$T_{Q_1} \coloneqq Trapdoor(sk, Q_1)$$
$$\vdots$$
$$T_{Q_d} \coloneqq Trapdoor(sk, Q_d)$$

Challenge

Chooses $\beta \in_R \{0,1\}$, and
Sets $S_\beta = PECK(pk, W_\beta)$

$\xleftarrow{W_0, W_1}$

$\xrightarrow[S_\beta]{}$

Chooses W_0, W_1 target keywords

Probing 2

- Chooses keyword sets $Q_{d+1}, \dots, Q_N$
- Queries $\mathcal{O}^{\text{Trapdoor}}$ and gets trapdoors:

$$T_{Q_{d+1}} \coloneqq Trapdoor(sk, Q_{d+1})$$
$$\vdots$$
$$T_{Q_N} \coloneqq Trapdoor(sk, Q_N)$$

None of $T_{Q_1}, \dots, T_{Q_N}$ should distinguish W_0 from W_1!

Guess

Outputs $\beta' \in \{0,1\}$
The Adversary wins if $\beta' = \beta$

<u>IND-CC-CKA Security definition</u> PECK is secure, if for any t-time adversary $\mathcal{A}$ who makes at most q_t trapdoor queries, we have:

$$Adv_{\text{PECK}, \mathcal{A}}^{\text{IND-CC-CKA}}(1^k) = \left|\Pr[\beta' = \beta] - \frac{1}{2}\right| < \epsilon$$

IND-CR-CKA GAME

Challenger | **Adversary**

Setup

$(params, pk, sk) \leftarrow KeyGen(1^k)$

$$\xrightarrow{\quad params, pk \quad}$$

Probing 1

- Chooses keyword sets $Q_1, \dots, Q_d$
- Queries $\mathcal{O}^{\text{Trapdoor}}$ and gets trapdoors:

$$T_{Q_1} \coloneqq Trapdoor(sk, Q_1)$$
$$\vdots$$
$$T_{Q_d} \coloneqq Trapdoor(sk, Q_d)$$

Challenge

Chooses W_0 target keyword

$$\xleftarrow{\quad W_0 \quad}$$

Chooses random keyword
W_1 and $\beta \in_R \{0,1\}$
Sets $S_\beta = PECK(pk, W_\beta)$

$$\xrightarrow[\quad S_\beta, W_1 \quad]{}$$

Probing 2

- Chooses keyword sets $Q_{d+1}, \dots, Q_N$
- Queries $\mathcal{O}^{\text{Trapdoor}}$ and gets trapdoors:

$$T_{Q_{d+1}} \coloneqq Trapdoor(sk, Q_{d+1})$$
$$\vdots$$
$$T_{Q_N} \coloneqq Trapdoor(sk, Q_N)$$

Guess

Outputs $\beta' \in \{0,1\}$
The Adversary wins if $\beta' = \beta$

None of $T_{Q_1}, \dots, T_{Q_N}$ should distinguish W_0 from W_1!

<u>IND-CR-CKA Security definition</u> PECK is secure, if for any t-time adversary $\mathcal{A}$ who makes at most q_t trapdoor queries, we have:

$$Adv_{\text{PECK},\ \mathcal{A}}^{\text{IND-CR-CKA}}(1^k) = \left|\Pr[\beta' = \beta] - \frac{1}{2}\right| < \epsilon$$

Limitation of PECK

- Situation

  Suppose that a data owner wants to share its data with n different users

- In **PECK**, the encrypted data is stored in form $\langle Enc(pk_i, m), PECK(pk_i, W)\rangle$
  - He has to upload to the server

$$\langle Enc(pk_1, m), PECK(pk_1, W)\rangle$$
$$\vdots$$
$$\langle Enc(pk_n, m), PECK(pk_n, W)\rangle$$

