Introduction

The Cloud

Access from Anywhere

Available for Everything

One can
- Store documents, photos, etc
- Share them with colleagues, friends, family
- Process the data
- Ask queries on the data
With Current Solutions

The Cloud provider
- knows the content
- and claims to actually
  - identify users and apply access rights
  - safely store the data
  - securely process the data
  - protect privacy

But...

For economical reasons, by accident, or attacks
- data can get deleted
- any user can access the data
- one can log
  - all the connected users
  - all the queries
to analyze and sell/negotiate the information

Requirements

Users need more
- Storage guarantees
- Privacy guarantees
  - confidentiality of the data
  - anonymity of the users
  - obliviousness of the queries

How to process users’ queries?

FHE: The Killer Tool

Fully Homomorphic Encryption allows to process encrypted data, and get the encrypted output

Some Approaches

[Gentry - STOC '09]
[Rivest-Adleman-Dertouzos - FOCS '78]
[Gentry - STOC '09]
FHE: The Killer Tool

**Fully Homomorphic Encryption** allows to process encrypted data, and get the encrypted output.

### Outsourced Processing

Symmetric encryption (secret key) is enough.

### Strong Privacy

**Privacy by design...**

### FHE: Ideal Solution?

- Allows private storage
- Allows private computations
  - Private queries in an encrypted database
  - Private « googling »
- The provider does not learn
  - the content
  - the queries
  - the answers

... But each gate requires huge computations...
Confidentiality & Sharing

Encryption allows to protect data
- the provider stores them without knowing them
- nobody can access them either, except the owner

How to share them with friends?

- Specific people have full access to some data: with public-key encryption for multiple recipients
- Specific people have partial access such as statistics or aggregation of the data

Broadcast Encryption

The sender can select the target group of receivers
- This allows to control who will access to the data

Functional Encryption

The user generates sub-keys $K_y$ according to the input $y$
- From $C = \text{Encrypt}(x), \text{Decrypt}(K_y, C)$ outputs $f(x, y)$
- This allows to control the amount of shared data

Outline

- Broadcast Encryption
  - Efficient solutions for sharing data
- Functional Encryption
  - Some recent efficient solutions for inner product
- Fully Homomorphic Encryption
  - Despite recent improvements, this is still inefficient
  - With 2-party computation one can get an efficient alternative
**Multi-Party Computation**

- **Secure Multi-Party Computation**
  - Ideally: each party gives its input and just learns its output for **any** ideal functionality

---

**Two-Party Computation**

- General construction: Yao Garbled Circuits
- For specific construction: quite inefficient

**Encryption Switching Protocols**

\[ f(x, y) = (x + y)^e \mod n \]

With **additive** encryption \( E^+ \), **multiplication** encryption \( E^x \), for which Alice and Bob share the decryption keys, and an interactive **switch** from \( c^+ \) to \( c^x \):
- Alices sends \( c^+_A = E^+(x) \), and Bob sends \( c^+_B = E^+(y) \)
- They compute \( c = c^+_A \oplus c^+_B = E^+(x+y) \)
- They run the **interactive switch** to get \( c' = E^x(x+y) \)
- They compute \( C = \otimes e^{c'} = E^x((x+y)^e) \)
- They run the **interactive decryption** to get \( z \)

[Coateau-Peters-P - Crypto '16]
**Homomorphic Encryption**

**Additive encryption on \( \mathbb{Z}_n \): Paillier encryption**

- **Public key:** \( n = pq \)
- **Secret key:** \( d = [\lambda^{-1} \mod n] \times \lambda \)
- **Encryption:** \( c = (1 + n)^m \cdot r^n \mod n^2 \)
- **Decryption:** \( m = [c^d - 1 \mod n^2] / n \)

- Additive homomorphic
- Efficient interactive decryption


**Multiplicative encryption on \( \mathbb{Z}_n \): ElGamal encryption**

- **Secret key:** \( x \in \mathbb{Z}_p \)
- **Public key:** \( h = g^x \)
- **Encryption:** \( c = (c_0 = g^{r}, c_1 = h^r \cdot m) \)
- **Decryption:** \( m = c_1 / c_0^x \)

- Multiplicatively homomorphic
- Efficient interactive decryption

If \( n = pq \), with safe primes \( p = 2p' + 1 \) and \( q = 2q' + 1 \)
Works for \( \mathbb{G} = QR_n \), under the DDH in \( \mathbb{Z}_{p'}^* \) and \( \mathbb{Z}_{q'}^* \)
Works for \( \mathbb{G} = \mathbb{J}_n \), under the additional QR assumption
But does not work in \( \mathbb{Z}_n^* \)

