Recitation 9

Yifan Wang

Logistics

- Intermediate report
  - Feedback is expected by this weekend.
  - Address the concerns via emails or attending OHs.

- Projects
  - Technology workshop.
  - Spendings.

Privacy & Encryption

Trusted Execution Environments (TEEs)

- Intel:
  - Software Guard eXtensions (SGX)
  - Management Engine (ME)
- AMD:
  - Memory Encryption Techniques
  - Platform Secure Processor

SGX

2 major changes:

- enclave memory access semantics
- protection of the address mappings

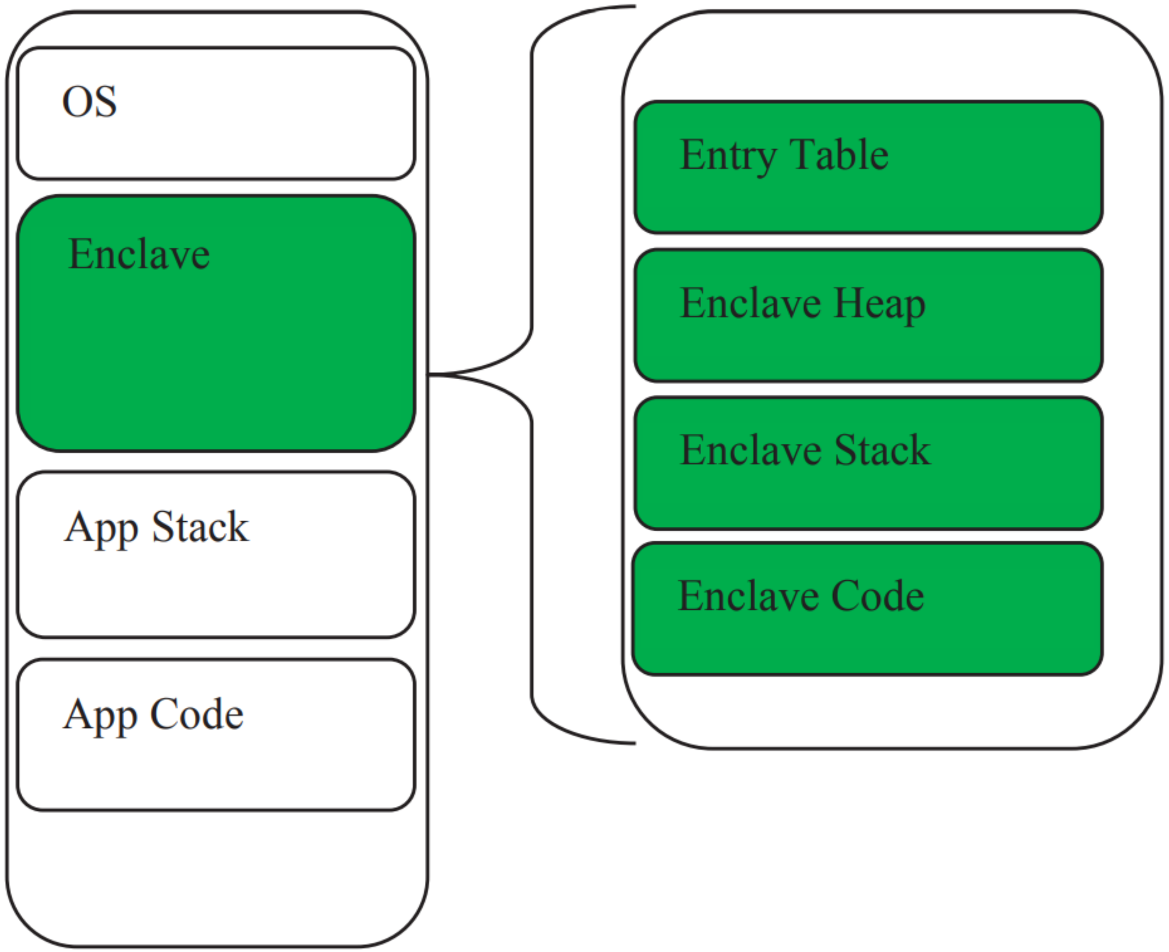
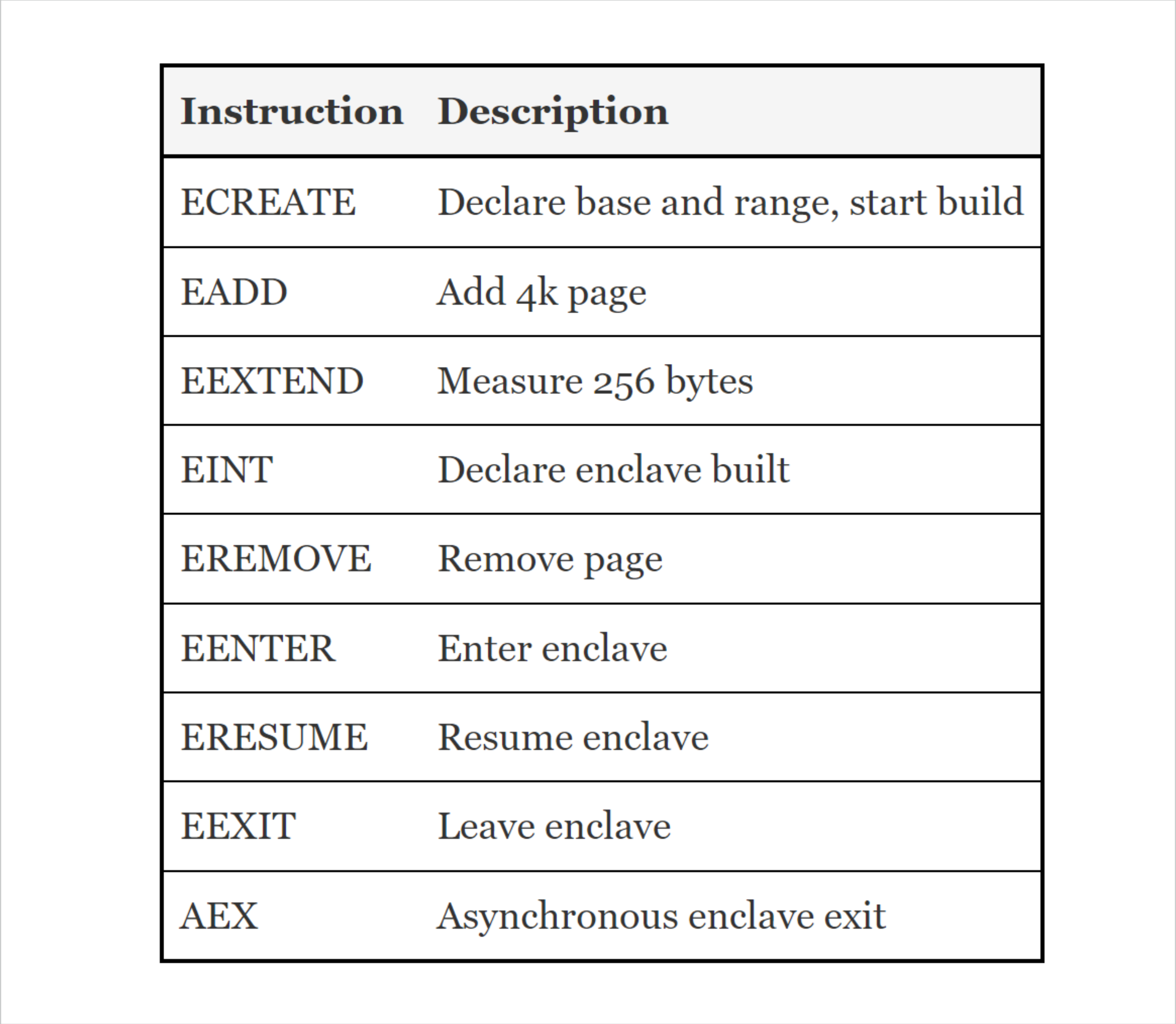


