

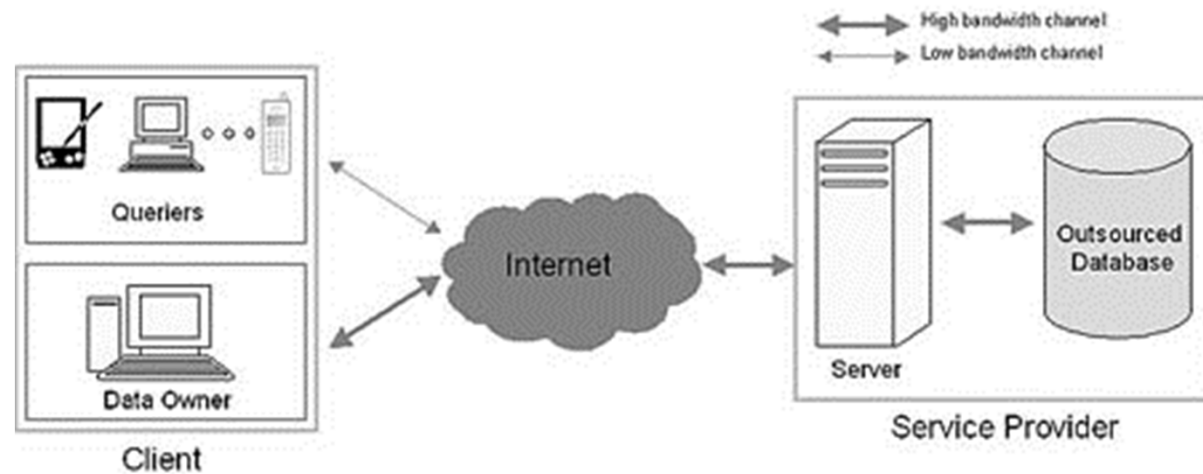
The slide features a decorative background with a yellow vertical bar on the left, a green horizontal bar at the bottom, and a light green horizontal bar at the very bottom. The main title is centered in the green bar.

