CS459/698
Privacy, Cryptography, Network and Data Security

A pinch of Homomorphic Encryption

Fall 2024, Tuesday/Thursday 02:30pm-03:50pm

What is Homomorphic Encryption?

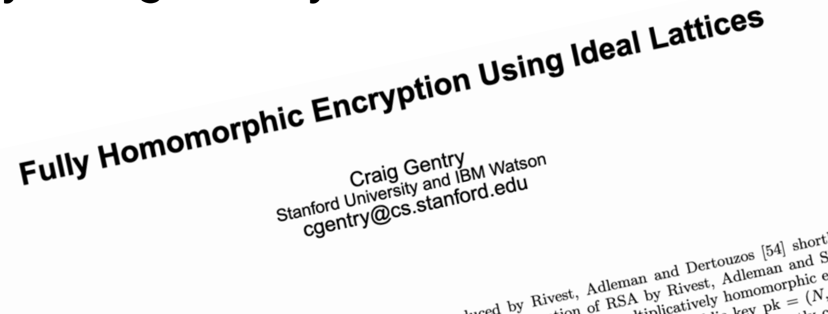
What is Homomorphic Encryption?

- **Definition:** Homomorphic encryption is a cryptographic technique that allows computations to be performed on encrypted data without requiring decryption.

- Raw data can remain fully encrypted while it's being processed, manipulated, and run through various algorithms.

What is Homomorphic Encryption?

- **Definition:** Homomorphic encryption is a cryptographic technique that allows computations to be performed on encrypted data without requiring decryption.
- Raw data can remain fully encrypted while it’s being processed, manipulated, and run through various algorithms.
- Idealized in 1978, fully realized in 2009 by Craig Gentry


Fully Homomorphic Encryption Using Ideal Lattices

Craig Gentry
Stanford University and IBM Watson
cgentry@cs.stanford.edu

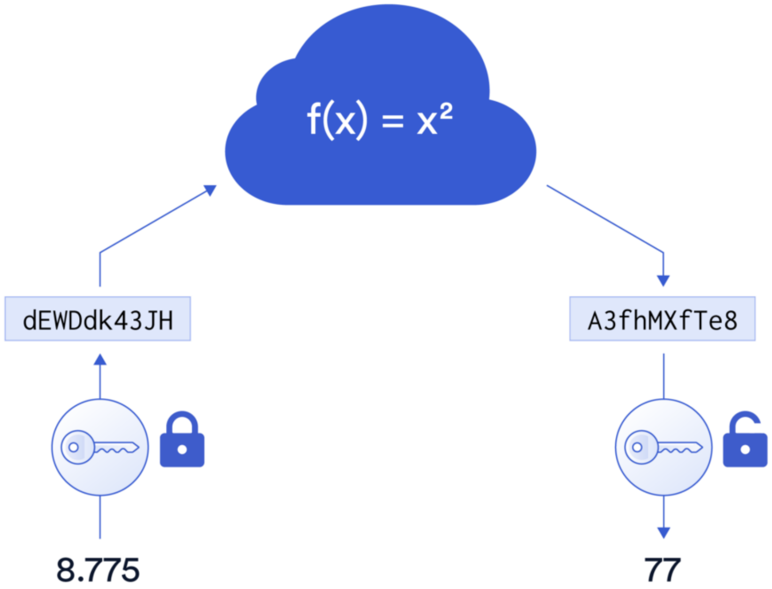
Homomorphic Encryption for Dummies

"Anybody can come and they can stick their hands inside the gloves and manipulate what's inside the locked box. They can't pull it out, but they can manipulate it; they can process it… Then they finish and the person with the secret key has to come and open it up—and only they can extract the finished product out of there."

-- Craig Gentry

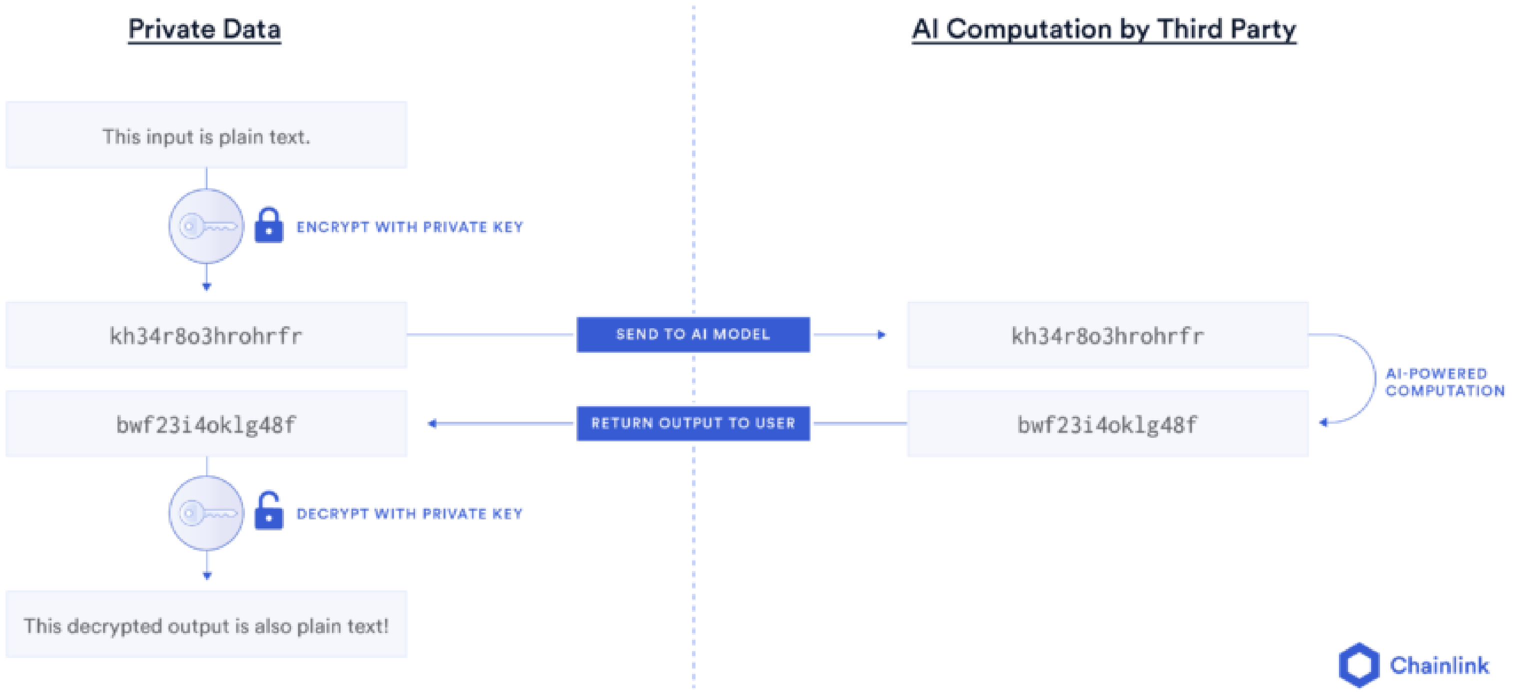
https://www.youtube.com/watch?v=pXb39wj5ShI

Computing on Ciphertexts (Simple Math)



https://chain.link/education-hub/homomorphic-encryption

Computing on Ciphertexts (More sophisticated math)



https://chain.link/education-hub/homomorphic-encryption

Homomorphic Encryption in the Wild

https://dualitytech.com/blog/homomorphic-encryption-making-it-real/

Homomorphic Encryption in the Wild

- Used as a tool in many real-world scenarios:
  - https://www.ibm.com/security/services/homomorphic-encryption
  - https://www.statcan.gc.ca/en/data-science/network/homomorphic-encryption
  - https://www.statcan.gc.ca/en/data-science/network/statistical-analysis-homomorphic-encryption
  - https://www.intel.com/content/www/us/en/developer/tools/homomorphic-encryption/overview.html
  - https://www.microsoft.com/en-us/research/project/microsoft-seal/
  - https://machinelearning.apple.com/research/homomorphic-encryption
  - https://github.com/apple/swift-homomorphic-encryption
  - https://github.com/google/heir
  - https://github.com/google/fully-homomorphic-encryption?tab=readme-ov-file

E.g., Homomorphic Encryption for Secure Voting

- Microsoft's ElectionGuard

Microsoft | **Microsoft On the Issues** Our Company News and Stories Topics More

Protecting democratic elections through secure, verifiable voting

May 6, 2019 | Tom Burt - Corporate Vice President, Customer Security & Trust

What does ElectionGuard do?