Figure 1: Enclave within Application's Virtual Address Space

SGX

protection of the address mappings

- Compiler support is needed.



Instruction	Description
ECREATE	Declare base and range, start build
EADD	Add 4k page
EEXTEND	Measure 256 bytes
EINT	Declare enclave built
EREMOVE	Remove page
EENTER	Enter enclave
ERESUME	Resume enclave
EEXIT	Leave enclave
AEX	Asynchronous enclave exit

SGX

protection of the address mappings

- Whether an access operation is from a processor running in the enclave mode.
- Whether a target physical address is in the EPC.
- Whether a target page belongs to the enclave (i.e., only the enclave code can access the enclave's data).
- (EPC = Enlave Page Cache)

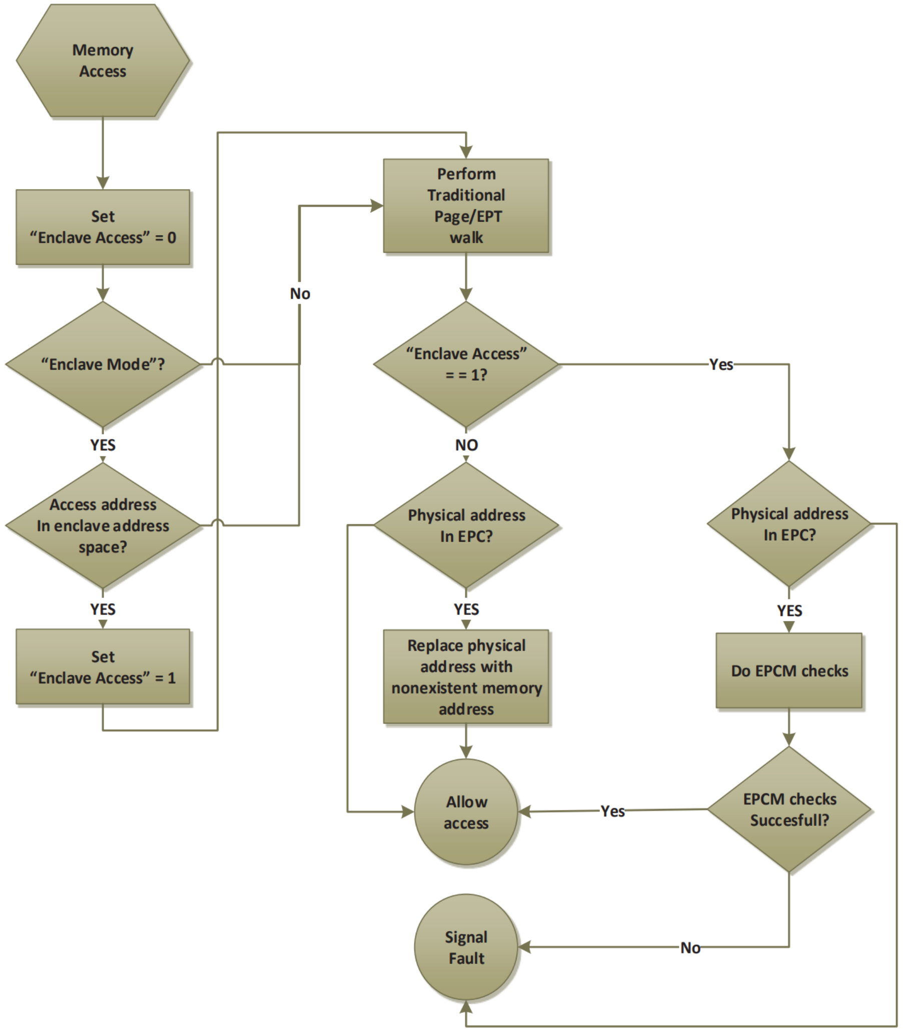
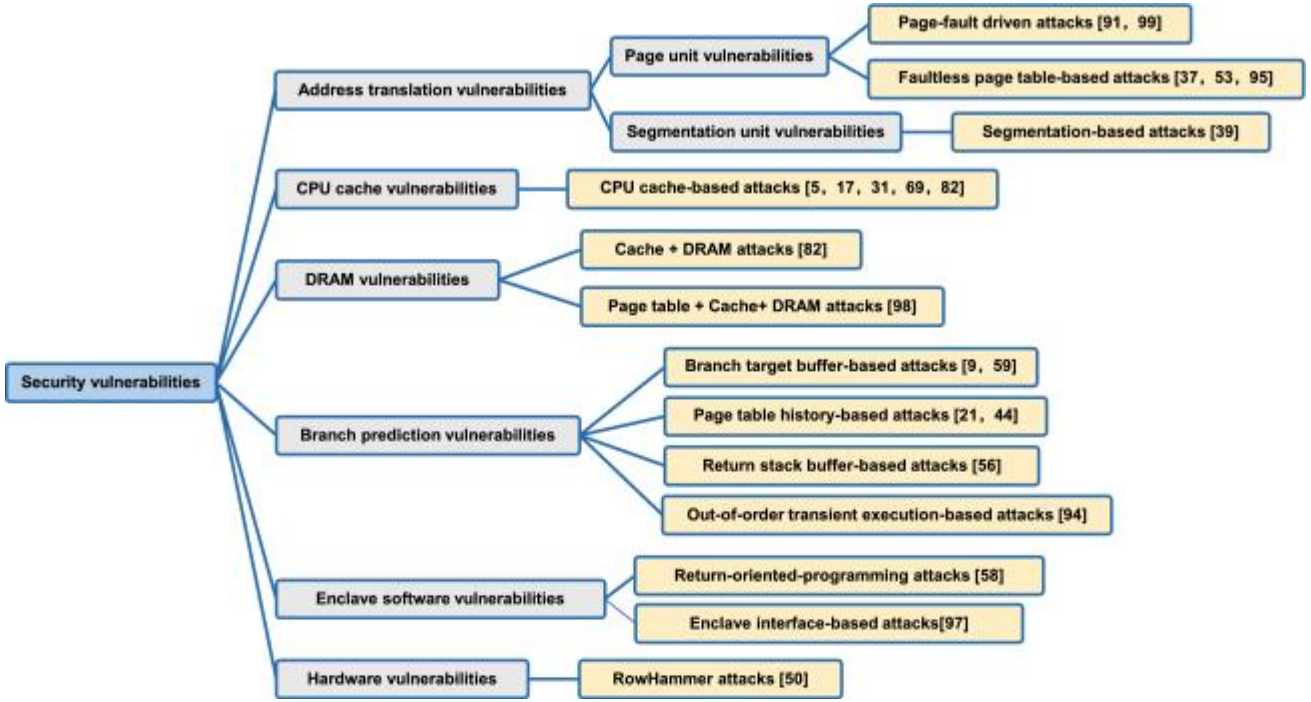