CLOUD ENABLED DATA SHARING MODEL

Nguyen Thanh Hung

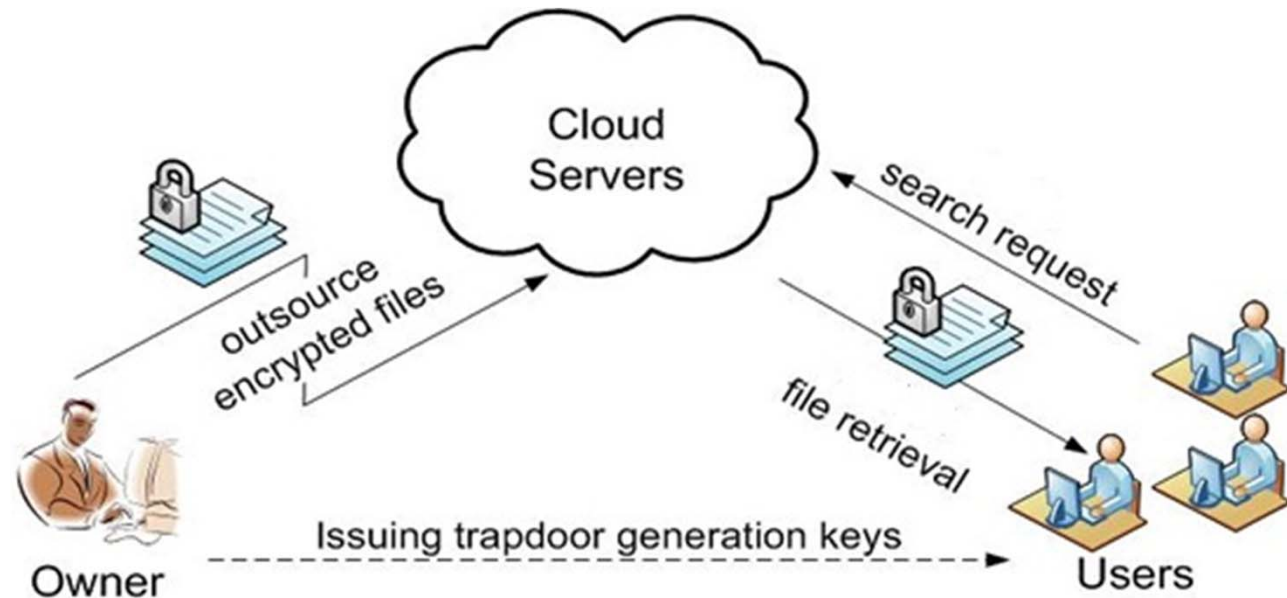
INTRODUCTION

① Cloud Computing as an outsourced database



DATA PRIVACY FOR OUTSOURCED DATABASE

- ⊙ Data are stored off-premises → cannot trust cloud server.
- ⊙ Need to encrypt the file and the encrypted database must be searchable.

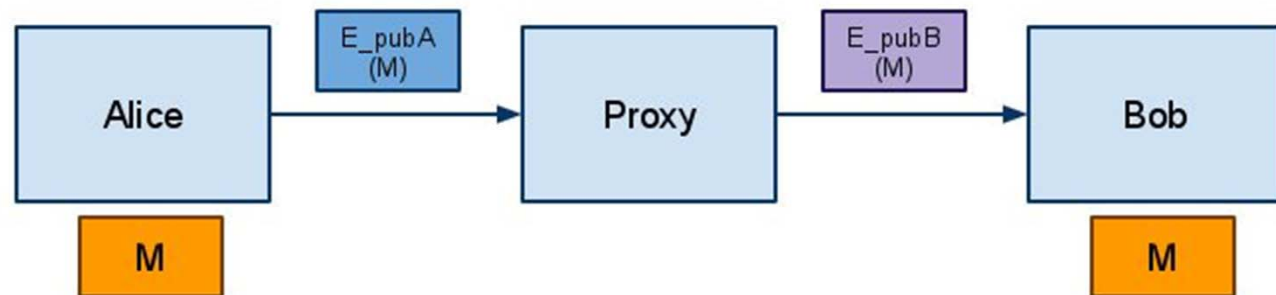




SINGLE USER VS. MULTIUSER

- ◎ **Single user:** can use either symmetric key system or public key system for storing and querying data.
- ◎ **Multi-user :** cannot use symmetric key system as key needs to be shared.
- Use public key system and must satisfy following properties:
 - There is **no key shared** between users.
 - If a user leaves the system, he cannot get access to the data anymore.

PROXY RE-ENCRYPTION





SYSTEM MODEL

- n users : U_1, U_2, \dots, U_n : storing and querying data
- a database server
- m proxies P_1, P_2, \dots, P_m :
 - helping users encrypt, decrypt and query
- a key manager (KM) : generating keys

USERS, KEY MANAGER AND PROXY

- ⊙ Key manager generates a master storage key s for encrypting and decrypting, a master query key q querying data.
- ⊙ Each user U_i has a storage key S_i^u and a query key q_i^u
- ⊙ For each proxy P_j , it stores the corresponding storage-query key $S_{i,j}^u, q_{i,j}^u$ for each user U_i

DATA FORMAT

- ⊙ Data stored as tables in the database. T is the plaintext table, T' is the encrypted table (r rows, c columns).
- ⊙ $T'_{x,y}$ is the cell at row x , column y , comprises of 2 parts:
 - ⊙ $T'_{x,y}(1) = E_S(T_{x,y})$: encrypted using master storage key s (can be decrypted)
 - ⊙ $T'_{x,y}(2) = f_q(T_{x,y})$: encrypted using master query key (acts as a trapdoor for querying data)



TRUST AND ATTACK MODEL

- ⊙ Key manager is fully trusted
 - ⊙ Can be kept off line most of the time
- ⊙ Database server, proxies, users can be modeled as semi-honest.
 - ⊙ Follow protocol
 - ⊙ Can collude with each other.

