IN THE NAME OF GOD

LATTICE-BASED SEARCHABLE ENCRYPTION FOR CLOUD STORAGE

POURIA DADKHAH

DEPT. OF ELECTRICAL ENGINEERING

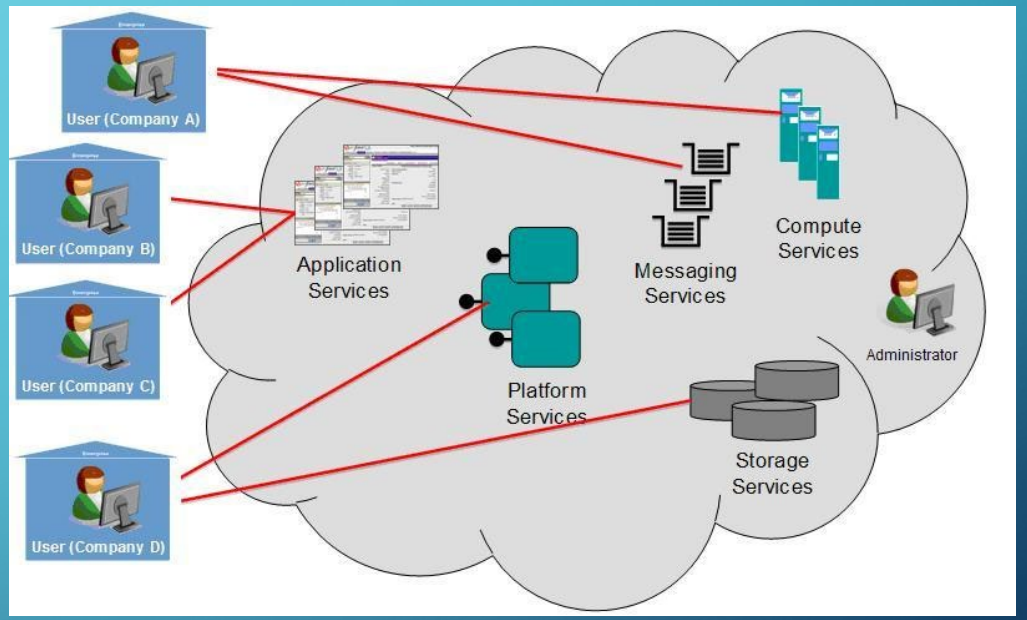
POURIA.DADKHAH@GMAIL.COM

SUPERVISOR: DR. T. EGHLIDOS

Table of Contents

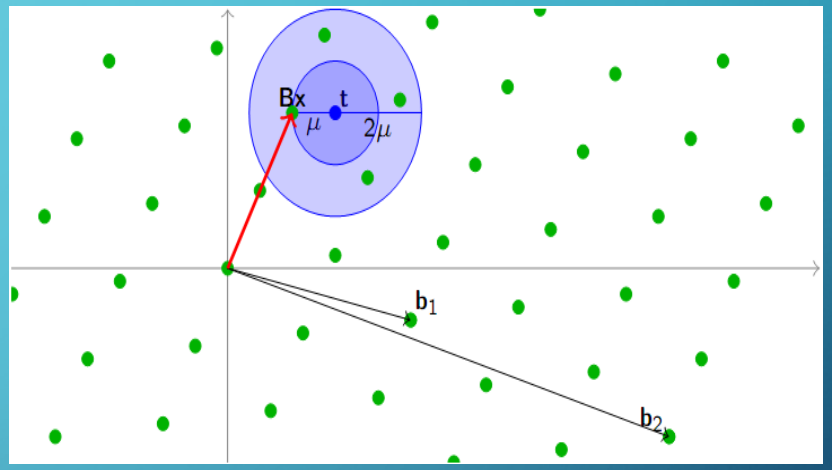
Review Prerequisites

- Motivation
  - Why need Encryption on Cloud?
- Tools and Requirements
  - Post-Quantum tools
- Path
  - Lattices
  - searchable algorithms
- implement a cryptosystem