ElectionGuard is a way of checking election results are accurate, and that votes have not been altered, suppressed or tampered with in any way. Individual voters can see that their vote has been accurately recorded, and their choice has been correctly added to the final tally. Anyone who wishes to monitor the election can check all votes have been correctly tallied to produce an accurate and fair result.

So what is this all about?

Homomorphic Encryption

Consider the following:

Two ciphertexts use the same key, **c** = E_K(**x**), **d** = E_K(**y**)

Let **f()** be a function that operates over plaintext **x** and **y**

Homomorphic Encryption

Consider the following:

Two ciphertexts use the same key, **c** = E_K(**x**), **d** = E_K(**y**)

Let **f()** be a function that operates over plaintext **x** and **y**

Goal: the existence of a function **g()** such that

$$g(c, d) = E_K(f(x, y))$$

Homomorphic Encryption

Consider the following:

Two ciphertexts use the same key, **c** = E_K(**x**), **d** = E_K(**y**)

Let **f()** be a function that operates over plaintext **x** and **y**

Goal: the existence of a function **g()** such that

$$g(c, d) = E_K(f(x, y))$$

g() is a **homomorphic function** on the ciphertexts **c**, **d**, …

Partial versus Fully Homomorphic Encryption

The function on the plaintexts is:

…either multiplication or addition **but not both.**

Partial HE

…either multiplication or addition, **or both**

Fully HE

4 Shades of Homomorphic Encryption

Homomorphic Encryption Types				
	Partially	Somewhat	Leveled Fully	Fully
Rating	Simple	Intermediate	Advanced	Most advanced
Computations	Addition or multiplication	Addition and / or multiplication	Complex but limited	Complex and unlimited
Use cases	Sum or product	Basic statistical analysis	AI/ML, MPC	AI/ML, MPC

https://chain.link/education-hub/homomorphic-encryption

4 Shades of Homomorphic Encryption

Only useful for **simpler** operations. Relatively **efficient**.

	Partially	Somewhat	Leveled Fully	Fully
Rating	Simple	Intermediate	Advanced	Most advanced
Computations	Addition or multiplication	Addition and / or multiplication	Complex but limited	Complex and unlimited
Use cases	Sum or product	Basic statistical analysis	AI/ML, MPC	AI/ML, MPC

https://chain.link/education-hub/homomorphic-encryption

4 Shades of Homomorphic Encryption

of operations that can be performed is **bounded** and the accuracy of the computation may **degrade** as more operations are performed.

	Partially	Somewhat	Leveled Fully	Fully
Rating	Simple	Intermediate	Advanced	Most advanced
Computations	Addition or multiplication	Addition and / or multiplication	Complex but limited	Complex and unlimited
Use cases	Sum or product	Basic statistical analysis	AI/ML, MPC	AI/ML, MPC

https://chain.link/education-hub/homomorphic-encryption

4 Shades of Homomorphic Encryption

Can perform an **arbitrary** # of computations on encrypted data, if it has a pre-defined set of computations **specified ahead of time**.

	Partially	Somewhat	Leveled Fully	Fully
Rating	Simple	Intermediate	Advanced	Most advanced
Computations	Addition or multiplication	Addition and / or multiplication	Complex but limited	Complex and unlimited
Use cases	Sum or product	Basic statistical analysis	AI/ML, MPC	AI/ML, MPC

https://chain.link/education-hub/homomorphic-encryption

4 Shades of Homomorphic Encryption

Enables **any** # of computations to be performed on encrypted data **without a predefined sequence or limit**. Computationally **expensive**.

	Partially	Somewhat	Leveled Fully	Fully
Rating	Simple	Intermediate	Advanced	Most advanced
Computations	Addition or multiplication	Addition and / or multiplication	Complex but limited	Complex and unlimited
Use cases	Sum or product	Basic statistical analysis	AI/ML, MPC	AI/ML, MPC

https://chain.link/education-hub/homomorphic-encryption

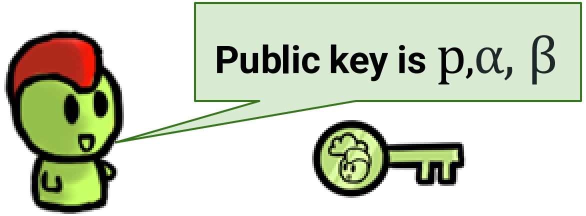
A partial homomorphic encryption scheme based on El Gamal

Recap: ElGamal Public Key Cryptosystem

- Let $\mathbf{p}$ be a prime such that the DLP in $(\mathbf{Z}_p^*, .)$ is infeasible
- Let $\boldsymbol{\alpha}$ be a generator in $\mathbf{Z}_p^*$ and "**a**" a secret value
- $\mathbf{Pub_K} = \{(p, \alpha, \beta) : \beta \equiv \alpha^a \pmod p\}$

- For message "**m**" and secret random "**k**" in $\mathbf{Z}_{p-1}$:
  - $e_K(m,k) = (y_1, y_2)$, where $\mathbf{y_1} = \alpha^k \bmod p$ and $\mathbf{y_2} = m\beta^k \bmod p$

- For y_1, y_2 in $\mathbf{Z}_p^*$:
  - $d_K(y_1, y_2) = y_2 (y_1{}^a)^{-1} \bmod p$



Consider Multiplicative HE

Bob's Pub_K → **(p, α, β)**

Bob's $Priv_K$ → **a**

$\mathbf{y_1} \equiv \alpha^k \pmod{p}$

$\mathbf{y_2} \equiv m\, \beta^k \pmod{p}$

$\beta \equiv \alpha^a \pmod{p}$

f(x, y) = x · y

Private key: a, **public key:** α^a

Instead of k, choose r and s

Consider Multiplicative HE

Bob's Pub_K → (p, α, β)

Bob's $Priv_K$ → a

$y_1 \equiv \alpha^k \pmod{p}$

$y_2 \equiv m\,\beta^k \pmod{p}$

$\beta \equiv \alpha^a \pmod{p}$

$f(x, y) = x \cdot y$

Private key: a, **public key:** α^a

Instead of k, choose r and s

Goal: show how the multiplication of ciphertexts corresponds to the multiplication of plaintexts.

