# Homomorphic Encryption & Encrypted Databases

Slides adapted from here, and here

# Paper review

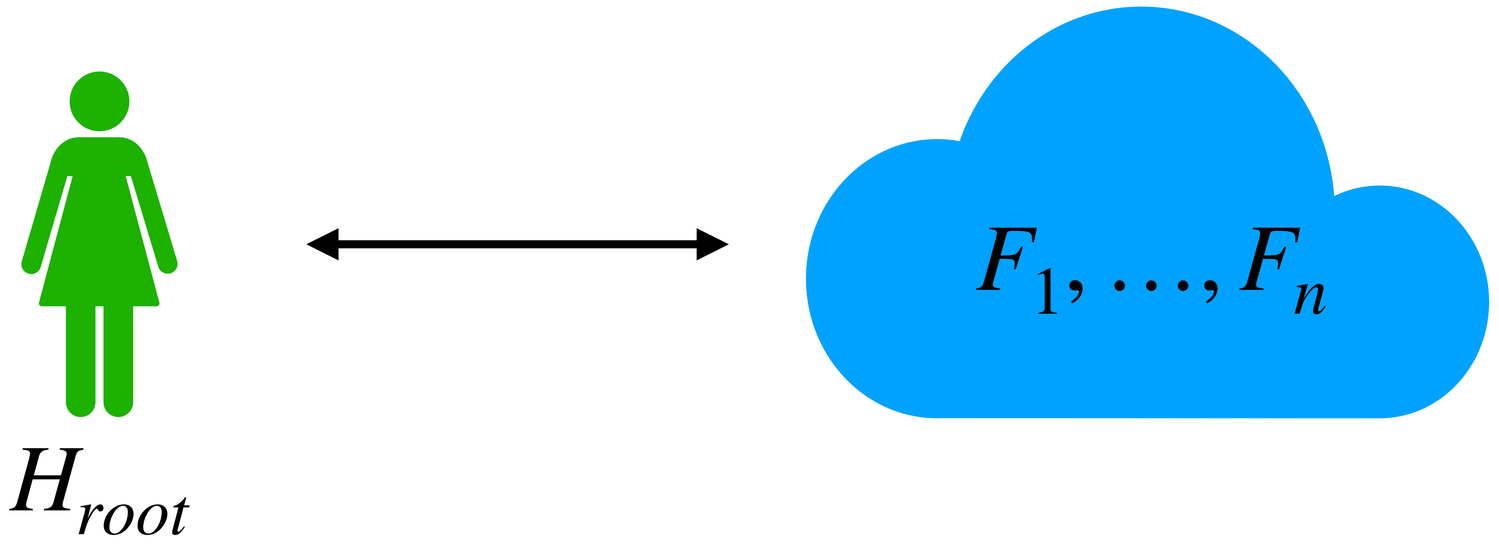
- Format
  - What problem is the paper addressing? (2-3 sentences)
  - Explain the techniques used in the paper (both crypto and systems). (2-3 paragraphs)
    - Don’t just list out the techniques, but explain how the system is built using these techniques.
  - What are the limitations? What are some interesting future works? (1 paragraph)
    - Think about functionality vs. security vs. efficiency
  - Discussion question (1 - 2 sentences)
- Due by **4 pm the day before lecture** (Sunday & Tuesday nights)