Figure 2 SGX Enclave Access Check

SGX Vulnerability

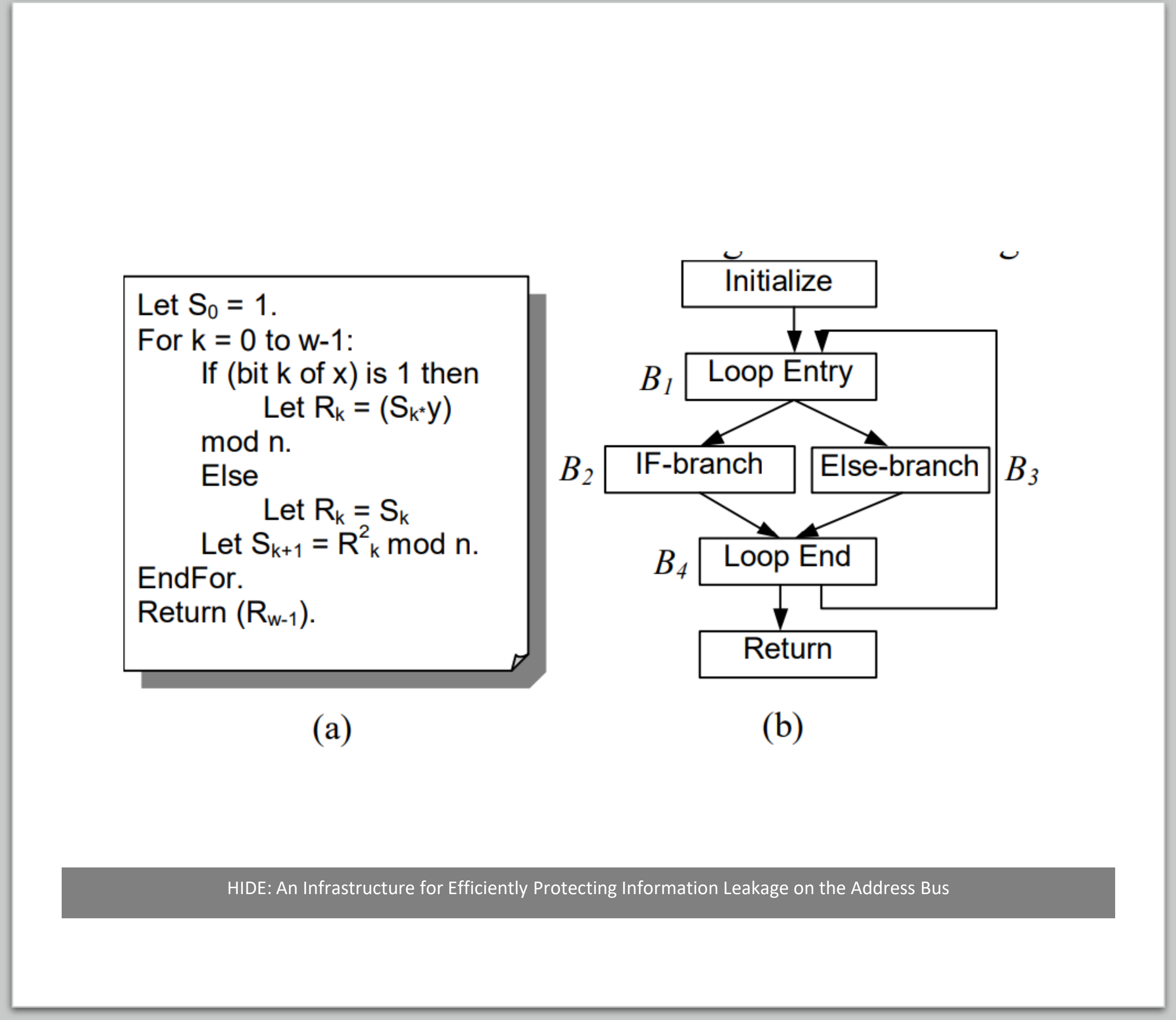


- Security vulnerabilities
  - Address translation vulnerabilities
    - Page unit vulnerabilities
      - Page-fault driven attacks [91，99]
      - Faultless page table-based attacks [37，53，95]
    - Segmentation unit vulnerabilities
      - Segmentation-based attacks [39]
  - CPU cache vulnerabilities
    - CPU cache-based attacks [5，17，31，69，82]
  - DRAM vulnerabilities
    - Cache + DRAM attacks [82]
    - Page table + Cache+ DRAM attacks [98]
  - Branch prediction vulnerabilities
    - Branch target buffer-based attacks [9，59]
    - Page table history-based attacks [21，44]
    - Return stack buffer-based attacks [56]