Consider Multiplicative HE

Bob's Pub_K → **(p, α, β)**

Bob's $Priv_K$ → **a**

$\mathbf{y_1} \equiv \alpha^k \pmod p$

$\mathbf{y_2} \equiv m\,\beta^k \pmod p$

$\beta \equiv \alpha^a \pmod p$

f(x, y) = x · y

Private key: a, **public key:** α^a

Instead of k, choose r and s

$c_1 = \alpha^r,\ c_2 = \mathbf{x}\,\alpha^{ra};$

$d_1 = \alpha^s,\ d_2 = \mathbf{y}\,\alpha^{sa}$

Idea: Create ciphertexts for the two different plaintexts x and y

Consider Multiplicative HE

Bob's $Pub_K \rightarrow$ **(p, α, β)**

Bob's $Priv_K \rightarrow$ **a**

$\mathbf{y_1} \equiv \alpha^k \pmod p$

$\mathbf{y_2} \equiv m\, \beta^k \pmod p$

$\beta \equiv \alpha^a \pmod p$

f(x, y) = x · y

Private key: a, **public key:** α^a

Instead of k, choose r and s

$c_1 = \alpha^r,\ c_2 = \mathbf{x}\, \alpha^{ra};$

$d_1 = \alpha^s,\ d_2 = \mathbf{y}\, \alpha^{sa}$

g(c, d):

- $e_1 = c_1 \cdot d_1 = \alpha^r \alpha^s = \alpha^{r+s}$
- $e_2 = c_2 \cdot d_2 = \mathbf{xy}\, \alpha^{ra} \alpha^{sa} = \mathbf{xy}\, \alpha^{a(r+s)}$

Idea: combine ciphertexts of two different plaintexts

Consider Multiplicative HE

Bob's $Pub_K \rightarrow$ **(p, α, β)**

Bob's $Priv_K \rightarrow$ **a**

$\mathbf{y_1} \equiv \alpha^k \pmod{p}$

$\mathbf{y_2} \equiv m\,\beta^k \pmod{p}$

$\beta \equiv \alpha^a \pmod{p}$

$f(x, y) = x \cdot y$

Private key: a, **public key:** α^a

Instead of k, choose r and s

$c_1 = \alpha^r,\ c_2 = \mathbf{x}\,\alpha^{ra};$

$d_1 = \alpha^s,\ d_2 = \mathbf{y}\,\alpha^{sa}$

$\mathbf{g(c, d)} = \mathbf{xy}\,\alpha^{a(r+s)}$

$\mathbf{xy} = \mathbf{xy}\,\alpha^{a(r+s)} / \alpha^{a(r+s)}$

Idea: decrypt the combined ciphertext

Consider Additive HE

Multiplicative: The math of ElGamal ensures that multiplying the encrypted values corresponds to multiplying the original plaintext values.

Additive: Here, we no longer have the same nice properties of how exponents play together.

- **"Crazy" idea:** Something like $g(E_K(\alpha^x), E_K(\alpha^y)) = E_A(\alpha^{x+y})$ could work
  - But we would need to break the discrete log of α^{x+y} to retrieve the sum
    - Only really works for small **x** and **y**

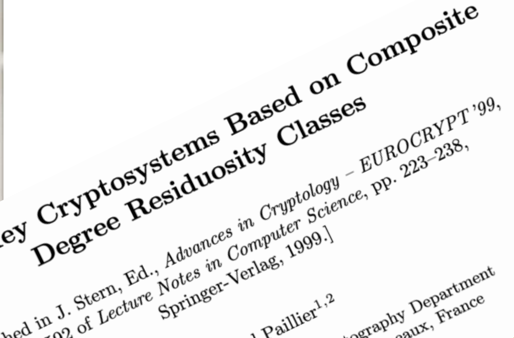
The Paillier Partially Homomorphic Encryption Scheme

Paillier's Encryption Scheme

- Proposed by Pascal Pailler in 1999
- The Paillier cryptosystem is a public-key cryptosystem known for its **additive** homomorphic properties.
- The security of the Paillier cryptosystem is based on the difficulty of the **composite residuosity class problem**
  - This problem involves determining whether a given number is an n-th residue modulo n^2 for a composite n.

Paillier’s Encryption Scheme

- Proposed by Pascal Pailler in 1999
- The Paillier cryptosystem is a public-key cryptosystem known for its **additive** homomorphic properties.
- The security of the Paillier cryptosystem is based on the difficulty of the **composite residuosity class problem**
  - This problem involves determining whether a given number is an n-th residue modulo n^2 for a composite n.


Public-Key Cryptosystems Based on Composite Degree Residuosity Classes

Stern, Ed., *Advances in Cryptology – EUROCRYPT ’99*, *Lecture Notes in Computer Science*, pp. 223–238, Springer-Verlag, 1999.]

Paillier's Encryption Scheme

- Let $\boldsymbol{p}$, $\boldsymbol{q}$ be two large primes; $\boldsymbol{N} = \boldsymbol{pq}$
- Ciphertexts are mod $\boldsymbol{N}^2$

Paillier's Encryption Scheme

- Let $\boldsymbol{p}$, $\boldsymbol{q}$ be two large primes; $\boldsymbol{N} = \boldsymbol{pq}$
- Ciphertexts are mod $\boldsymbol{N}^2$
- Choose $\boldsymbol{r}$; plaintext $\boldsymbol{m}$ (mod **p**) is encrypted as $\mathbf{g^m\, r^N \ (mod\ N^2)}$

g is a generator

Paillier's Encryption Scheme

- Let ***p***, ***q*** be two large primes; ***N*** = ***pq***
- Ciphertexts are mod N^2
- Choose ***r*** ; plaintext ***m*** (mod **p**) is encrypted as $g^m r^N \pmod{N^2}$

Paillier's Encryption Scheme

- Let ***p***, ***q*** be two large primes; ***N*** = ***pq***
- Ciphertexts are mod $\boldsymbol{N}^2$
- Choose ***r*** ; plaintext ***m*** (mod **p**) is encrypted as $\mathbf{g^m \, r^N \, (mod\ N^2)}$

From the **product of ciphertexts** to **addition of plaintexts**

- Multiply encryption of $\mathbf{m_1}$ and $\mathbf{m_2}$:

$$E(m_1,r_1) \cdot E(m_2,r_2) \bmod N^2 =$$
$$g^{m1} \cdot g^{m2} \cdot r_1^N \cdot r_2^N \bmod N^2 =$$
$$g^{m1+m2} \cdot (r_1 \cdot r_2)^N \bmod N^2$$

Paillier's Encryption Scheme

- Multiply encryption of $\mathbf{m_1}$ and $\mathbf{m_2}$:

$$E(m_1,r_1) \cdot E(m_2,r_2) \bmod N^2 =$$
$$g^{m1} \cdot g^{m2} \cdot {r_1}^N \cdot {r_2}^N \bmod N^2 =$$
$$g^{m1+m2} \cdot (r_1 \cdot r_2)^N \bmod N^2$$

- If factorization of ***N*** is known, breaking the DL is efficient

  ⇒ Efficient additive HE, even for large numbers

Paillier's Encryption Scheme

- Multiply encryption of $\mathbf{m_1}$ and $\mathbf{m_2}$:

$$E(m_1,r_1) \cdot E(m_2,r_2) \bmod N^2 =$$

$$g^{m1} \cdot g^{m2} \cdot {r_1}^N \cdot {r_2}^N \bmod N^2 =$$

$$g^{m1+m2} \cdot (r_1 \cdot r_2)^N \bmod N^2$$

- If factorization of ***N*** is known, breaking the DL is efficient

⇒ Efficient additive HE, even for large numbers

$$D(E(m_1, r_1) \cdot E(m_2, r_2) \bmod n^2) = m_1 + m_2 \bmod n$$

Simplica Numara!

DGHV:
A Fully Homomorphic Encryption Scheme

Fully Homomorphic Encryption (FHE)

- Many schemes now, usually abbreviated by the first letters of the last names of the authors
- Different security assumptions (not factoring or discrete log)
  - Lattice problems: Learning with errors, …

Examples:

- First construction by Gentry in 2009
- E.g. FV, BGV, or DGHV (not used in practice)

The **DGHV** Fully Homomorphic Encryption Scheme

- FHE scheme whose security is based on the difficulty of the **approximate greatest common divisor** (AGCD) problem.
  - Finding the greatest common divisor of a set of integers that are close to multiples of a secret integer.