# Final project

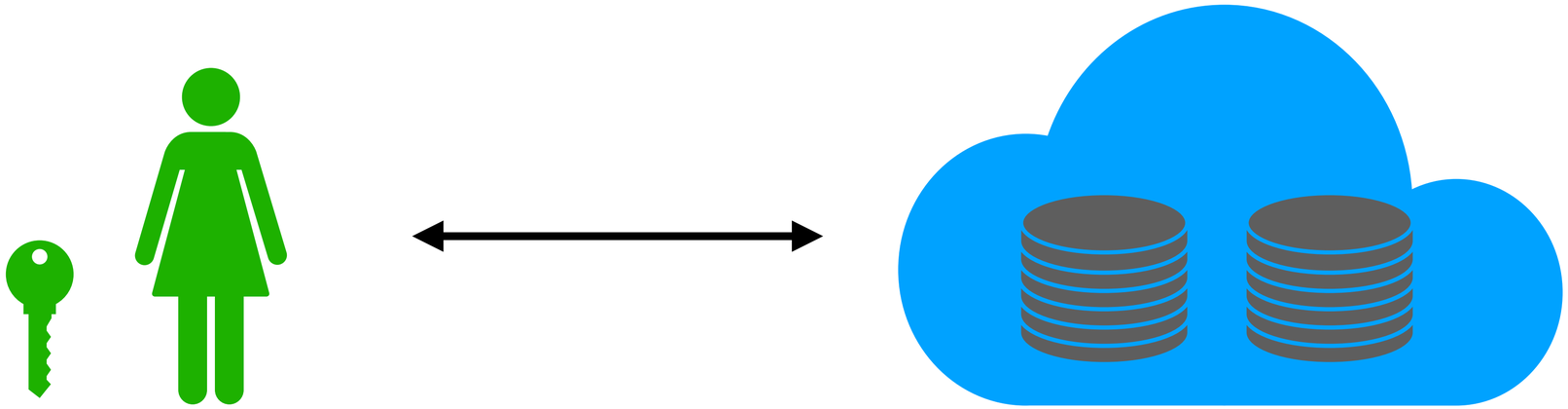
- Information about the final project now updated on the website
- If you need a project partner, email me two topic by Thursday and I will match you
- Hybrid mode teaching over Zoom

# Last week

- Alice has files $F_1, \ldots, F_n$ and wants to store them in the cloud
- Alice does not entirely trust the cloud with respect to data integrity
- When she retrieves $F_i$, she can use Merkle proof to audit the integrity of that file



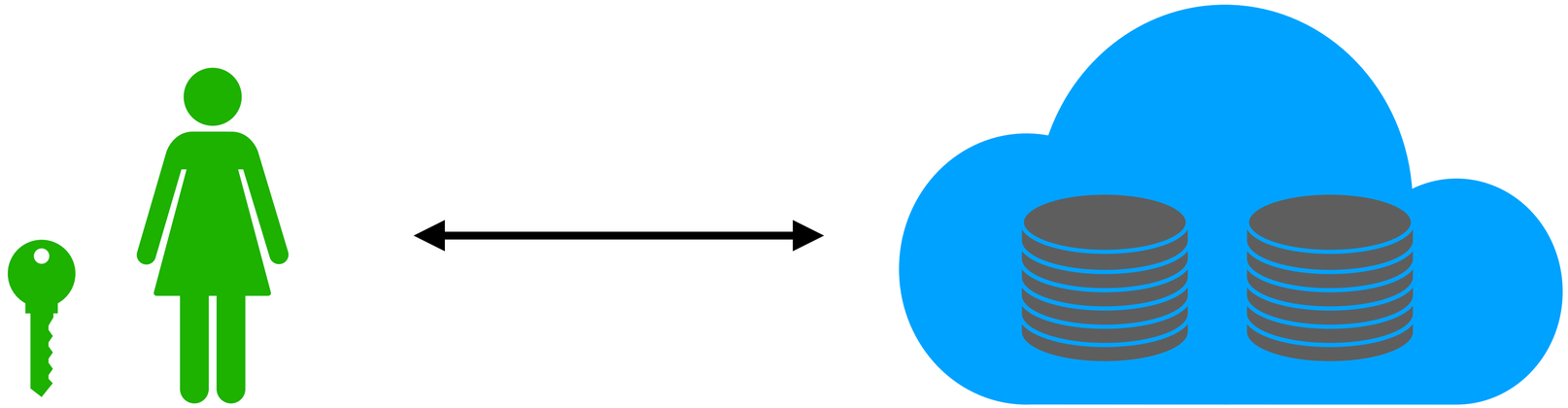
# Alice wants privacy



Alice does not want the cloud to see its data

→ people today use encryption at rest

# Alice also wants functionalities



Alice wants to be able to process its data

→ ???