    - Out-of-order transient execution-based attacks [94]
  - Enclave software vulnerabilities
    - Return-oriented-programming attacks [58]
    - Enclave interface-based attacks[97]
  - Hardware vulnerabilities
    - RowHammer attacks [50]

https://dl.acm.org/doi/fullHtml/10.1145/3456631#Bib0068

SGX Vulnerability

- Memory access pattern is not hidden.
  - I can guess which algorithm is used if that's a widely used library.
    - RSA as an example.



```
Let $S_0$ = 1.
For k = 0 to w-1:
    If (bit k of x) is 1 then
        Let $R_k$ = ($S_k$*y) mod n.
    Else
        Let $R_k$ = $S_k$
    Let $S_{k+1}$ = $R^2_k$ mod n.
EndFor.
Return ($R_{w-1}$).
```

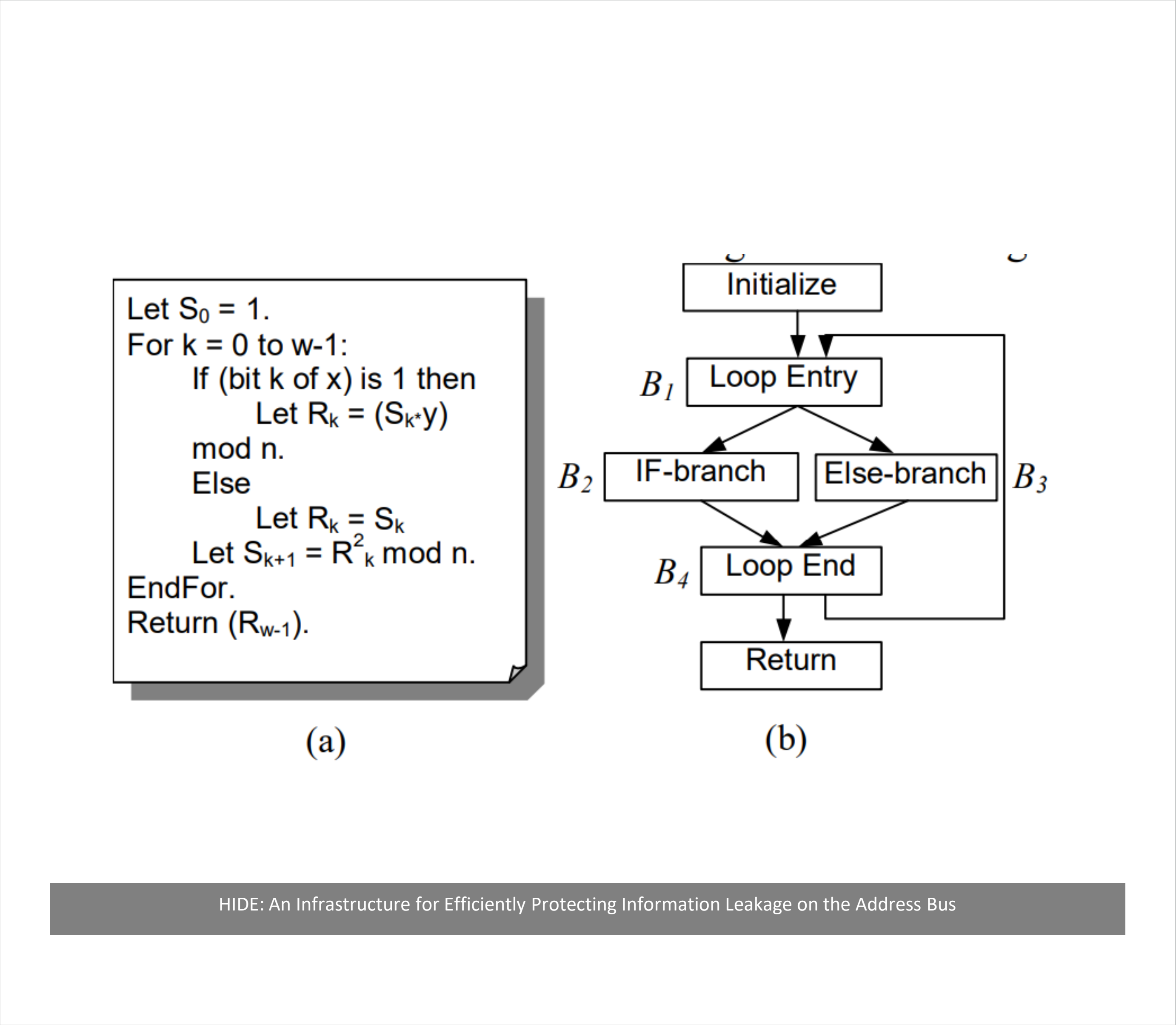
(a)