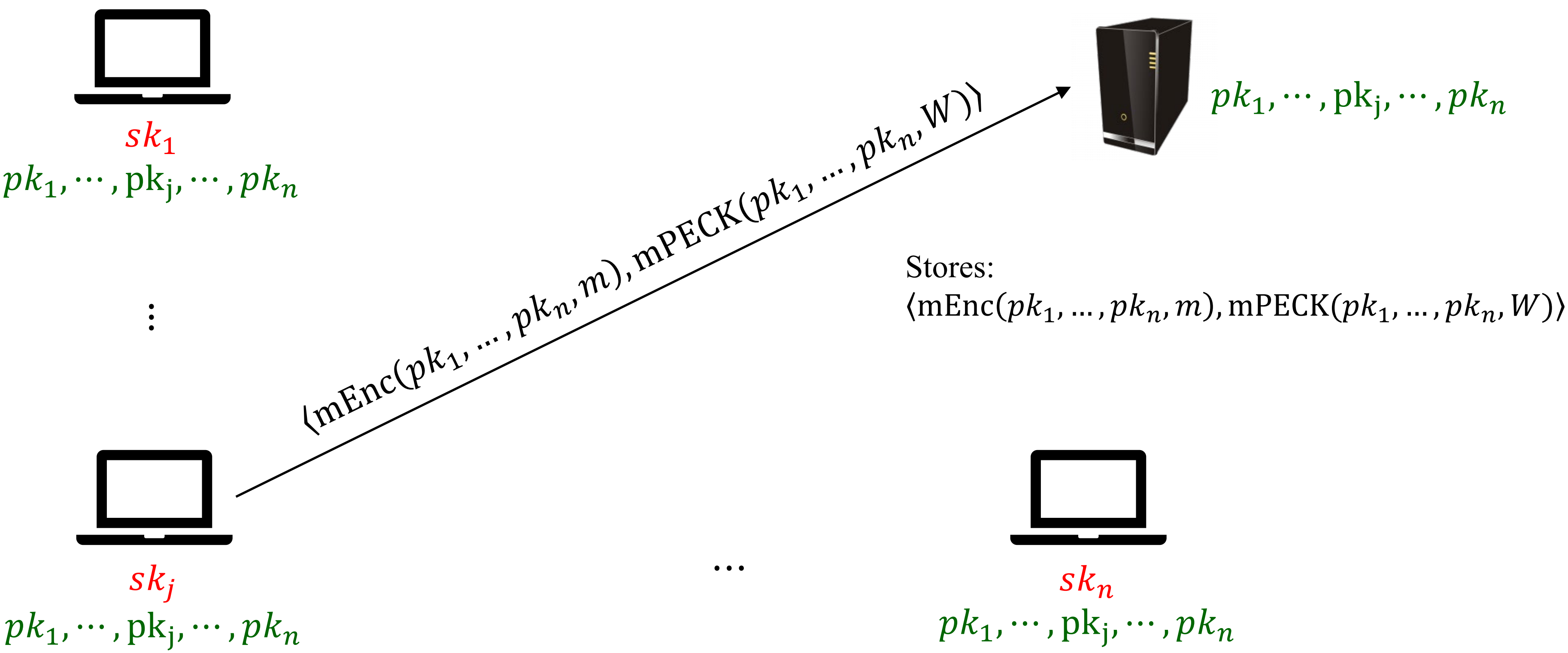
  - The server stores them separately

mPECK solves this limitation

GENERIC mPECK SCHEME

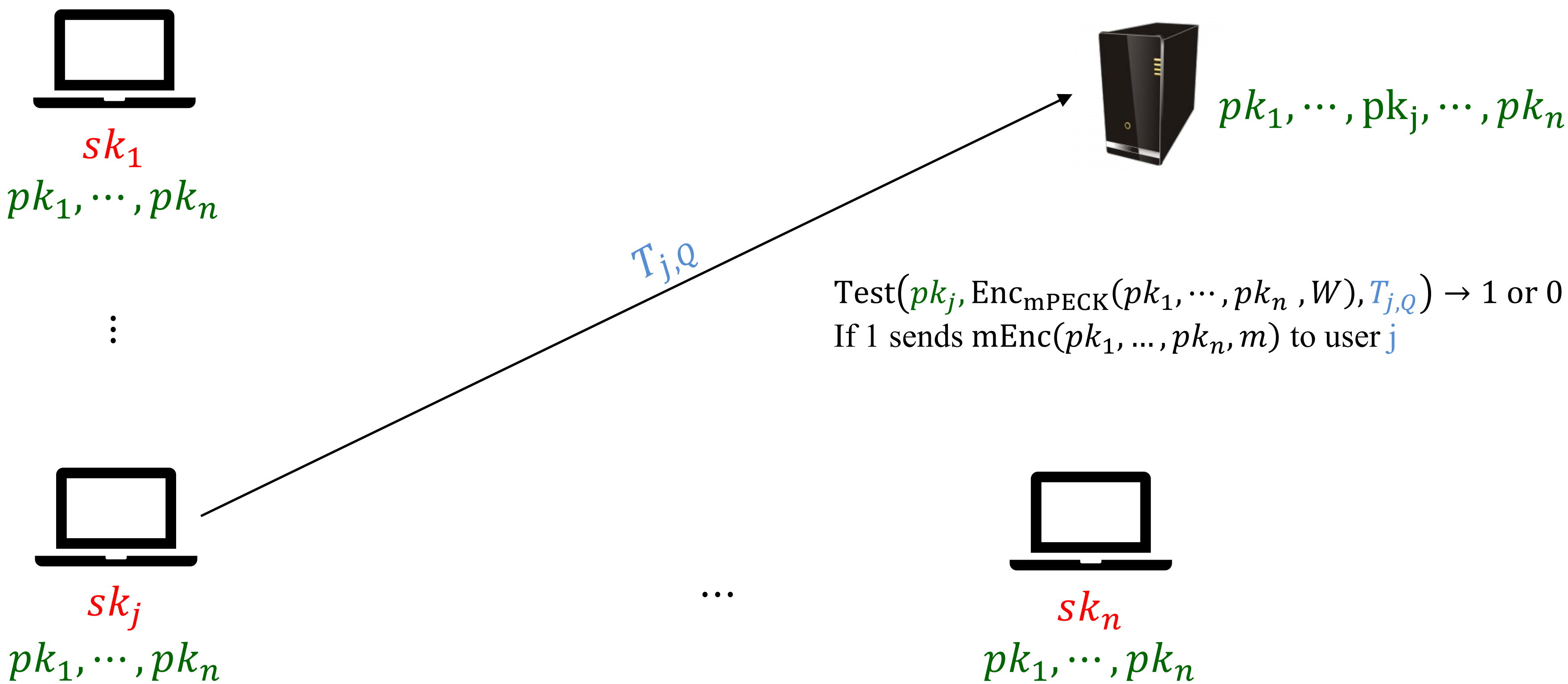
Overview – Upload

*__Recall:__ mEnc($pk_1, \ldots, pk_n, \ \cdot$)
Multi-receiver PKE: Encrypt in a way that allows several users to decrypt



GENERIC mPECK SCHEME

Overview – Query



GENERIC mPECK SCHEME (1)

mPECK scheme consists of four polynomial time algorithms:

- **KeyGen**
  - Input
    - 1^k: security parameter
  - Output
    - $params$: system's parameters
    - $(pk_1, sk_1), \cdots, (pk_n, sk_n)$: public/private keypairs

- **mPECK:** "Run by the **sender (data owner)** to encrypt a keyword set"
  - Input
    - $pk_1, \cdots, pk_n$: public keys
    - $W = \{w_1, \dots, w_\ell\}$: keyword set
  - Output
    - S: a searchable keyword encryption of W (under public keys $pk_1, \cdots, pk_n$)

GENERIC mPECK SCHEME (2)

mPECK scheme consists of four polynomial time algorithms:

- **Trapdoor:** “Run by the **sender** to enable the **server** to retrieve the keywords of S
  - Input
    - sk_j: secret key