Fully Homomorphic Encryption over the Integers

Marten van Dijk
MIT

Craig Gentry
IBM Research

Shai Halevi
IBM Research

Vinod Vaikuntanathan
IBM Research

June 8, 2010

https://medium.com/@j248360/explaining-the-dghv-encryption-scheme-1acb6cd74dd6
https://www.esat.kuleuven.be/cosic/blog/co6gc-homomorphic-encryption-part-1-computing-with-secrets/
https://github.com/coron/fhe

Consider Simplified DGHV (not used in practice)

- $\boldsymbol{m} \in \{0, 1\}$
- Secret key: prime $\boldsymbol{p}$

Consider Simplified DGHV (not used in practice)

- $\boldsymbol{m} \in \{0, 1\}$
- Secret key: prime $\boldsymbol{p}$

- **Encryption**
    - Choose $\boldsymbol{q}$, $\boldsymbol{r}$ such that $r < p$ → $\boldsymbol{r}$ is random noise
    - $\boldsymbol{c} = \boldsymbol{q}.\boldsymbol{p} + 2.\boldsymbol{r} + \boldsymbol{m}$

Consider Simplified DGHV (not used in practice)

- $\boldsymbol{m} \in \{0, 1\}$
- Secret key: prime $\boldsymbol{p}$

- **Encryption**
  - Choose $\boldsymbol{q}$, $\boldsymbol{r}$ such that $r < p$ → $\boldsymbol{r}$ is random noise
  - $\boldsymbol{c} = \boldsymbol{q}.\boldsymbol{p} + 2.\boldsymbol{r} + \boldsymbol{m}$

- **Decryption**
  - $\boldsymbol{m} = \boldsymbol{c} \bmod 2 \oplus (\lfloor \boldsymbol{c}/\boldsymbol{p} \rceil \bmod 2)$

Computing with Simplified DGHV

- **Ciphertexts**
  - $c_1 = q_1.p + 2.r_1 + m_1$
  - $c_2 = q_2.p + 2.r_2 + m_2$

Computing with Simplified DGHV

- **Ciphertexts**
  - $c_1 = q_1.p + 2.r_1 + m_1$
  - $c_2 = q_2.p + 2.r_2 + m_2$
- **Addition**
  - $c_1 + c_2 = (q_1+q_2).p + 2.(r_1+r_2) + m_1 + m_2$

Note that noise grows **linearly**

Computing with Simplified DGHV

- **Ciphertexts**
  - $c_1 = q_1.p + 2.r_1 + m_1$
  - $c_2 = q_2.p + 2.r_2 + m_2$
- **Addition**
  - $c_1 + c_2 = (q_1+q_2).p + 2.(r_1+r_2) + m_1 + m_2$
- **Multiplication**
  - $c_1 \cdot c_2 = q'.p + 2.r' + m_1.m_2$
    - $r' = 2.\mathbf{r_1.r_2} + r_1.m_2 + r_2.m_1$
    - $q' = q1 \cdot q2 \cdot p + q1 \cdot m2 + q2 \cdot m1$

Note the increased growth of the noise. (no longer linear). One gets a new ciphertext with noise **roughly twice larger** than in the original ciphertexts c1 and c2.

The bootstrapping problem in FHE

Bootstrapping… in Fully HE Schemes

- If $\boldsymbol{r} > \boldsymbol{p}/2 \Rightarrow$ decryption fails on DGHV
  - Also a problem for other schemes.

- If the noise **grows too much**, it can **corrupt** the encrypted data and make it unusable

- Each operation **increases the noise**, so one must **control** this growth

Bootstrapping… in Fully HE Schemes

- To obtain a FHE scheme, (i.e. unlimited addition and multiplication on ciphertexts), one must **reduce** the amount of noise in a ciphertext

- **Bootstrapping** is a procedure that reduces noise to it's initial lenght
  - Still, bootstrapping is slow in most fully HE schemes
  - Thus, w/ fully HE, aim to avoid subsequent multiplications
  - DGHV does not have bootstrapping

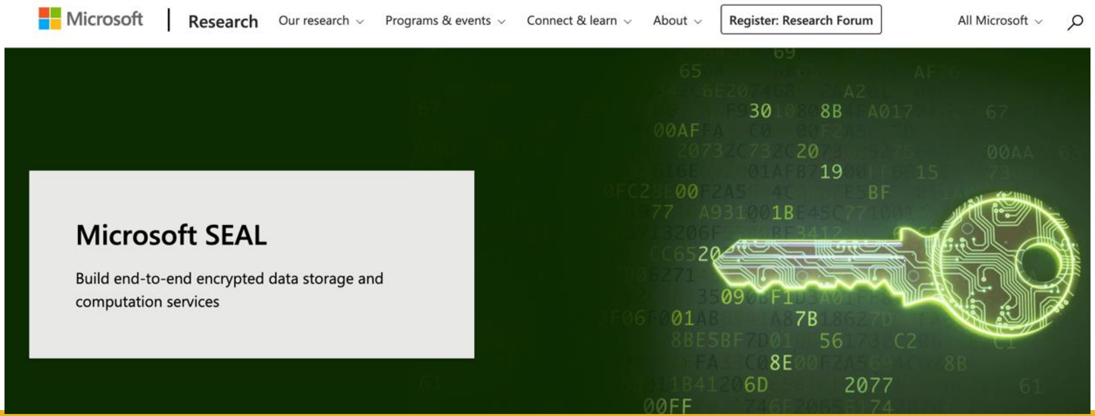
Practical FHE Schemes

- **FV, BGV, BFV, CKKS**
  - Lattice-based encryption schemes
  - Encrypt vectors (usually as polynomials)

- **TFHE**
  - Fully HE over the Torus
  - Usually encrypts bits
  - Very fast bootstrapping (frequently performed)
  - https://tfhe.github.io/tfhe/

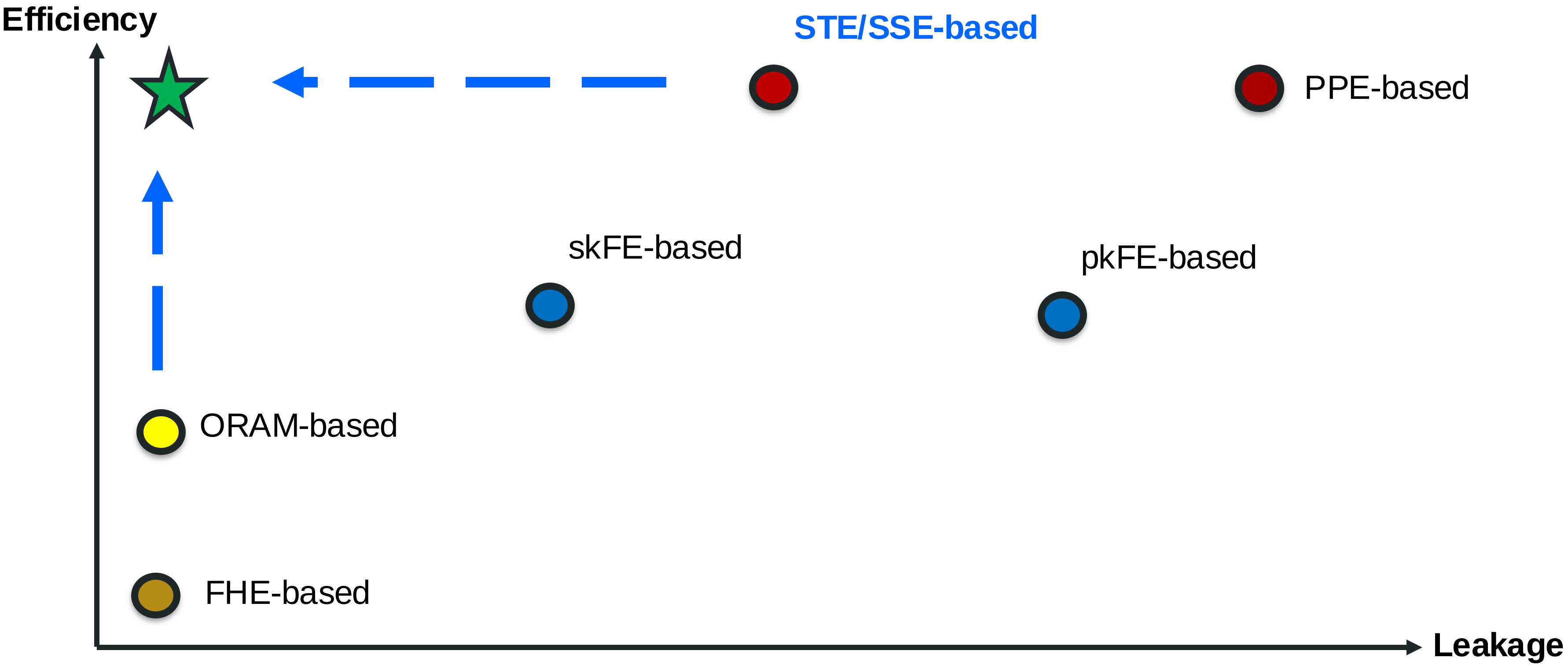
Try it… on your own ☺

- Download Microsoft's SEAL library and hack away!
  - https://www.microsoft.com/en-us/research/project/microsoft-seal/
  - Create a key
  - Encrypt two 8 bit numbers bit-wise using batch encoding (allows rotation)
  - Perform comparison, for each position: If prefix is equal and bits are different, output 1 if bit of first number is 1; else output 0
  - Decrypt result