(b)

HIDE: An Infrastructure for Efficiently Protecting Information Leakage on the Address Bus

SGX Vulnerability

- Memory access pattern is not hidden.
  - I can guess which algorithm is used if that’s a widely used library.
  - I might be able to guess private key somehow.
    - Branching to the old location?
    - Branching to a new location?



```
Let $S_0$ = 1.
For k = 0 to w-1:
    If (bit k of x) is 1 then
        Let $R_k$ = ($S_k$*y)
    mod n.
    Else
        Let $R_k$ = $S_k$
    Let $S_{k+1}$ = $R^2_k$ mod n.
EndFor.
Return ($R_{w-1}$).
```

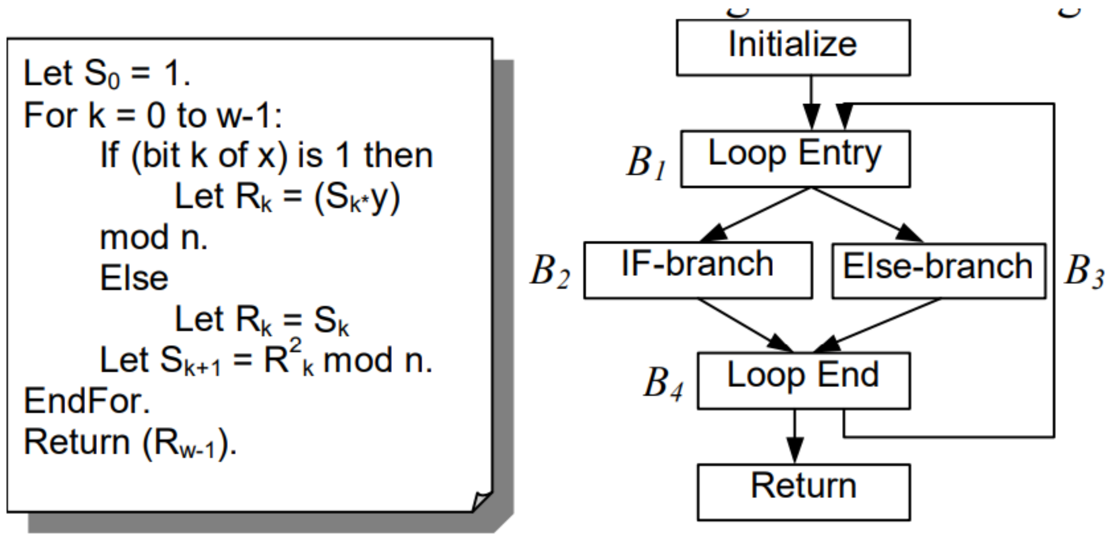
(a)

(b)

HIDE: An Infrastructure for Efficiently Protecting Information Leakage on the Address Bus

SGX Vulnerability

- Memory access pattern is not hidden.
    - I can guess which algorithm is used if that’s a widely used library.
    - I might be able to guess private key somehow.
        - Branching to the old location?
        - Branching to a new location?
- A big assumption is network connection is safe.
- It’s slow.



```
Let $S_0$ = 1.
For k = 0 to w-1:
        If (bit k of x) is 1 then
                Let $R_k$ = ($S_{k*}$y)
        mod n.
        Else
                Let $R_k$ = $S_k$
        Let $S_{k+1}$ = $R^2_k$ mod n.
EndFor.
Return ($R_{w-1}$).
```

(a) (b)

HIDE: An Infrastructure for Efficiently Protecting Information Leakage on the Address Bus

Differential Privacy

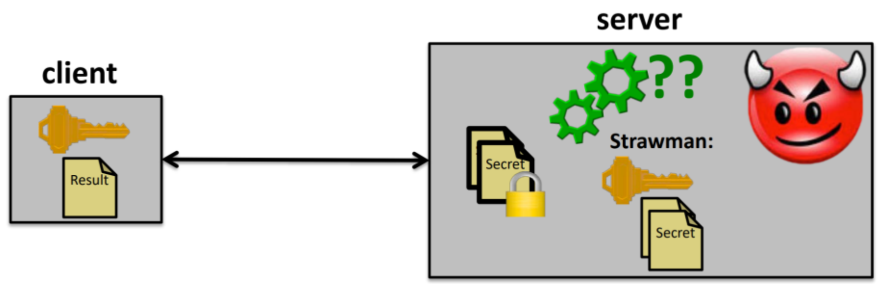
- We add noise and hope that the noise can cancel each other.
- Only make sense on aggregated results, e.g., sum, average, etc.