    - Query: $Q = \{I_1, \dots, I_m, w_{I_1}, \dots, w_{I_m}\}$ for $m \le \ell$ where I_i is an index of a location of w_{I_i}
  - Output
    - $\boldsymbol{T_{j,Q}}$: a trapdoor for the conjunctive search of the given keyword query Q

- **Test:** “Run by the **server** to search the documents with the keywords of a trapdoor $T_{j,Q}$”
  - Input
    - $pk_j, S, T_{j,Q}$
  - Output
    - ‘1’ if S includes Q, and ‘0’ otherwise

IND-mCR-CKA GAME

Challenger | **Adversary**

Setup

$\begin{pmatrix} params, pk_1, \ldots, pk_n \\ , sk_1, \ldots, sk_n \end{pmatrix} \leftarrow KeyGen(1^k)$ $\xrightarrow{params, pk_1, \ldots, pk_n}$

Probing 1

- Chooses a fixed user j
- Chooses keyword sets $Q_1, \ldots, Q_d$
- Queries $\mathcal{O}^{\text{Trapdoor}}$ and gets trapdoors:

$$T_{j,Q_1} \coloneqq Trapdoor(sk_j, Q_1)$$
$$\vdots$$
$$T_{j,Q_d} \coloneqq Trapdoor(sk_j, Q_d)$$

Challenge

Chooses random keyword
W_1 and $\beta \in_R \{0,1\}$
Sets $S_\beta = PECK(pk_1, \ldots, pk_n, W_\beta)$

$\xleftarrow{W_0}$

$\xrightarrow[S_\beta, W_1]{}$

Chooses W_0 target keyword

None of $T_{Q_1}, \ldots, T_{Q_N}$ should distinguish W_0 from W_1!