TECHNICAL PRELIMINARY

- ⊙ Bilinear Map: G_1 and G_2 are 2 groups of prime order p and a bilinear map $e: G_1 \times G_1 \rightarrow G_2$ between them. It satisfies following properties:
 - ⊙ Computable: given $g, h \in G_1$, there is a polynomial time algorithms to compute $e(g, h) \in G_2$
 - ⊙ Bilinear: for any $x, y \in \mathbb{Z}_p^*$, we have $e(g^x, g^y) = e(g, g)^{xy}$
 - ⊙ Non-degenerate: if g is a generator in G_1 then $e(g, g)$ is a generator in G_2
- ⊙ G_1 is a Gap-Diffie-Hellman group where the Decisional Diffie-Hellman problem (DDH) is easy while the Computational Diffie-Hellman problem (CDH) is still hard

SYSTEM SETUP

- ⊙ $SysGen(1^k)$ be the master key generation algorithm.
- ⊙ The key manager given the security parameter k sets up G_1, G_2 be the 2 cyclic groups of large prime order p and the bilinear map e between them. g is the generator of G_1 and $w = e(g, g)$ is the generator of G_2 .
- ⊙ KM chooses a random master storage key s and master query key q ($s, q \in_R \mathbb{Z}_p^*$) and keeps them secret.
- ⊙ KM publicizes (G_1, G_2, p, e, g, w) .

USER ENROLLMENT

⊙ To enroll new user U_i , key manager does:

⊙ Split the master storage key s :

$$s = s_i^U + s_{i,1}^P + s_{i,2}^P + \dots + s_{i,m}^P = s_i^U + \sum_{j=1}^m s_{i,j}^P$$

⊙ Split the master query key q :

$$\begin{aligned} q &= q_i^U \times q_i^P = q_i^U \times (q_{i,1}^P + q_{i,2}^P + \dots + q_{i,m}^P) \\ &= q_i^U \times \sum_{j=1}^m q_{i,j}^P \end{aligned}$$

s_i^U, q_i^U : key shares hold by user U_i

$s_{i,j}^P, q_{i,j}^P$: key shares of proxy P_j corresponding to user U_i

DATA ENCRYPTION

- ⊙ User U_i want to encrypt $T_{x,y}$
- ⊙ To encrypt the first part $T'_{x,y}\langle 1 \rangle$:
 - ⊙ U_i chooses a random number r and compute $c_1 = T_{x,y} w^{rs_i^U}$ and $c_2 = w^r$. He sends (c_1, c_2) to database server and c_2 to all proxies.
 - ⊙ After receiving c_2 , each proxy computes $c'_{2,j} = c_2^{s_{i,j}^P} = w^{rs_{i,j}^P}$ and send $c'_{2,j}$ to the server.
 - ⊙ Server computes:
$$c'_1 = c_1 \times \prod_{j=1}^m c'_{2,j} = T_{x,y} \times w^{rs_i^U} \times \prod_{j=1}^m w^{rs_{i,j}^P}$$
$$= T_{x,y} \times w^{r(s_i^U + \sum_{j=1}^m s_{i,j}^P)} = T_{x,y} \times w^{rs}$$
 - ⊙ It stores $T'_{x,y}\langle 1 \rangle = (c'_1, c_2) = (T_{x,y} \times w^{rs}, w^r)$

DATA ENCRYPTION

- ⊙ To encrypt the second part $T'_{x,y} \langle 2 \rangle$:
- ⊙ U_i chooses a random number r' and computes
$$d_1 = e(H(T_{x,y})^{q_i^U}, g^{r'}) = e(H(T_{x,y}), g)^{r' q_i^U} \text{ and } d_2 = g^{r'}.$$
(with H is a hash function)
- ⊙ The user send d_1 to all proxies and d_2 to the server.
- ⊙ After receiving d_1 , proxy computes:
$$d'_{1,j} = d_1^{q_{i,j}^P} = e(H(T_{x,y}), g)^{r' q_i^U q_{i,j}^P} \text{ and sends it to server.}$$
- ⊙ Server computes:
$$\begin{aligned} d'_1 &= \prod_{j=1}^m d'_{1,j} = \prod_{j=1}^m e(H(T_{x,y}), g)^{r' q_i^U q_{i,j}^P} \\ &= e(H(T_{x,y}), g)^{r' q_i^U (\sum_{j=1}^m q_{i,j}^P)} = e(H(T_{x,y}), g)^{r' q} \end{aligned}$$
- ⊙ It stores: $T'_{x,y} \langle 2 \rangle = (d'_1, d_2) = (e(H(T_{x,y}), g)^{r' q}, g^{r'})$