	A	B	C	D	E	F
Age	20	19	18	21	22	23
Age_Noise	22	17	20	19	24	21

For odd column, we +2, for even column, we -2.

Encrypted Database

- Key idea:
  - We don't trust the DB.
  - We only trust the device on hand.

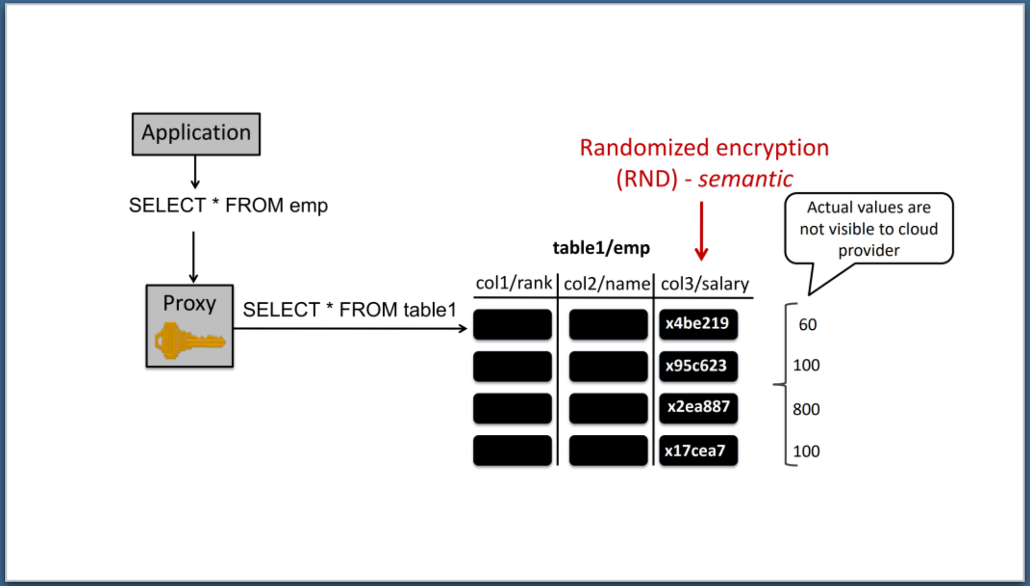


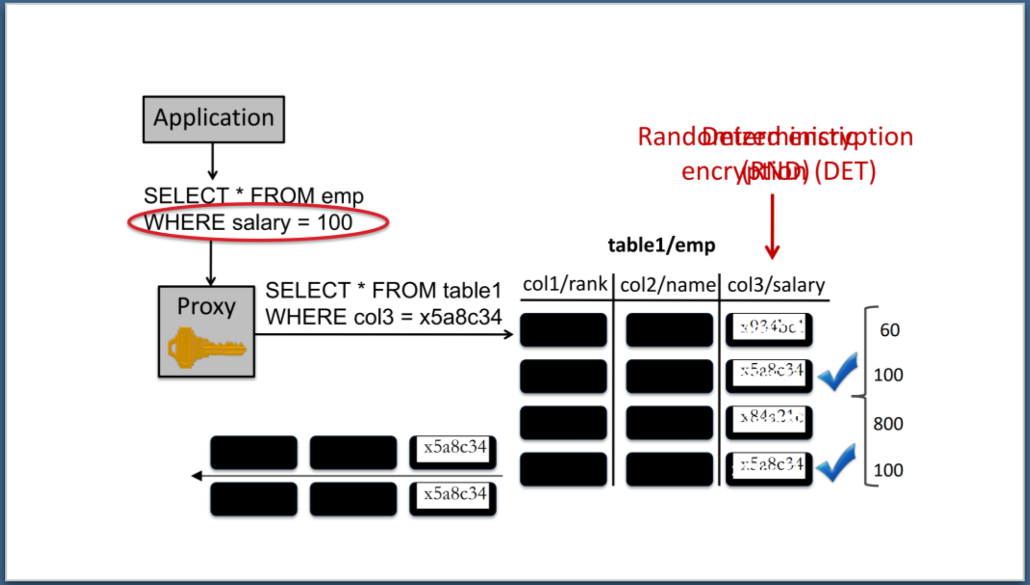
Encrypted Database

- What is in our tool box?
  - Trustable local environment: browser, application, etc.
  - Encryption algorithms:
    - DET: encryption that guarantees same input is mapped to the same output, potential leakage, used for =
    - RND: encryption with randomness, useful for data moving, e.g., select
    - HOM: basic calculation, e.g., HOM(a+b) = HOM(a) + HOM(b).
    - OPE: Comparable, >, <, max, min
    - JOIN, SEARCH, …
  - Commercial non-encrypted databases