Encrypted Search Algorithms

Tradeoffs: Efficiency vs. Security

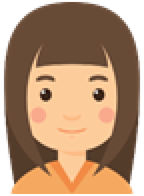


Cryptographic Mechanisms

- Query from an authenticated client

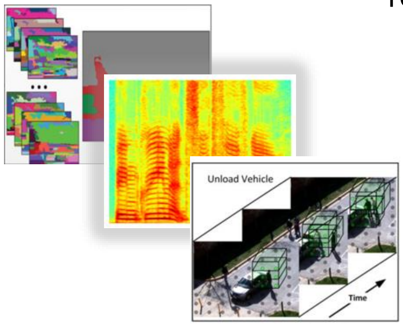
- Schemes to search over encrypted data
- Server-side processing of encrypted data
- Server knows nothing* about the data

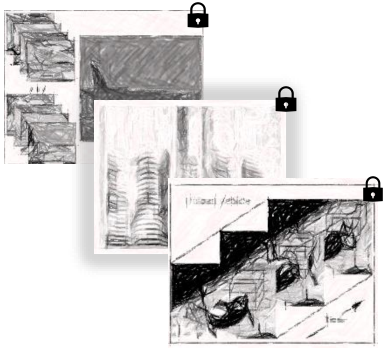
Setup(1^k, DS) $\Longrightarrow$ (K, EDS)