QUERY

- ⊙ Assume user U_i wants to execute a query:
*Select * from table T where $A_y = v$* (with A_y is column y)
- ⊙ User U_i computes: $t = H(v)^{q_i^U}$ and sends it to all proxies.
- ⊙ Each proxy computes: $t'_j = t^{q_{i,j}^P} = H(v)^{q_i^U q_{i,j}^P}$ and send it to server.
- ⊙ Server computes:
$$t'' = \prod_{j=1}^m t'_j = \prod_{j=1}^m H(v)^{q_i^U q_{i,j}^P}$$
$$= H(v)^{q_i^U (\sum_{j=1}^m q_{i,j}^P)} = H(v)^q$$
- ⊙ Then for each entry $T'_{x,y} \langle 2 \rangle = (d'_1, d_2) = (e(H(T_{x,y}), g)^{r'q}, g^{r'})$ of column j , server computes $d = e(t'', g^{r'})$
- ⊙ The test whether $d = d'_1$. If true, return
 $T'_{x,y} \langle 1 \rangle = (T_{x,y} \times w^{rs}, w^r).$

QUERY

⊙ Correctness:

$$\begin{aligned}d = d'_1 &\Leftrightarrow e(t'', g^{r'}) = e(H(T_{x,y}), g)^{r'q} \\ &\Leftrightarrow e(H(v)^q, g^{r'}) = e(H(T_{x,y}), g)^{r'q} \\ &\Leftrightarrow e(H(v), g)^{r'q} = e(H(T_{x,y}), g)^{r'q} \\ &\Leftrightarrow v = T_{x,y}\end{aligned}$$

DATA DECRYPTION

- ⊙ User U_i want to decrypt $T'_{x,y} \langle 1 \rangle = (T_{x,y} \times w^{rs}, w^r)$.
- ⊙ Server sends $T'_{x,y} \langle 1 \rangle$ to user U_i and w^r to all proxies.
- ⊙ Each proxy P_j computes $f_j = w^{rs_{i,j}^P}$ and sends it to U_i
- ⊙ U_i computes:

$$\begin{aligned}\eta &= w^{rs_i^U} \prod_{j=1}^m f_j = w^{rs_i^U} \prod_{j=1}^m w^{rs_{i,j}^P} \\ &= w^{rs_i^U + r \sum_{j=1}^m s_{i,j}^P} = w^{r(s_i^U + \sum_{j=1}^m s_{i,j}^P)} = w^{rs}\end{aligned}$$

- ⊙ To retrieve the plaintext, he computes:

$$T_{x,y} \times w^{rs} \times \eta^{-1} = T_{x,y} \times w^{rs} \times (w^{rs})^{-1} = T_{x,y}$$

SECURITY ANALYSIS

- ⊙ Based on the security of El-Gamal and Bilinear map.
- ⊙ Each user given his own secret keys cannot deduce the master keys from the cipher text $c = T_{x,y} \times w^{rs}$

- ⊙ Encryption scheme is semantically secure:

$$T'_{x,y}\langle 1 \rangle = (c'_1, c_2) = (T_{x,y} \times w^{rs}, w^r)$$

$$T'_{x,y}\langle 2 \rangle = (d'_1, d_2) = (e(H(T_{x,y}), g)^{r'q}, g^{r'})$$

- ⊙ As each proxy does know users' secret keys and each users have different keys, each user's query privacy is protected.
- ⊙ When a user leaves the system, if he reveals his key shares, the master key is still safe.

OTHER CONSIDERATIONS

- ⊙ **User revocation:** key manager just needs to inform all proxies to delete the corresponding key pair $s_{i,j}^u, q_{i,j}^u$ for user U_i .
- ⊙ **Collusion between parties:**
 - ⊙ If an adversary Adv gains access to all the proxies and knows one user key pair, he can recover all the master keys as well as key shares held by all other users.
 - ⊙ To increase the security of the system, more proxies can be used but the tradeoff is computation and communication costs.