Probing 2

- Chooses keyword sets $Q_{d+1}, \ldots, Q_N$
- Queries $\mathcal{O}^{\text{Trapdoor}}$ and gets trapdoors:

$$T_{j,Q_{d+1}} \coloneqq Trapdoor(sk_j, Q_{d+1})$$
$$\vdots$$
$$T_{j,Q_N} \coloneqq Trapdoor(sk_j, Q_N)$$

Guess

Outputs $\beta' \in \{0,1\}$
The Adversary wins if $\beta' = \beta$

IND-mCR-CKA Security definition: mPECK is secure, if for any t-time adversary $\mathcal{A}$ who makes at most q_t trapdoor queries, we have:

$$Adv_{\text{mPECK}, \mathcal{A}}^{\text{IND-mCR-CKA}}(1^k) = \left|\Pr[\beta' = \beta] - \frac{1}{2}\right| < \epsilon$$

CONCRETE mPECK SCHEME CONSTRUCTION (1)

- **KeyGen**
  - Input
    - 1^k: security parameter
  - Output
    - $params$
      - $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$
      - $\mathbb{G}_1 = \langle g \rangle$
      - $H_1, H_2: \{0,1\}^{\log(w)} \rightarrow \mathbb{G}_1$ are two different collision-resistant hash functions
    - Public/private keypairs $(pk_1, sk_1), \cdots, (pk_n, sk_n)$
      - $(pk_i, sk_i) = (y_i, x_i)$ where for $i = 1 \dots n$
        - $x_i \in_R \mathbb{Z}_p^*$ and $y_i = g^{x_i}$

CONCRETE mPECK SCHEME CONSTRUCTION (2)

- **<u>mPECK:</u>** “Run by the **sender** to encrypt a keyword set”
  - Input
    - $pk_1, \cdots, pk_n$: public keys ($pk_i = y_i$)
    - $W = \{w_1, \ldots, w_\ell\}$: keyword set
  - Algorithm
    - Compute: $h_i = H_1(w_i)$ and $f_i = H_2(w_i)$ for all $w_i \in W$
    - Select: $\boldsymbol{s}, \boldsymbol{r} \in_R \mathbb{Z}_p^*$
    - Compute
      - $A = g^r$ and $B_j = y_j^s$ for all $1 \leq j \leq n$
      - $C_i = h_i^r f_i^s$ for all $1 \leq i \leq \ell$
    - Output
      - $S = \langle A, B_1, \ldots, B_n, C_1, \ldots, C_\ell \rangle$

CONCRETE mPECK SCHEME CONSTRUCTION (3)

- **Trapdoor:** "Run by the **sender** to enable the **server** to retrieve the keywords of S "
  - Input
    - $sk_j = x_j$: secret key
    - Query: $Q = \{I_1, \ldots, I_m, w_{I_1}, \ldots, w_{I_m}\}$ for $m \leq \ell$ where I_i is an index of a location of w_{I_i}
  - Algorithm
    - Select: $\boldsymbol{t} \in_R \mathbb{Z}_p^*$
    - Compute
      - $T_{j,Q_1} = g^t$
      - $T_{j,Q_2} = \left(h_{I_1} \ldots h_{I_m}\right)^t$ where $h_{I_i} = H_1(w_{I_i})$
      - $T_{j,Q_3} = \left(f_{I_1} \ldots f_{I_m}\right)^{\frac{t}{x_j}}$ where $f_{I_i} = H_2(w_{I_i})$
  - Output
    - $T_{j,Q} = \left(T_{j,Q_1}, T_{j,Q_2}, T_{j,Q_3}, I_1, \ldots, I_m\right)$

CONCRETE mPECK SCHEME CONSTRUCTION (4)

*__Recall:__ $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$

- **Test:** "Run by the **server** to search the documents with the keywords of a trapdoor $T_{j,Q}$"
  - Input
    - $pk_j = y_j$
    - $S = \langle A, B_1, \dots, B_n, C_1, \dots, C_\ell \rangle$
    - $T_{j,Q} = \left(T_{j,Q_1}, T_{j,Q_2}, T_{j,Q_3}, I_1, \dots, I_m\right)$
  - Check
    - Output '1' if the equation holds and '0' otherwise
    - $\hat{e}\left(T_{j,Q_1}, \prod_{i=1}^{m} C_{I_i}\right) \stackrel{?}{=} \hat{e}\left(A, T_{j,Q_2}\right) \cdot \hat{e}\left(B_j, T_{j,Q_3}\right)$

*__Remember:__
$y_j = g^{x_j}$ and $B_j = y_j^s$
$B_j = g^{s \cdot x_j}$

Why does it work?