Encrypted Database

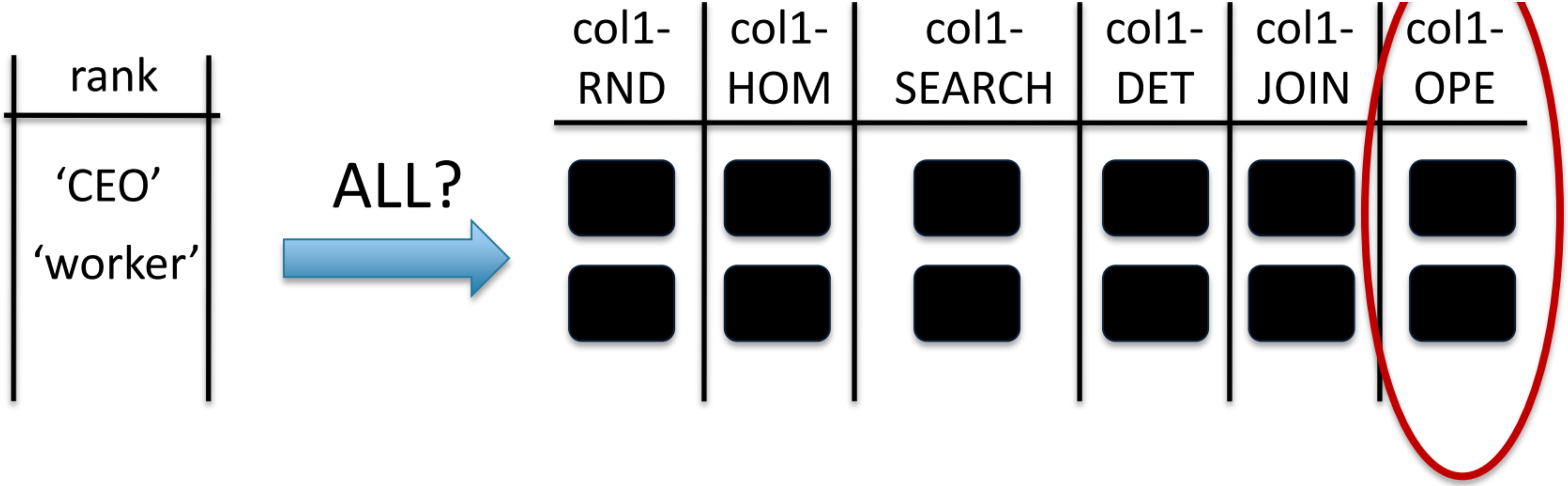
- Challenge
  - We don’t know what is in the query, so we don’t know which encryption algorithm to use.
  - Complex query operation might go beyond the capability of existing encryption algorithms.





Encrypted Database

Idea 1: Let's just expand the table and create a new column for each algorithm!



rank
'CEO'
'worker'

http://www.cs.cornell.edu/courses/cs5412/2022fa

Leaks order!

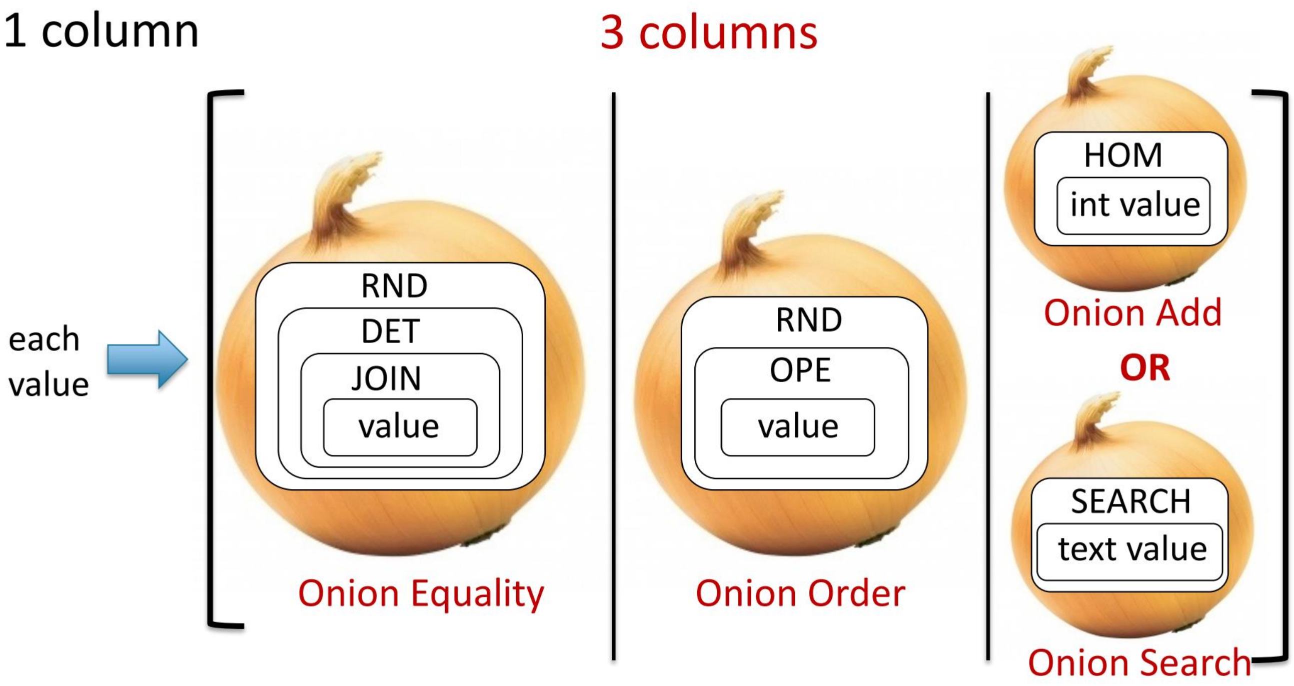
Encrypted Database

- Idea 1:
  - Information leakage is inevitable.
    - From OPE column, I can compare each person’s rank and figure our who is CEO, who is worker, what’s the percentage of management, etc.
    - Combined with DET column, I might be able to guess the salary of each class.

  - This consumes lots of space! If I have N algorithm, the new table is N times larger!