# Computing on *encrypted data*?

- Many cryptographic techniques
- The concept of *fully homomorphic encryption* (FHE) was originally proposed by Rivest, Adleman and Dertouzos [RAD78]
- Unsolved for many many years until **Craig Gentry's breakthrough in 2009** using ideal lattices

# FHE properties

- **Semantic security:** We say that an encryption scheme $(G, E, D)$ is semantically secure if for all PPT algorithms $\mathscr{A}$, and for $\forall m_0, m_1$, if $b$ is chosen uniformly random from $\{0,1\}$, and $c = \texttt{Enc}(m_b)$, then

$$|Pr[\mathscr{A}(c) = b] - \frac{1}{2}| = \text{negligible}$$

  - $\mathscr{A}$ cannot guess $\hat{b} = b$ with more advantage than just guessing randomly
  - Encryption scheme has to be probabilistic

- **Functionality:** for any polynomially computable function $f$, FHE guarantees

$$f(\texttt{Enc}(x_1), \ldots, \texttt{Enc}(x_n)) = \texttt{Enc}(f(x_1, \ldots, x_n))$$

# Partially homomorphic encryption

- An encryption scheme that is homomorphic with respect to a specific function, but cannot compute arbitrary functions like FHE
- Many constructions known for decades
- Sometimes faster than FHE due to *specialization*

# DDH assumption

- Primer: $Z_q = \{0,1,\cdots,q-1\}$
- Let $\mathbb{G}$ be a cyclic group of prime order $q$ generated by $g \in \mathbb{G}$
  - Challenger computes $\alpha, \beta, \gamma \leftarrow \mathbb{Z}_q, u \leftarrow g^{\alpha}, v \leftarrow g^{\beta}, w_0 \leftarrow g^{\alpha\beta}, w_1 \leftarrow g^{\gamma}$
  - $(u, v, w_0) = (g^{\alpha}, g^{\beta}, g^{\alpha\beta})$ is a Diffie-Hellman tuple
  - Challenger gives $(u, v, w_b)$ to the adversary where $b \leftarrow \{0,1\}$
  - Adversary computes $\hat{b} \in \{0,1\}$
- DDH assumption holds if adversary cannot guess $\hat{b} = b$ with more advantage than guessing randomly

# ElGamal (1985)

- Setup
  - Let $\mathbb{G}$ be a cyclic group of order $q$ with generator $g$, such that DDH is hard in $\mathbb{G}$
- KeyGen($1^k$):
  - $x \xleftarrow{R} \mathbb{Z}_q$
  - Output $(pk, sk) = (g^x, x)$
- Enc($pk, m$):
  - $r \xleftarrow{R} \mathbb{Z}_q, v \leftarrow g^r$
  - Encode message as $m \in \mathbb{G}$
  - Output $c = (g^r, m \cdot pk^r) = (g^r, m \cdot g^{xr})$
- Dec($sk, (c_1, c_2)$):
  - Compute $c_2 / c_1^x = m \cdot g^{xr} / g^{xr} = m$

# ElGamal security

- **Theorem:** If the DDH problem is computationally hard, then the ElGamal encryption scheme has semantic security

- **Proof:** Assume we have an adversary $\mathscr{A}$ who can break the semantic security of ElGamal encryption, then we can construct an adversary $\mathscr{B}$ who can break DDH.

  - Given two messages $m_0, m_1$, public key $pk$, $\mathscr{A}$ plays game with challenger who gives $c = \text{Enc}(m_b)$ to $\mathscr{A}$, where $b \leftarrow \{0,1\}$

  - $\mathscr{A}(pk, c)$ can produce $b'$ such that $b' = b$ with **non-neglibigle** probability

# ElGamal security

- **Proof (cont'd):** We will construct $\mathscr{B}$ with $\mathscr{A}$.
  - $\mathscr{B}(u, v, w)$, needs to distinguish whether $(u, v, w)$ is a DH tuple
    - Set $pk = (g, u)$
    - Receive $m_0, m_1$; choose $b \leftarrow \{0,1\}$
    - Set $c = (v, m_b \cdot w)$
    - Run $\mathscr{A}(pk, c)$ to get output $b'$
    - If $b' = b$, then guess that $(u, v, w)$ is a DH tuple, otherwise guess no