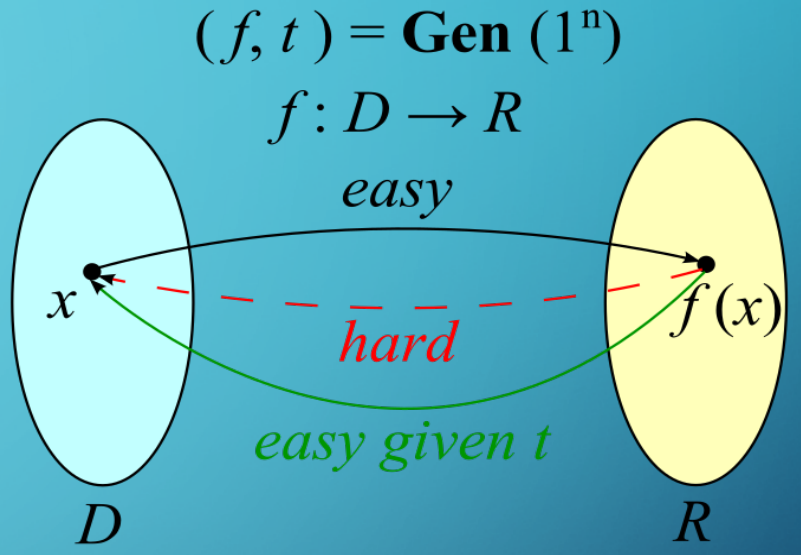
Review Prerequisites

- What is a Lattice ?
  - $\mathcal{L}(b_1,\ldots,b_n) \triangleq \{\sum_{i=1}^{n} x_i b_i \mid x_i \in \mathbb{Z}\}$
- Simple Operation and implementation
- Hard problems
  - SVP , CVP
    - ✓ $\forall y \in \mathbb{Z}^n \backslash \{0\} : y \neq x \Longrightarrow \|Bx\| \leq \|By\|$
    - ✓ $\forall y \in \mathbb{Z}^n : y \neq x \Longrightarrow \|Bx - t\| \leq \|By - t\|$
    - ✓ $\forall y \in \mathbb{Z}^n : y \neq x \Longrightarrow \|Bx - t\| \leq \gamma \|By - t\|$
- Ajtai Lattices
  - ✓ $A \in Z_q^{n \times m}$ with Uniform Dist.
  - ✓ $\Lambda_q^{u}(A) = \{e \in \mathbb{Z}^m : A.e = u \ (mod\ q)\ for\ some\ u \in \mathbb{Z}^n\}$



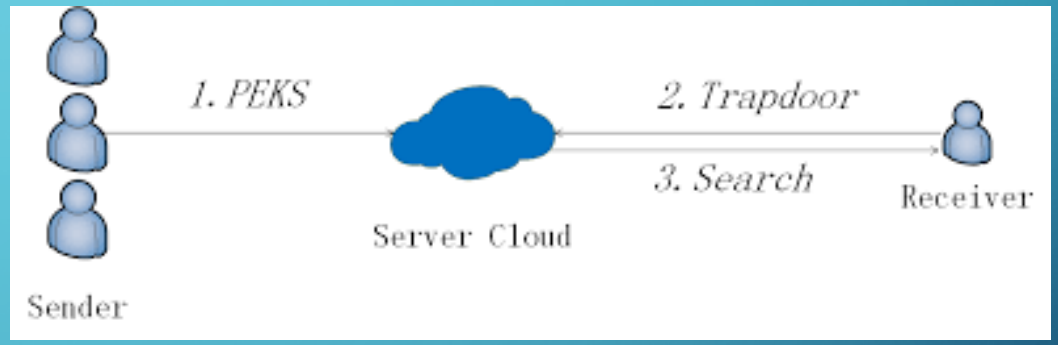
Review Prerequisites

- Trapdoors
- Learning With Errors (LWE) Decision Problem
  - Parameters
    - ✓ m, n, q ∈ $\mathbb{Z}$
    - ✓ $\chi \sim N(\mu, \sigma^2)$
    - ✓ $A \in \mathbb{Z}_q^{m \times n}$ Uniform Dist.
  - Algorithm
    - ✓ (A, v)
    - ✓ v $\begin{cases} v \sim U(\mathbb{Z}_q^m) \\ v = As + e \end{cases}$ $That: s \in \mathbb{Z}_q^n, e \in \mathbb{Z}_q^m \sim \chi^m$
    - ✓ Distinguish v ?