Token(K, q) $\Longrightarrow$ tk

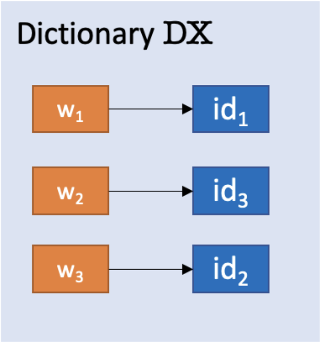
Query(EDS, tk) $\Longrightarrow$ ans





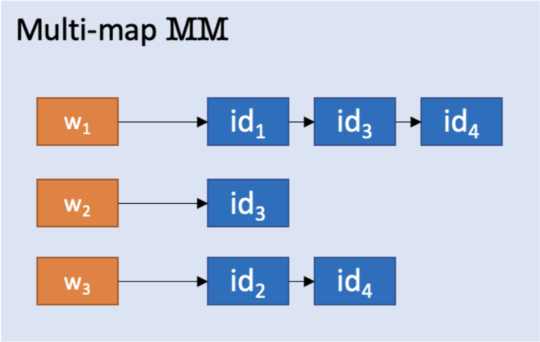
Examples of Data Structures

- Dictionaries map labels to values



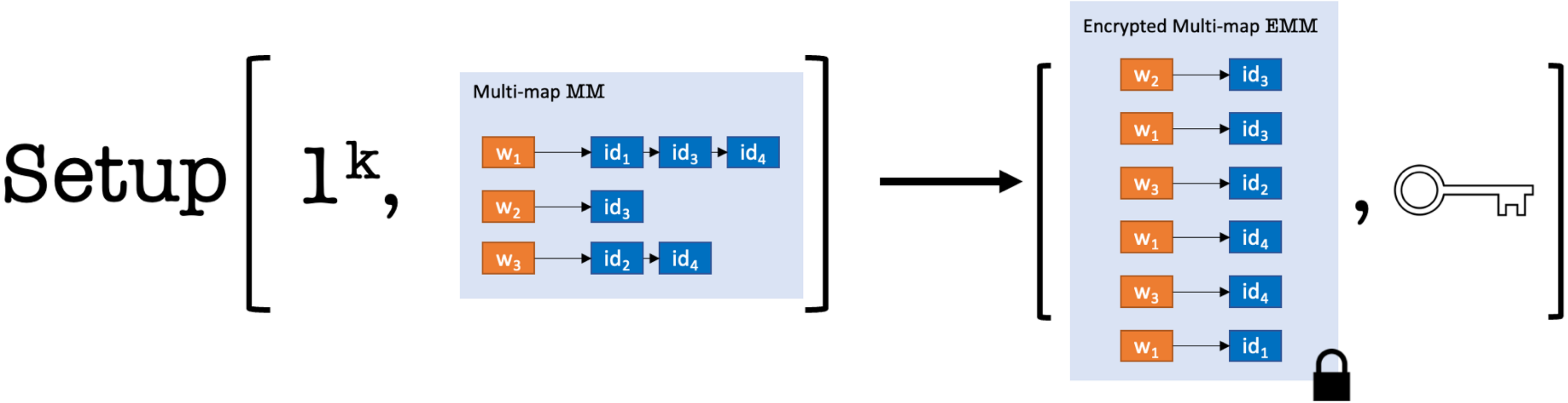
- Get: DX[w_3] returns id_2

- Multi-maps map labels to tuples

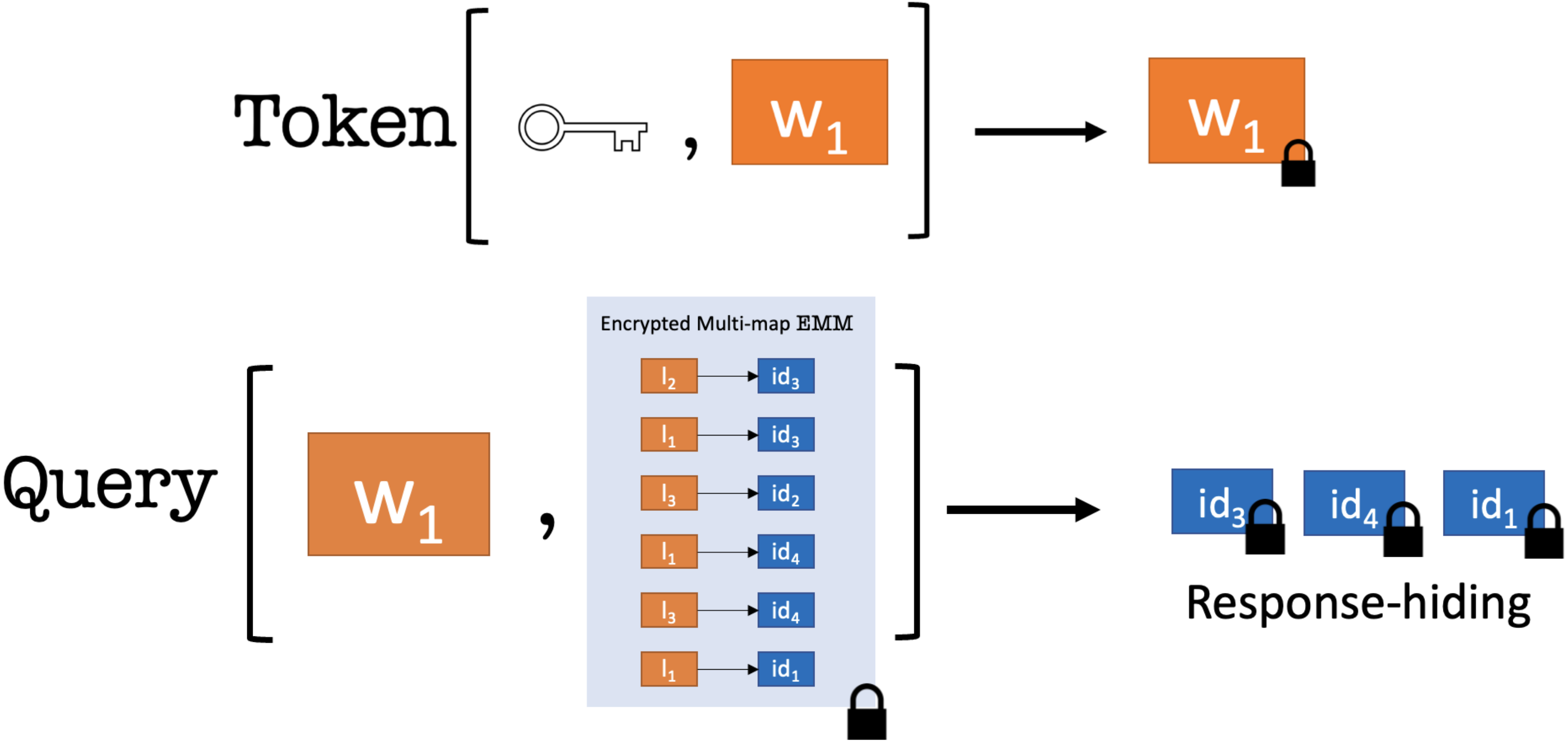


- Get: MM[w_3] returns (id_2 , id_4)

Examples of Data Structures



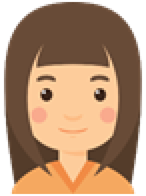
Examples of Data Structures



Encrypted Search Algorithms (ESAs)

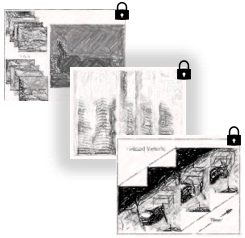
Encrypted Search Algorithms (ESAs)

Trusted Client



Untrusted Server

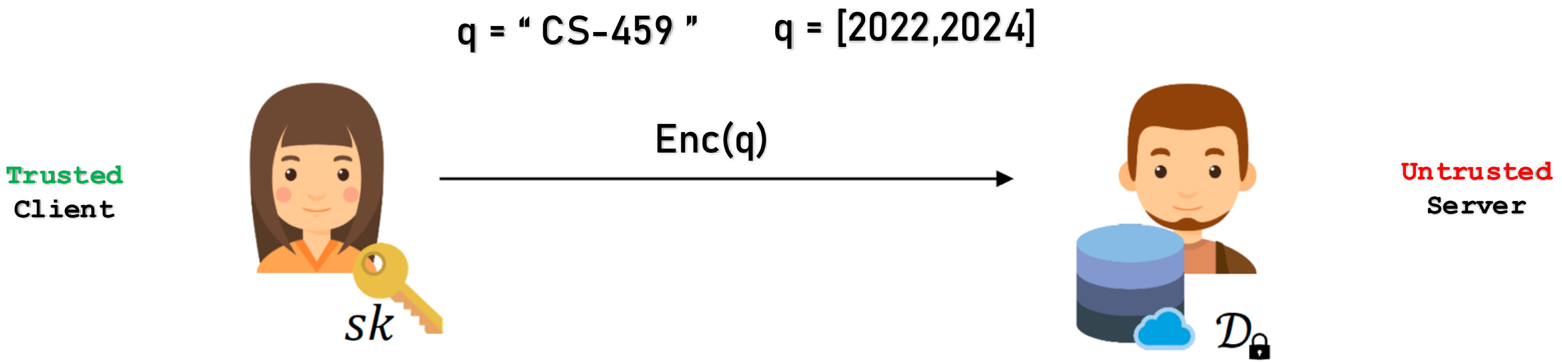
Data



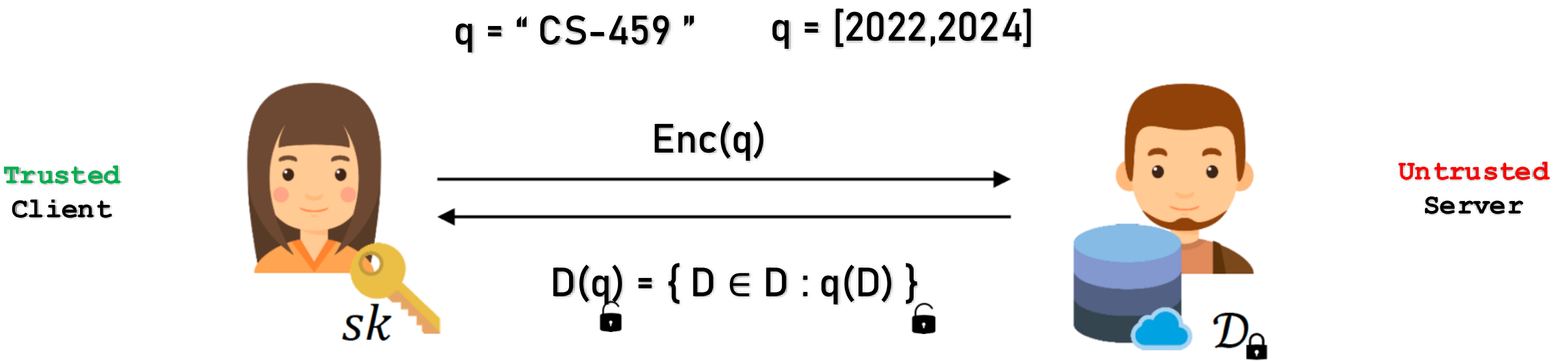
Enc(Search Index)

Encrypted Search Algorithms (ESAs)

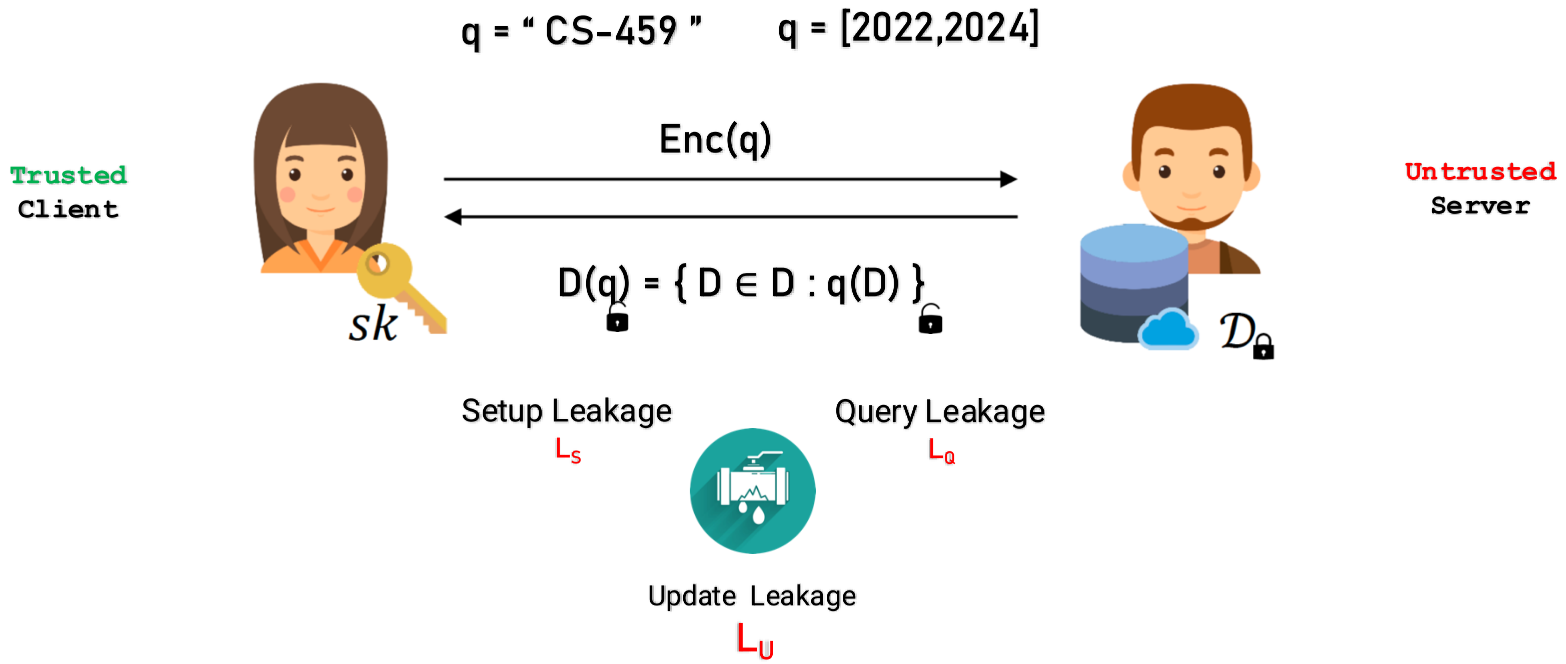
Encrypted Search Algorithms (ESAs)