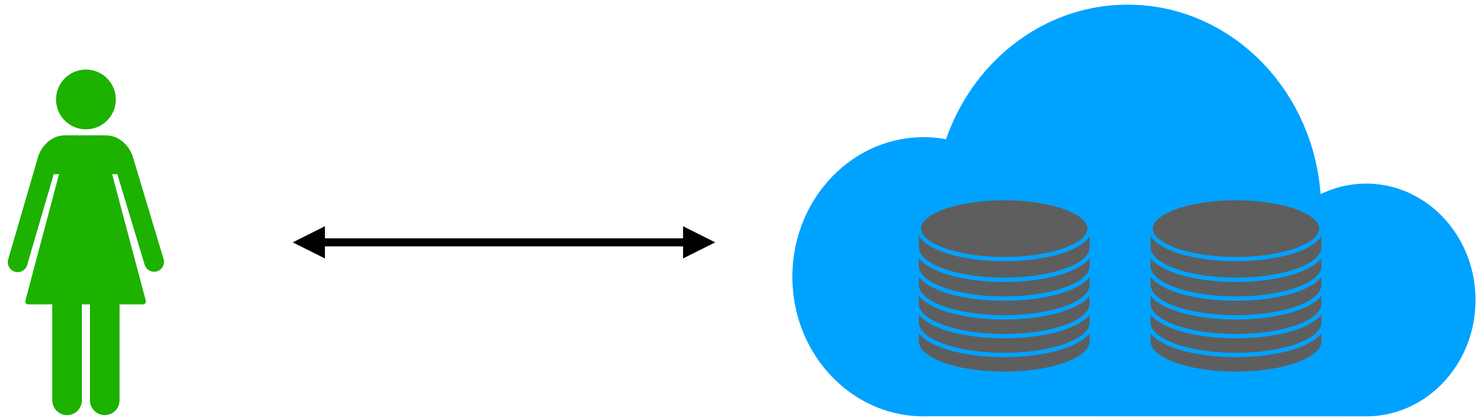
- If $(u, v, w) = (g^{\alpha}, g^{\beta}, g^{\alpha\beta})$, then $c = (g^{\beta}, m_b \cdot g^{\alpha\beta})$, which means $\mathscr{A}$ should be able to distinguish between $m_0$ and $m_1$

- If $(u, v, w)$ is not a DH tuple, then $c = (v, m_b \cdot w)$ is not a valid encryption.

# ElGamal functionality

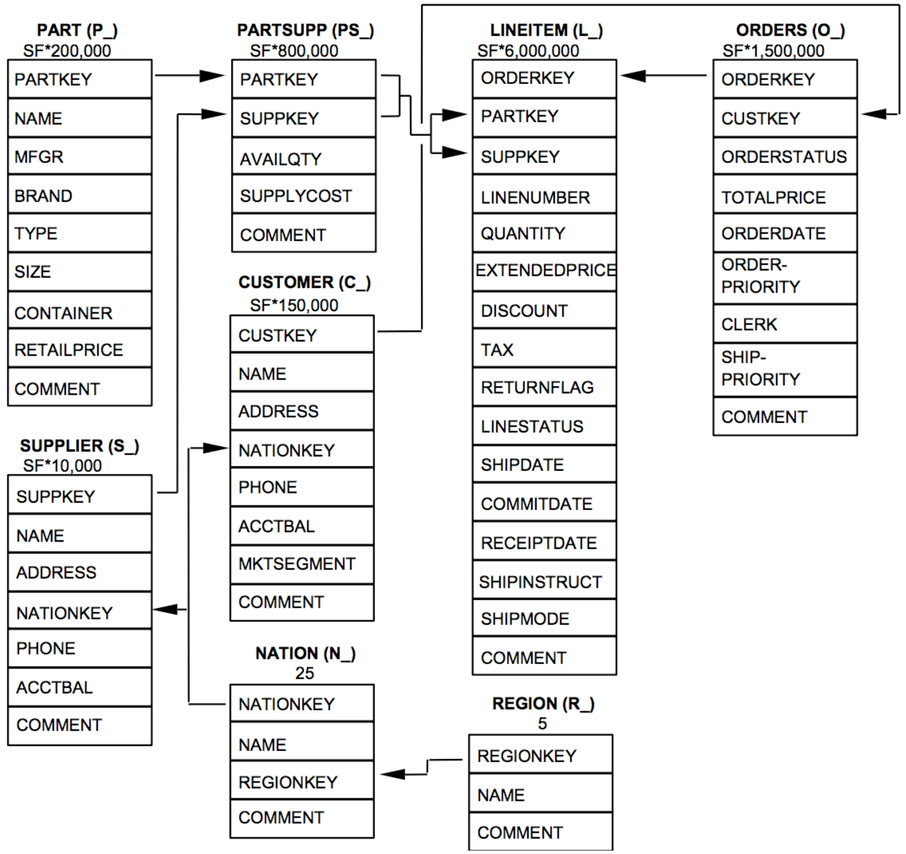
- Given two ciphertexts $c_0, c_1$, can we compute a functionality directly on these ciphertexts?
  - $c_0 = (g^{r_0}, m_0 \cdot pk^{r_0}), c_1 = (g^{r_1}, m_1 \cdot pk^{r_1})$
  - $c_0 \cdot c_1 = (g^{r_0+r_1}, m_0 \cdot m_1 \cdot pk^{r_0+r_1})$, which is a valid ciphertext of $m_0 \cdot m_1$!