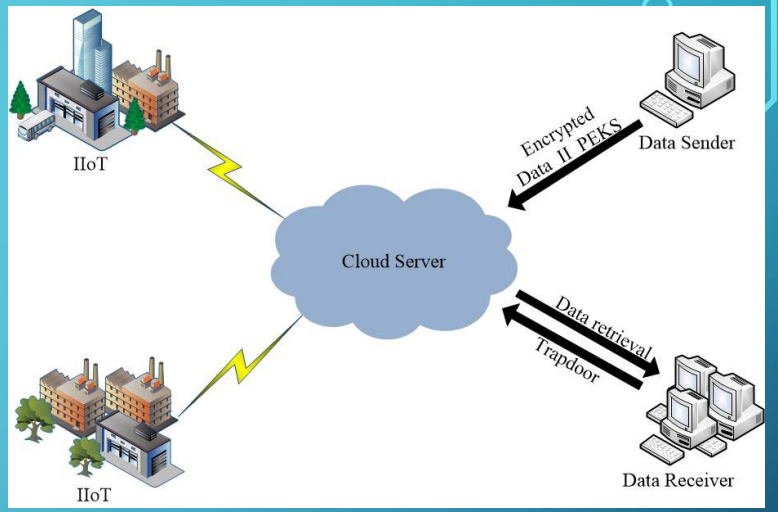
What is PEKS

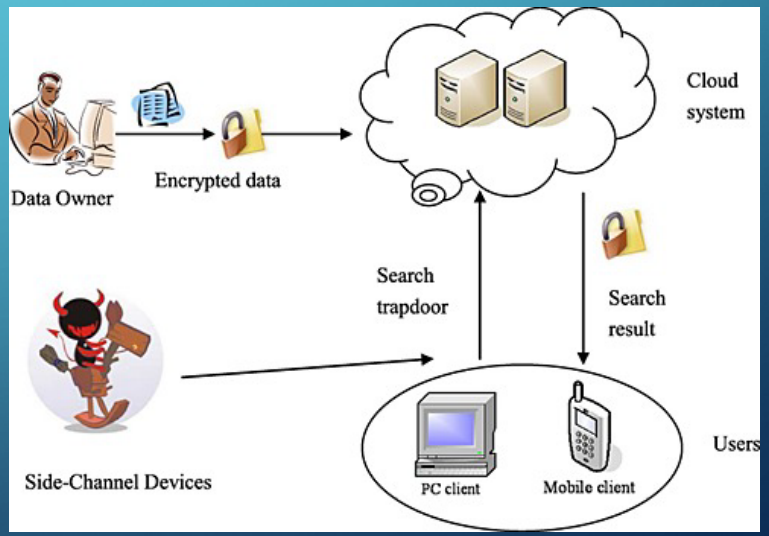
- Goal
  - Shared storage
  - Security
  - Keyword search
- Approach
- Challenges
  - Hardness assumptions
  - End-to-end computation delay
  - Key exposure
  - …