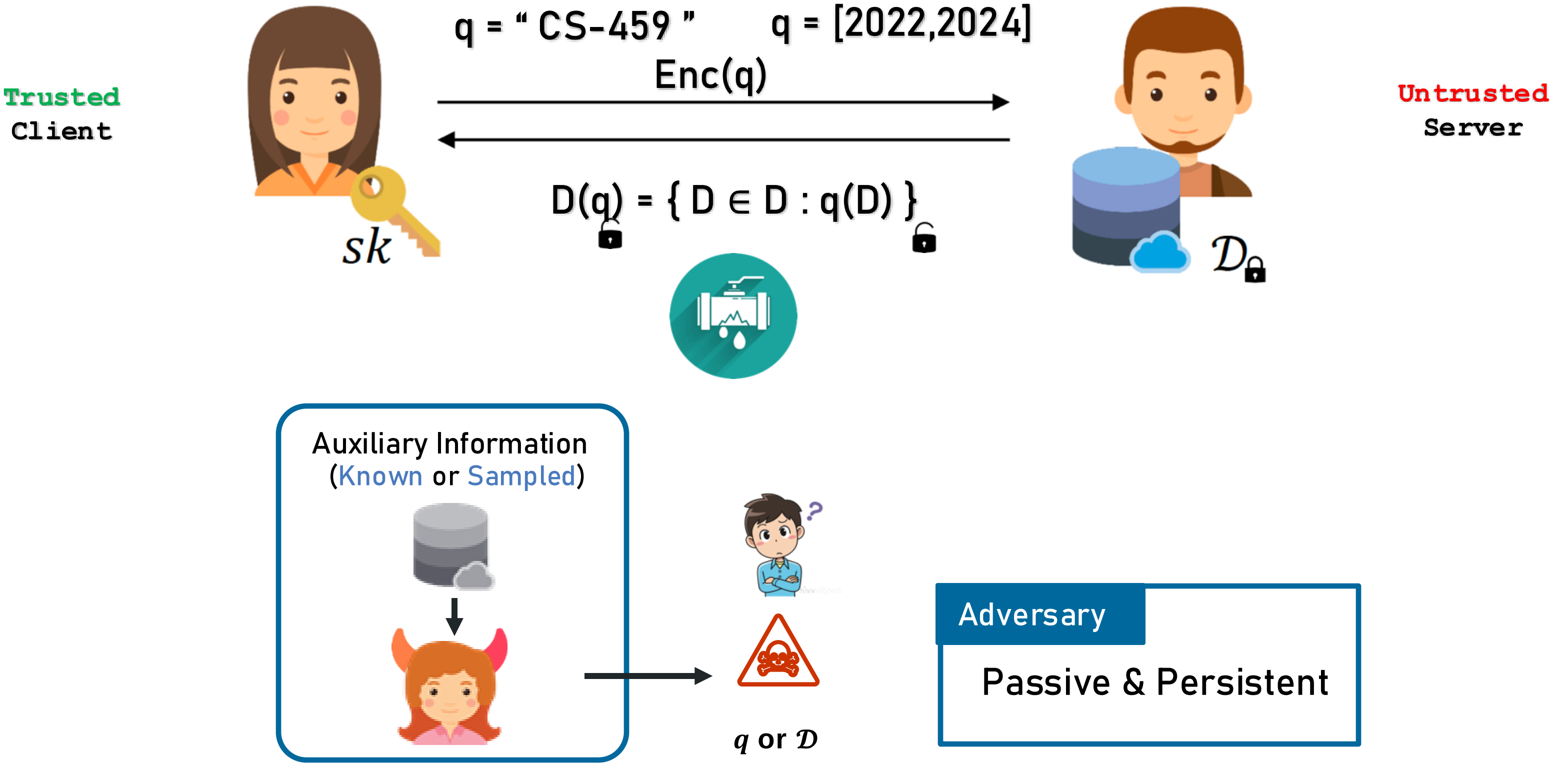
Encrypted Search Algorithms (ESAs)



Encrypted Search Algorithms (ESAs)



Encrypted Search Algorithms (ESAs)



How do we model leakage?

- The "Baseline" leakage profile for response-revealing EMMs
  - ✓ $(\mathcal{L}_S, \mathcal{L}_Q, \mathcal{L}_U)$ = (dsize, (qeq, rid), usize)
- The "Baseline" leakage profile for response-hiding EMMs
  - ✓ $(\mathcal{L}_S, \mathcal{L}_Q, \mathcal{L}_U)$ = (dsize, qeq, usize)
- There exists several new constructions with better leakage profiles
  - ✓ AZL and FZL [Kamara-Moataz-Ohirimenko'18]
  - ✓ VHL and AVHL [Kamara-Moataz'19]

Leakage	Information
Response Length	$\lvert D(q) \rvert$
Query Equality	$q_i = q_j$
Co-Occurrence	$\lvert D(q_i) \cap D(q_j) \rvert$
Response Identity	$\{i : D_i \in q(D)\}$
Response Volume	$\{\lvert D_i \rvert_b : D_i \in q(D)\}$

(Simplified)

Leakage Attacks Types

Keyword (point) queries

[IKK12,CGPR15,BKM20,RPH21]

Keyword	Document IDs
'Encrypted'	2,5,11,13,20,31
'systems'	3,5,10,11,13,25
'lab'	5,11,21,27