# SQL workload?



# Data stored in tables

**Figure 2: The TPC-H Schema**




**Legend:**

- The parentheses following each table name contain the prefix of the column names for that table;
- The arrows point in the direction of the one-to-many relationships between tables;
- The number/formula below each table name represents the cardinality (number of rows) of the table. Some are factored by SF, the Scale Factor, to obtain the chosen database size. The cardinality for the LINEITEM table is approximate (see Clause 4.2.5).

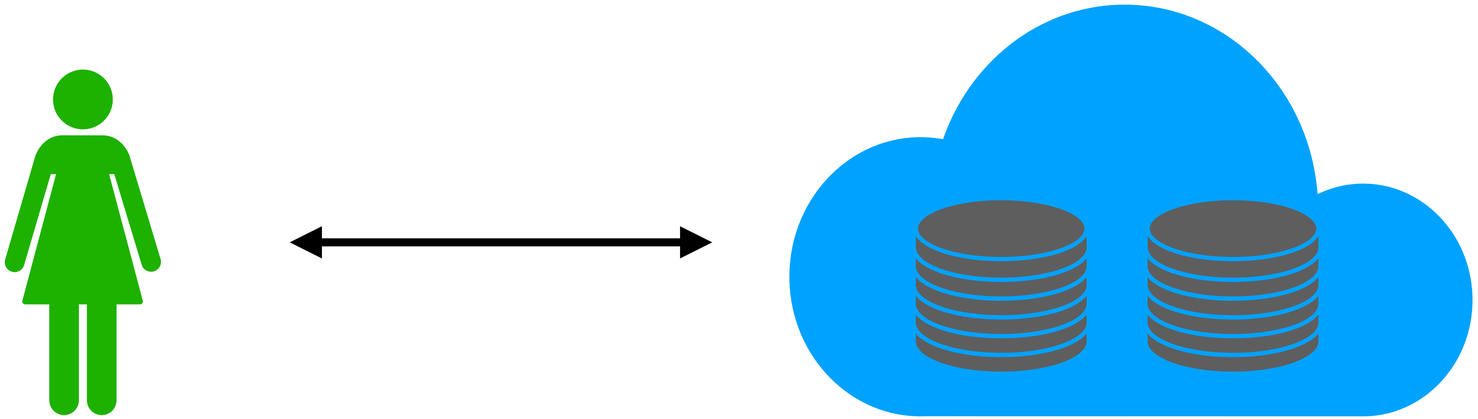
# DBMS provide data manipulation language

- Expressions on a set of attributes
  - Arithmetic expressions, string manipulation, etc.
- Operators on tables
  - Select, Projection, Filter, Union, Join, Aggregation, Order
- Nested scalar subqueries within expressions…

# Why SQL?

- “Narrow waist” of data analytics
  - Very powerful language
  - Can support machine learning and graph analytics as well
- Automated query optimizations & planning
  - Logical optimizations such as filter pushdown
  - Physical planning such as join reordering

# SQL workload?



```
SELECT name, avg(salary)
FROM employee
ORDER BY name;
```

# Today's readings:
# Running SQL on encrypted data