Secure cloud communication

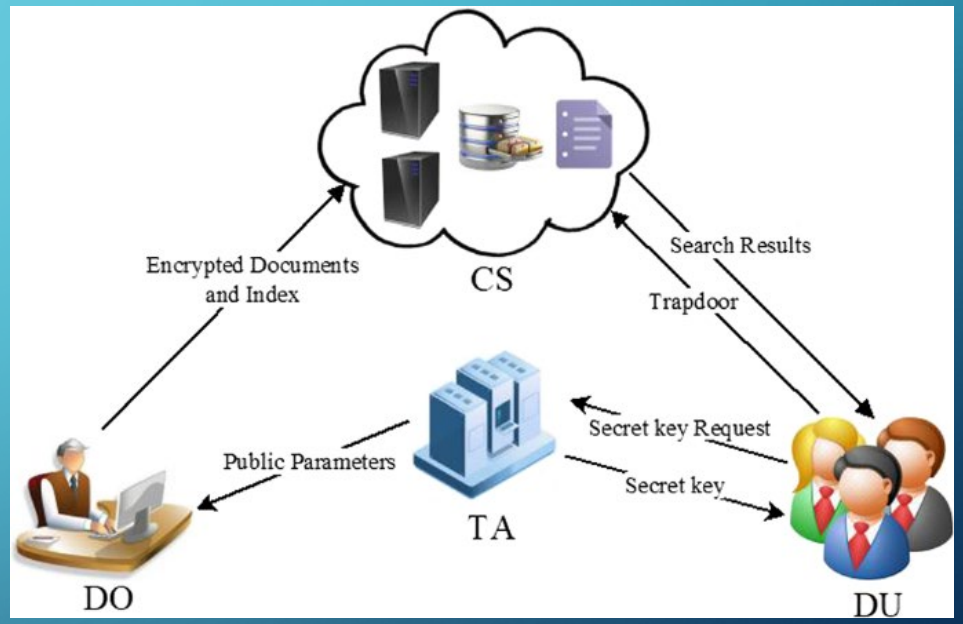
- Data Sender
  - $E(M, Pk_r) \,||\, PEKS(w_1, Pk_r) \,||\, PEKS(w_2, PK_r) \,||\, \ldots$
  - Send to server
- Data Receiver
  - $W \text{ --- } Sk_r \text{ --- } > T_w$
  - Send to server
- Cloud Server
  - Input: T_w
  - Test Algorithm over data
  - Output: Corresponding data to receiver





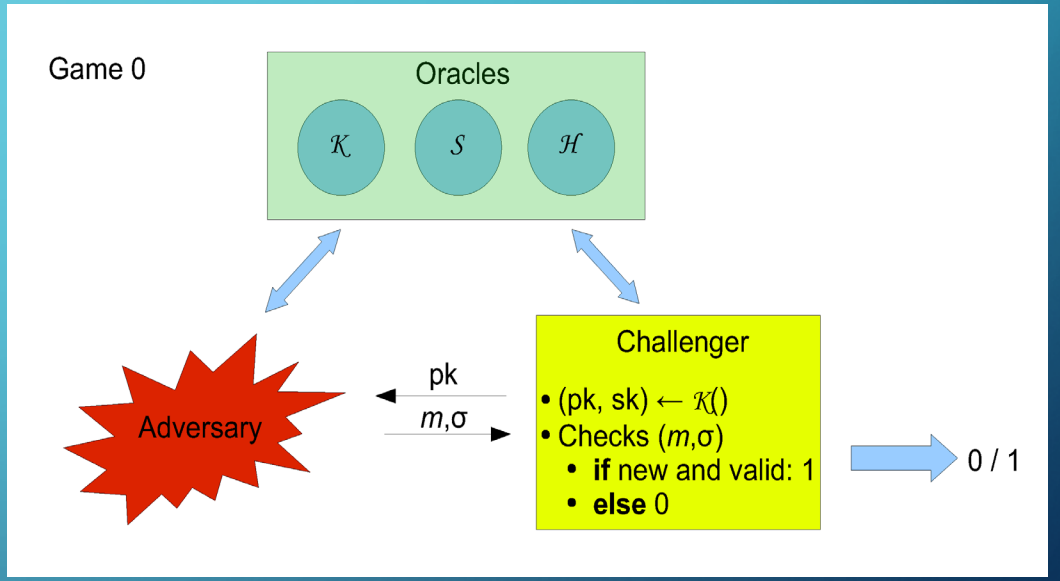
Primitive PEKS scheme

- Setup
  - Input: Parameter Security κ
  - Output: system public parameter Σ, Pk, Sk
- PEKS
  - Inputs: Σ, Pk_r , keyword w
  - Outputs: PEKS CT_w
- Trapdoor
  - Inputs: Σ, (Pk_r , Sk_r), w
  - Outputs: Trapdoor T_w
- Test
  - Inputs: Trapdoor T_w , PEKS CT
  - Outputs: 1 if CT and T_w contain the same keyword w, and 0 otherwise.