✓ $\hat{e}\left(T_{j,Q_1}, \prod_{i=1}^{m} C_{I_i}\right) = \hat{e}\left(g^{t}, \prod_{i=1}^{m}(h_{I_i}^{r} f_{I_i}^{s})\right) = \hat{e}\left(g^{t}, \prod_{i=1}^{m} h_{I_i}^{r}\right) \cdot \hat{e}\left(g^{t}, \prod_{i=1}^{m} f_{I_i}^{s}\right) = \underbrace{\hat{e}\left(g, \prod_{i=1}^{m} h_{I_i}\right)^{t \cdot r} \cdot \hat{e}\left(g, \prod_{i=1}^{m} f_{I_i}\right)^{t \cdot s}}$

$$=$$

✓ $\hat{e}\left(A, T_{j,Q_2}\right) \cdot \hat{e}\left(B_j, T_{j,Q_3}\right) = \hat{e}\left(g^{r}, \prod_{i=1}^{m} h_{I_i}^{t}\right) \cdot \hat{e}\left(g^{s \cdot \cancel{x_j}}, \prod_{i=1}^{m} f_{I_i}^{\frac{t}{\cancel{x_j}}}\right) = \overbrace{\hat{e}\left(g, \prod_{i=1}^{m} h_{I_i}\right)^{t \cdot r} \cdot \hat{e}\left(g, \prod_{i=1}^{m} f_{I_i}\right)^{t \cdot s}}$

ElGamal Type Multi-receiver Encryption Scheme

In mPECK, encrypted data is uploaded as:

$$\boxed{(\boldsymbol{E}, \boldsymbol{S}) \leftarrow \langle mEnc(y_1, \dots, y_n, \boldsymbol{msg}), mPECK(y_1, \dots, y_n, W)\rangle}$$

- The Data owner uses the same random $s, r \in_R \mathbb{Z}_p^*$ used to generate $\mathbf{S} := mPECK(y_1, \dots, y_n, W)$ to encrypt message $\boldsymbol{msg}$ via ElGamal Multi-receiver Encryption Scheme

- Encrypts $\boldsymbol{msg}$ as follows
  - $\boldsymbol{E} := mEnc(y_1, \dots, y_n, msg) = H_0(\hat{e}(g, g)^{rs}) \oplus msg$
  - where
    - $H_0: \mathbb{G}_2 \rightarrow \mathcal{M}$ is another one-way hash function
    - $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$
    - $\mathbb{G}_1 = \langle g \rangle$

ElGamal Type Multi-receiver Encryption Scheme

Stored encrypted data of the form:

$$\boxed{(\boldsymbol{E}, \boldsymbol{S}) \leftarrow \langle mEnc(y_1, \ldots., y_n, \boldsymbol{msg}), mPECK(y_1, \ldots, y_n, W) \rangle}$$

***Remember:**
$A = g^r$
$B_j = y_j^s = g^{s \cdot x_j}$
$\boldsymbol{E} = H_0(\hat{e}(g,g)^{rs}) \oplus msg$

- A user u_j makes a query by sending a trapdoor to the server
- The server returns (A, B_j, E)
- User u_j uses his private key $\boldsymbol{x_j}$ to decrypt $\boldsymbol{E}$ as follow
  - Computes: $X_j = H_0\left(\hat{e}(A, B_j)^{1/x_j}\right) = H_0\left(\hat{e}(g^r, g^{s \cdot x_j})^{1/x_j}\right)$
    $$= H_0\left(\hat{e}(g,g)^{\frac{r \cdot s \cdot \cancel{x_j}}{\cancel{x_j}}}\right) = H_0(\hat{e}(g,g)^{rs})$$
  - Outputs: $mDec(x_j, E, B_j) = E \oplus X_j = \cancel{H_0(\hat{e}(g,g)^{rs})} \oplus msg \oplus \cancel{H_0(\hat{e}(g,g)^{rs})} = msg$
- If user u_j is a legitimate, he will output "msg" otherwise random