KEY RENEWAL

- ⊙ In reality, we expect some key shares will be compromised with high probability → storage and query keys need to be changed periodically.
- ⊙ Naïve way: decrypt the whole database then encrypt with new keys.
- ⊙ Employ proxy re-encryption scheme to encrypt database without the need to decrypt it.

$$\begin{aligned} T'_{x,y}\langle 1 \rangle &= (c'_1, c_2) = (T_{x,y} \times w^{rs}, w^r) \\ T'_{x,y}\langle 2 \rangle &= (d'_1, d_2) = (e(H(T_{x,y}), g)^{r'q}, g^{r'}) \end{aligned}$$

KEY RENEWAL

- ⊙ **Re-encrypt first part:** $T'_{x,y}\langle 1 \rangle = (c'_1, c_2) = (T_{x,y} \times w^{rs}, w^r)$
- ⊙ KM choose a random key s' and k proxies for help.
- ⊙ Split s' : $s' = s'_1 + s'_2 + \dots + s'_{k-1} + s'_k$.
- ⊙ Server send w^r to all proxies.
- ⊙ Each proxy computes: $\beta_i = w^{rs'_i}$ and sends to server
- ⊙ After receiving β_i , server computes:

$$\begin{aligned}\theta_1 &= T_{x,y} \times w^{rs} \times \prod_{i=1}^k \beta_i = T_{x,y} \times w^{rs} \times \prod_{i=1}^k w^{rs'_i} \\ &= T_{x,y} \times w^{rs + \sum_{i=1}^k (rs'_i)} = T_{x,y} \times w^{r(s + \sum_{i=1}^k s'_i)} \\ &= T_{x,y} \times w^{r(s+s')} = T_{x,y} \times w^{rs_0}\end{aligned}$$

KEY RENEWAL

⊙ **Re-encrypt second part:** $T'_{x,y}\langle 2 \rangle = (d'_1, d_2) = (e(H(T_{x,y}), g)^{r'q}, g^{r'})$

⊙ KM chooses a random q' and divides into k shares:

$$q' = q'_1 + q'_2 + \dots + q'_{k-1} + q'_k.$$

⊙ Server sends $e(H(T_{x,y}), g)^{r'q}$ to all proxies.

⊙ Each proxy P_i computes: $\alpha_i = e(H(T_{x,y}), g)^{r'qq'_i}$ and sends to server.

⊙ Server computes:
$$\begin{aligned} \theta_2 &= \prod_{i=1}^k \alpha_i = \prod_{i=1}^k e(H(T_{x,y}), g)^{r'qq'_i} \\ &= e(H(T_{x,y}), g)^{r'q \times (\sum_{i=1}^k q'_i)} \\ &= e(H(T_{x,y}), g)^{r'qq'} = e(H(T_{x,y}), g)^{r'q_0} \end{aligned}$$

KEY RENEWAL

KM sets $s_0 = s + s'$ is the new master storage key.

Server sets $T'_{x,y}\langle 1 \rangle = (T_{x,y} \times w^{r s_0}, w^r)$

KM sets $q_0 = q \times q'$ is the new master query key.

Server sets $T'_{x,y}\langle 2 \rangle = (e(H(T_{x,y}), g)^{r' q_0}, g^{r'})$

A TRUST-BASED PROXY ALLOCATION

- ⊙ Collusion problem less likely for highly trusted users.
- ⊙ Allocate proxies based on users' trust level

ALLOCATION OF PROXIES

User	Trust Level	Class	Number of Proxies	Proxy List
U_1	3	B	2	P_1, P_2
U_2	1	D	4	P_1, P_2, P_3, P_4
U_3	3.85	A	1	P_1
U_4	1.925	C	3	P_1, P_2, P_3



CONCLUSION

- ③ Encryption scheme that supports multiuser, still preserving privacy and security.
- ③ A scheme to periodically renew the key.
- ③ An trust-based proxy allocation