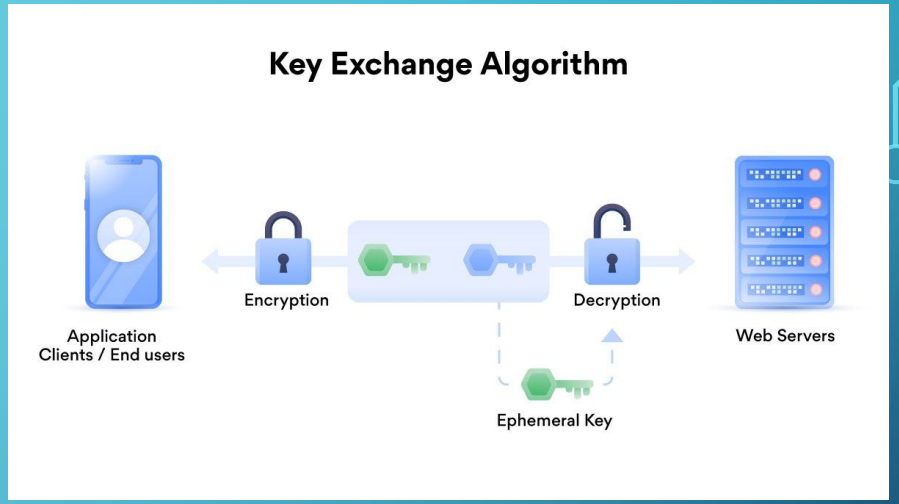
security model

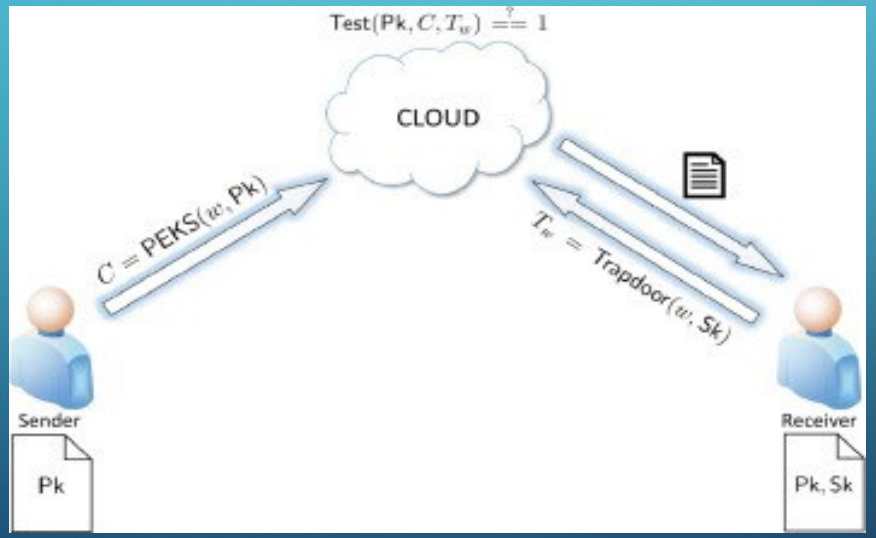
- Setup
  - Challenger C runs KeyGen: key pair (Pk, Sk)
  - Gives Pk to Adversary A
- Trapdoor oracle
  - A Choose keyword w, ask for Trapdoor T_w
- Challenge Phase
  - A choose (w_0^*, w_1^*) send to C
  - C choose a random bit b ∈ (0,1)
  - Send $CT_b^* = PEKS(w_b^*, Pk_r)$ to A
- Guess
  - A output b' ∈ (0,1)
- $Adv_A^C(\kappa) = |Pr(b'=b) - 1/2|$