$$q = w$$
$$\mathcal{D}(q) = \{D \in \mathcal{D}: q \in D)\}$$
Recover q

$q =' \text{Defense}'$

Range queries

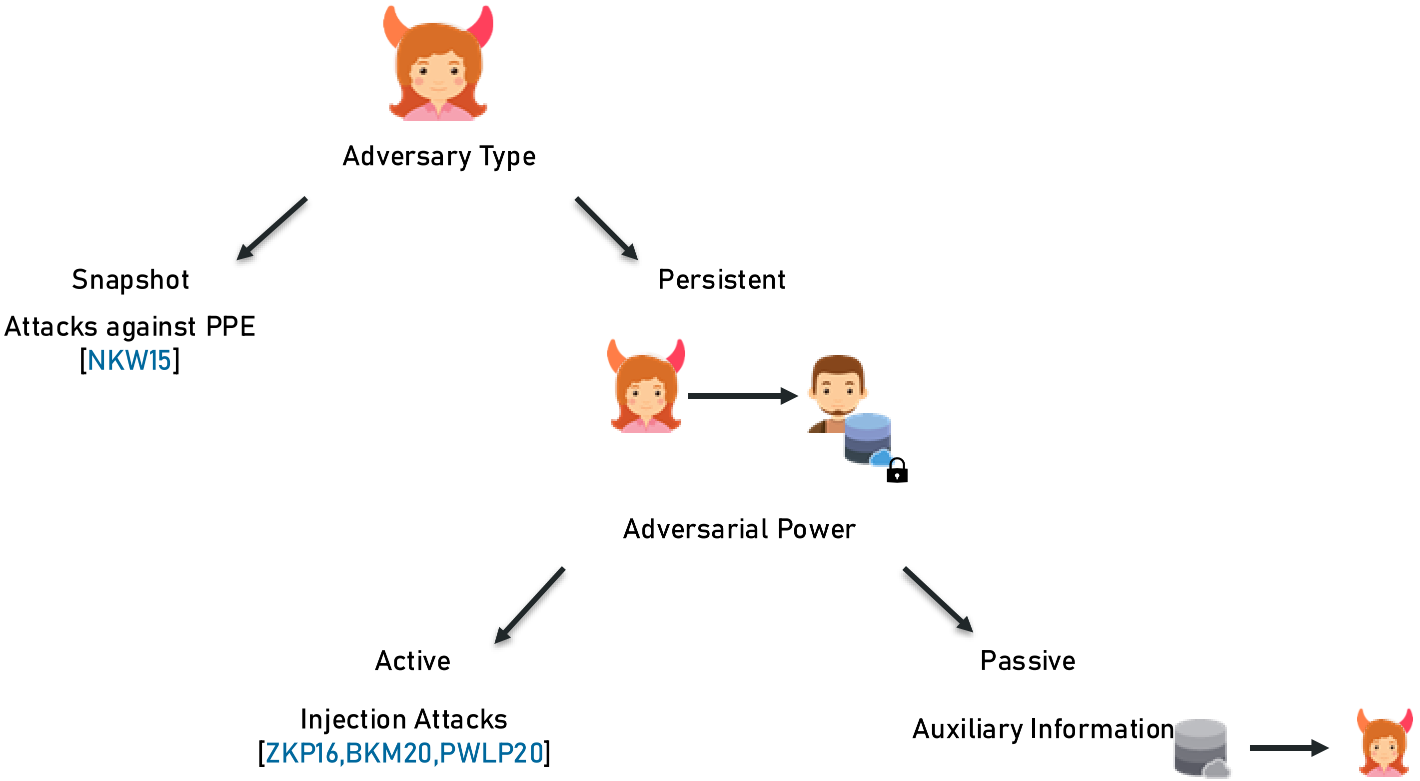
[KKNO16,LMP18,GLMP18, GLMP19,GJW19,KPT20,KPT21]

ID	Age
1	65
2	7
3	27

$$q = (a, b)$$
$$\mathcal{D}(q) = \{r \in \mathcal{D}: a \leq r \leq b\}$$
Recover $\mathcal{D}$

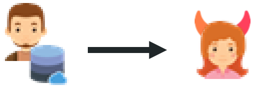
$q = (18,39)$

Leakage Attacks against ESAs



Leakage Attacks against ESAs

Persistent Passive



Auxiliary Information

Sampled-data or sampled-query

Keyword & Range attacks

[LZWT14,LMP18,GLMP18,GJW19,OKa21,DHP21, GPP21,OKb21]

Known-data

Keyword attacks

[IKK12*,CGPR15,BKM20, RPH21]

$q = w$

$\mathcal{D}(q) = \{D \in \mathcal{D}: q(D)\}$

Recover q

None

Range attacks

[KKNO16,LMP18,GLMP18, GLMP19,GJW19,KPT20, KPT21]

$q = (a, b)$

$\mathcal{D}(q) = \{r \in \mathcal{D}: a \leq r \leq b\}$

Recover $\mathcal{D}$

ESA Techniques Overview

Technique	Leakage	Query Time
Fully Homomorphic Encryption (FHE)	• None	Linear
Oblivious RAM (ORAM)	• Response Length + Volume	Sublinear
Structured Encryption (STE)	• Query Equality • Response Identities + Volumes	Optimal
Property-Preserving Encryption (PPE)	• Ciphertext Equality • Ciphertext Order • All STE leakage	Optimal

FHE: Considered secure but inefficient

ORAM and STE (Our work): Considered efficient and Has some leakage

PPE: Considered efficient but provides low level of security [NKW15]

Uncertainty Of Security

Constructions

"Benign leakage"

"Common leakage"

"Standard leakage"

"Accepted leakage"

"[Attacks] assume extremely strong adversarial models"

"Leakages [...] are not exploitable via leakage-abuse attacks in practice"

Attacks & Countermeasures

"Severe threat"

"Devastating results"

"[ESAs] are extremely vulnerable to [attacks]"

"[ESA] schemes should no longer be used without countermeasures"

"Our assumptions on background information are weak"

"With some prior knowledge [...] an honest-but-curious server can recover the underlying keywords"

Uncertainty Of Security

Constructions

"Benign leakage"

"Co…"

"Standard leakage"

"Accepted l…"

"[Attacks] assume extremely strong adver… models"

"Leakages […] are not exploitable via leakage-… attacks in practice"

Attacks & Countermeasures

"Severe threat"

"Devastating results"

"[ESAs] are extremely vulnerable to [attacks]"

"[ESA] schemes should no longer be used without countermeasures"

"Our assumptions on background information are weak"

"With some prior knowledge […] an honest-but-curious server can recover the underlying keywords"

Encrypted Search Algorithms: Real-World Deployments.

[Always Encrypt '15]
- Encrypted Relational Database (ERDs)
- Property-Preserving Encryption (PPE)

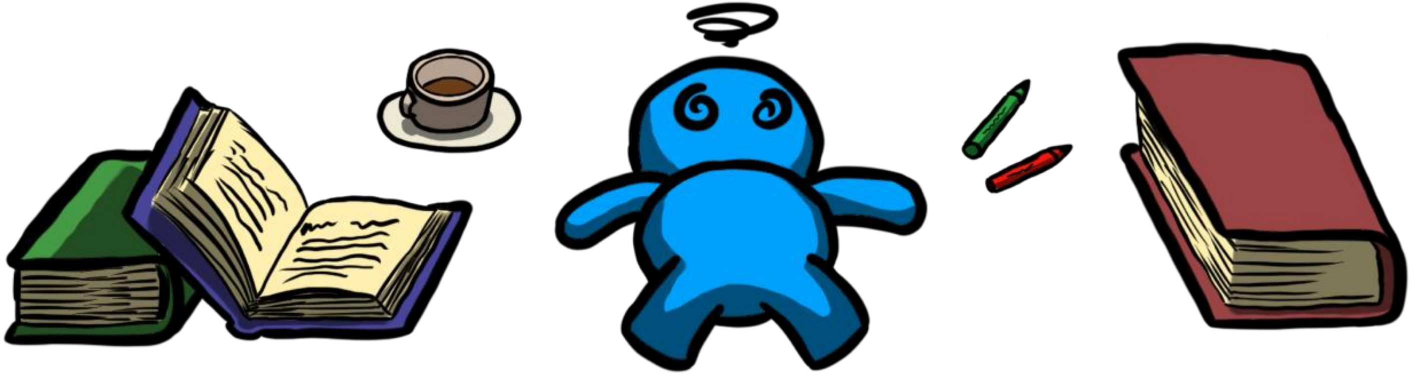
[Client-Side Field Level Encryption'19]
- Encrypted Non-Relational Database (EnRDs)
- Property-Preserving Encryption (PPE)

[Queryable Encryption'23]
- **Encrypted Non-Relational Database (EnRDs)**
- **Structured Encryption (STE)**

[Document Encryption'23]
- Encrypted Non-Relational Database (EnRDs)
- Property-Preserving Encryption (PPE)

A Few Announcements

- Assignment 3 is <u>due today</u> 4pm
  - No-penalty late policy period until Saturday 4pm



- Student Course Perceptions – Available until Dec 3
  - https://perceptions.uwaterloo.ca/

Thanks for tagging along!