**Encoding of Messages**

**Multiplicative encryption on \( \mathbb{Z}_n^* \): by encoding \( m \in \mathbb{Z}_n^* \) into \( \mathbb{J}_n \)**

- For \( n = pq \), generator \( g \) of \( \mathbb{J}_n \) of order \( \lambda \)
  \( \chi \in \mathbb{Z}_n^* \setminus \mathbb{J}_n \), using the CRT:
  \( \chi = g^{t_p} \mod p \), for an even \( t_p \): \( \chi \in QR_p \)
  \( \chi = g^{t_q} \mod q \), for an odd \( t_p \): \( \chi \notin QR_q \)
  hence \( \chi \in \mathbb{Z}_n^* \setminus \mathbb{J}_n \)

- For \( m \in \mathbb{Z}_n^* \), \( a \in \{1, \ldots, n/2\} \), so that \( \chi^a \cdot m \in \mathbb{J}_n \)

- From \( m_1 = g^a \mod n\)
- From \( m_2 \), one gets \( \alpha = m_2 / \alpha \mod n \)

- Multiplicatively homomorphic
- Efficient interactive decryption
- Efficient encryption switching protocols with the Paillier encryption
Two-Party Computation?

The two homomorphic encryption schemes together with the encryption switching protocols:
- Efficient two-party computation
- But in the intersection of the plaintext spaces!
\[ \mathbb{Z}_n \cap \mathbb{Z}_n^* = \mathbb{Z}_n^* \]
- Cannot deal with zero!
- But cannot avoid zero either during computations!

How to Handle Zero?

In order to multiplicatively encrypt \( m \in \mathbb{Z}_n \):
- One defines \( b = 1 \) if \( m = 0 \), and \( b = 0 \) otherwise
- One encrypts \( A = m + b \mod n \)
- One encrypts \( B = T^b \mod n \) for a random square \( T \)

One can note that
- \( A \in \mathbb{Z}_n^* \), unless \( m \) is a non-trivial multiple of \( p \) or \( q \)
- \( B \in \text{QR}_n \)
- \( \implies \) they can both be encrypted with appropriate ElGamal-like encryption

- Multiplicatively homomorphic: 0 is absorbing in \( B \)
- Encrypted Zero Test protocols: \( E^+(m) \to E^+(b) \)

Set Disjointness Testing

Alice’s friends: \( A = \{a_1, \ldots, a_m\} \) Bob’s friends: \( B = \{b_1, \ldots, b_n\} \)
\( A \cap B = \emptyset ? \)
- Alice computes \( P(X) = \prod_i (X - a_i) = \sum_i A_i X^i \), and sends \( C_i = E^+(A_i) \)
- Bob computes \( B_j = E^+(P(b_j)) = \sum_i b_j C_i \)
- They switch to \( B'_j = E^+(P(b_j)) \)
- They compute \( C' = E^+(\prod_i P(b_j)) = \prod_j B'_j \)
- They decrypt \( C' \to c = \prod_j P(b_j) = \prod_i \prod_j (b_j - a_i) \)
- \( c = 0 \iff A \cap B \neq \emptyset \)

Outsourced Computations

- The user possesses \( n=pq \)
- The user gives the shares to 2 independent servers

Interactive Fully Homomorphic Encryption
Homomorphic Encryption

Additive encryption on $\mathbb{Z}_n$: BCP encryption

Parameters: $n = pq$ and a square $g \in \mathbb{Z}_{n^2}^*$

Secret key: $x \in \mathbb{Z}_{\lambda(n)}$

Public key: $h = g^x \mod n^2$

Encryption: $c_0 = g^r \mod n^2$, for $n \in [1..n^2/2]$

Decryption: $m = [c_1/c_0^2 - 1 \mod n^2]/n$

Alternatively: with $\lambda(n) \rightarrow x_0 = x \mod n$

(Where $x = x_0 + nx_1$)

$$c_1/c_0^{x_0} = g^{(x - x_0)r} \cdot (1 + mn) = (g^{rx_1})^n \cdot (1 + mn)$$

= $u^n \cdot (1 + n)^m \mod n^2$

Multi-User Setting

- The two independent servers share the Paillier’s secret key for $n=pq$ and setup a BCP scheme
- The servers can convert BCP ciphertexts into Paillier ciphertexts, and run the 2-party protocol
- The servers can convert a Paillier ciphertext into a BCP ciphertext for a specific user

⇒ Secure efficient outsourced computations

More servers can be used:
unless all the servers corrupted, privacy guaranteed

Conclusion

- Threat
  However strong the trustfulness of the Cloud provider may be, any system or human vulnerability can be exploited against privacy

- Privacy by design
  Tools to limit data access

- The provider is just trusted to
  - store the data (can be controlled)
  - process and answer any request (or DoS)