Fs & IKGA

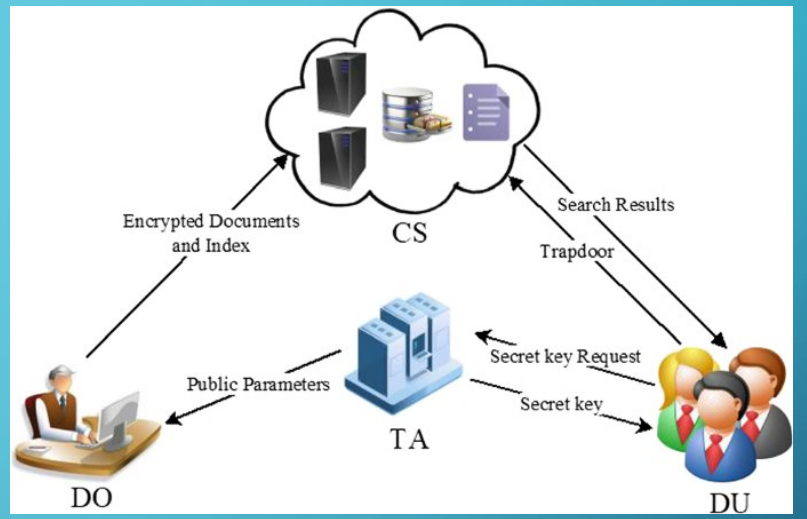
- Forward Security
  - Key exposure
  - Approach
- Keyword guessing attack
  - Problem
  - Approaches
- IKGA

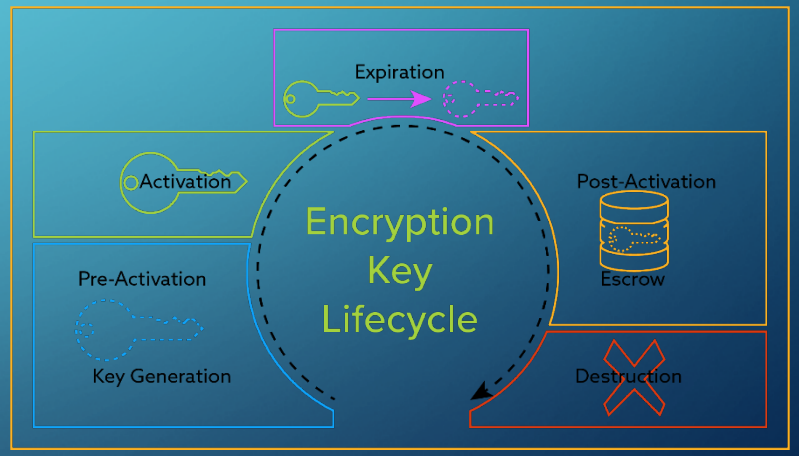




FS-PEKS scheme

- Setup
  - Input: Parameter Security κ
  - Output: system public parameter Σ, initial* Pk, Sk
- Key Update*
  - Inputs: ($Pk_{r||i}$, $Sk_{r||i}$)
  - Outputs: ($Pk_{r||j}$, $Sk_{r||j}$) (j> i)
- PEKS
  - Inputs: Σ, $Pk_{r||j}^{*}$, keyword w
  - Outputs: PEKS CT_j^{*}
- Trapdoor
  - Inputs: Σ, $(Pk_{r||j}, Sk_{r||j})^{*}$, w
  - Outputs: Trapdoor $T_{w||j}^{*}$
- Test
  - Inputs: Trapdoor $T_{w||j}^{*}$, PEKS CT_j
  - Outputs: 1 if CT and $T_{w||j}$ contain the same keyword w, and 0 otherwise.





Secure and efficient Algorithm

- Setup
  - Input : n
  - n, q -- (TrapGen) -- > $(A, T_A) \equiv (Pk, Sk)$
  - H1, H2
- Key Update
  - $R_{r||i} = H_1(A_{r||i}) \dots H_1(A_{r||1})$
  - $NewBasisDel(A_{r||i}; R_{r||i \to j}; T_{r||i}) = Sk_{r||j}$
- PEKS
  - $\gamma_j = (1; 1; \quad ; 1)$
  - $u = H_2(w)$, $s \in Z_q^n$
  - Output (p,c): $p = A_{r||j}^T . S + x$, $c = u^T .s + y$

Reminder

$A \in Z_q^{n \times m}$ with Uniform Dist.

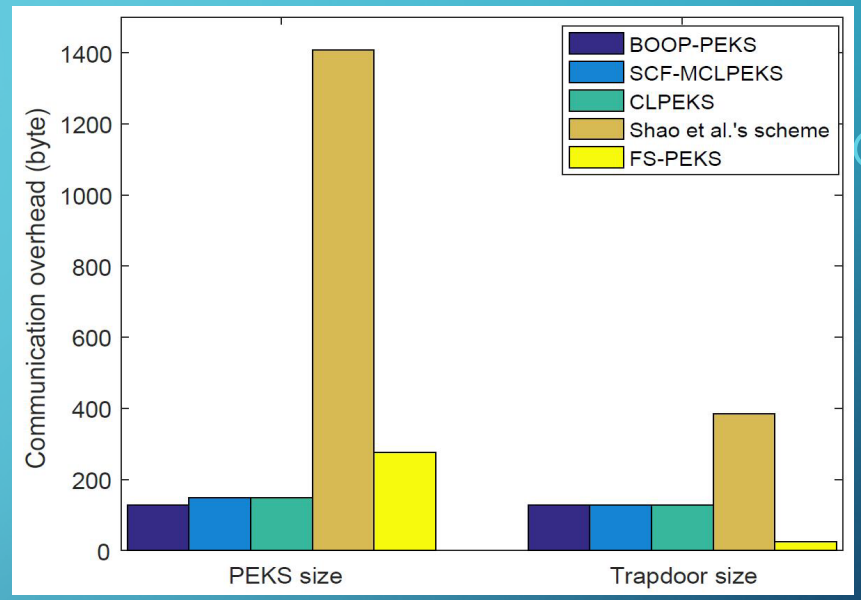
$\Lambda_q^u(A) = \{e \in \mathbb{Z}^m : A.e = u \ (mod\ q)\ for\ some\ u \in \mathbb{Z}^n\}$

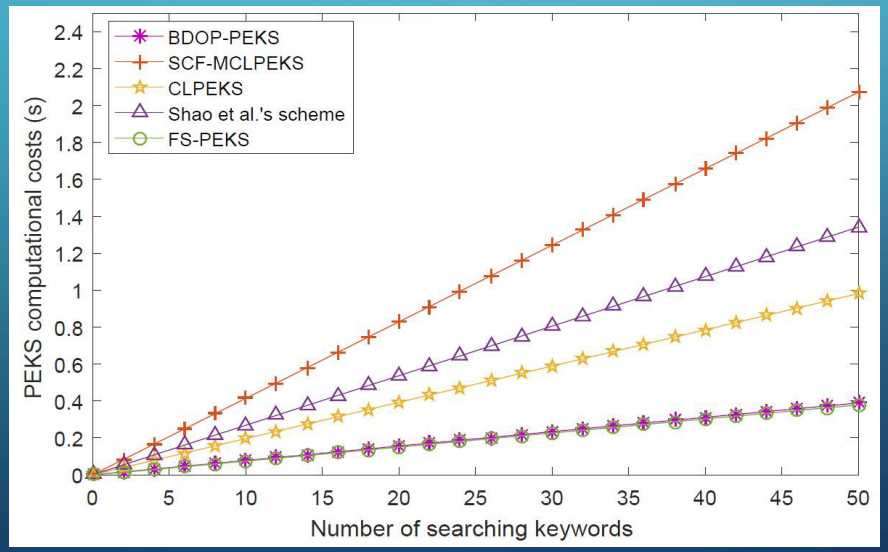
Secure and efficient Algorithm

- Trapdoor
  - $u = H(w)$
  - $t_w = \text{SamplePre}(A_{r||I}, u, T_A, \sigma)$; $A.t_w = u$
- Test
  - input: (p,c) , t_w
  - $b = c - e^T . p$
  - if $|b| < q/4$ output 1 o.w. 0
  - $b = (u^T - (A_{r||j}.e)^T).s + y - e^T .x$

Efficiency

- PEKS computation
  - Sum ($m.n \log(q)^2 + m.\log(q)$, $n.\log(q)^2 + \log(q)$)
- Key Update
  - $O(m \log (q)^2)$
- Total
  - $m.\log(q)^2 + \log(q)$





Further work & Conclusion

- ✓ Define Problem
  - ✓ cloud computing, Security, performance, challenges
- ✓ Threats and Tools
  - ✓ lattices
- ✓ design provable efficient scheme

- ✍ analysis combinatorial and new ideas
- ✍ Implementation
- ✍ Tester

Thanks For Your Attention