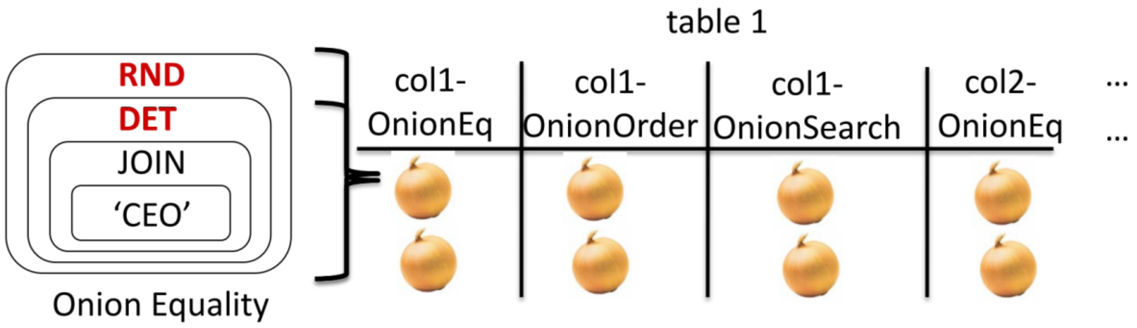
Encrypted Database

- Idea 2: Onion of algorithms.



Encrypted Database

- Idea 2:
  - Some encryption algorithms are "stackable".
  - E.g., first DET then RND can support select at the "first layer" and = if we "peel off" the RND layer.
  - We never peel off the most inner layer!



SELECT * FROM emp WHERE rank = 'CEO'

⇩

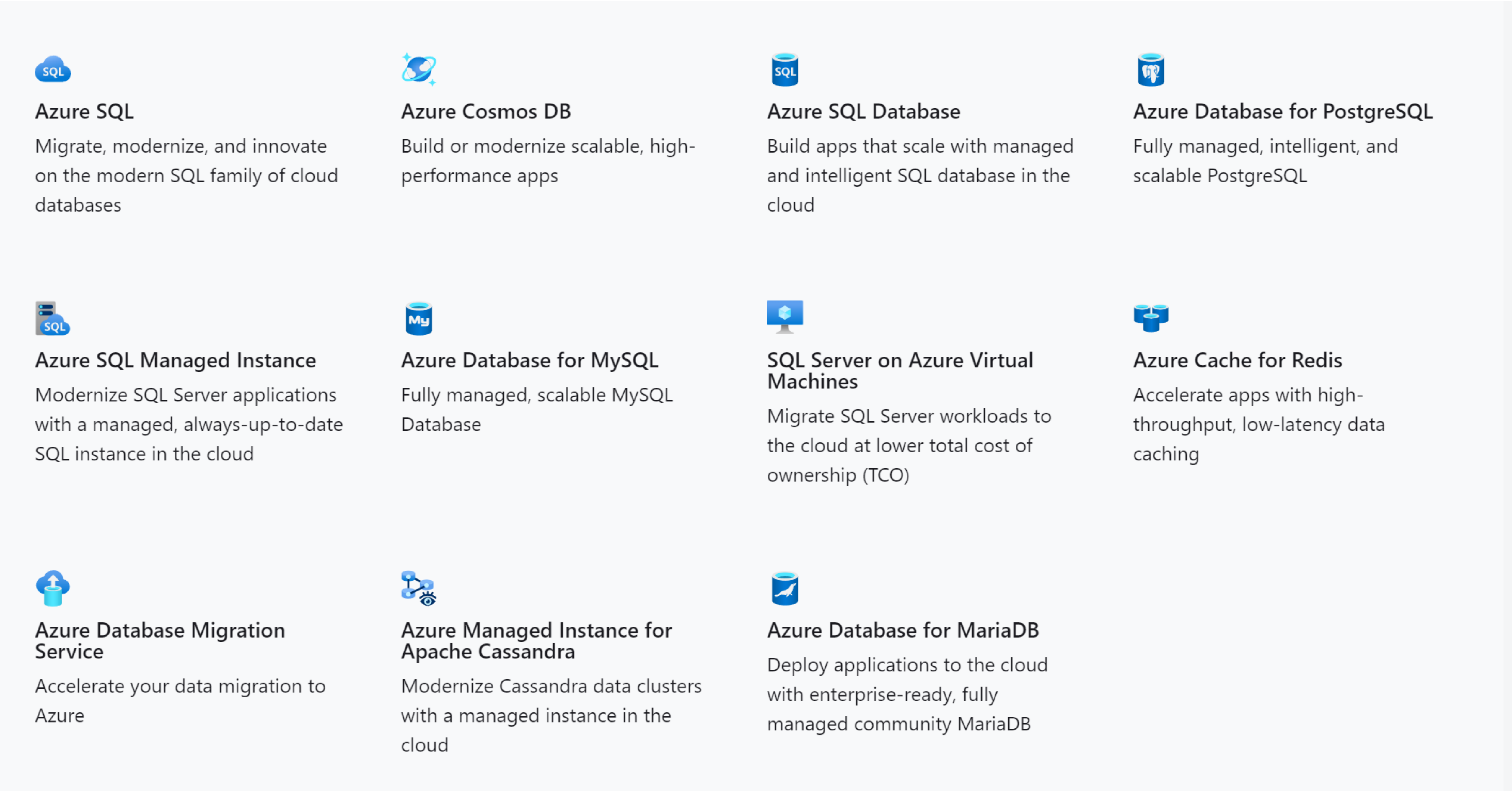
UPDATE table1 SET col1-OnionEq =
Decrypt_RND(key, col1-OnionEq)

SELECT * FROM table1 WHERE col1-OnionEq = xda5c0407

Encrypted Database

- Idea 2:
  - Performs well, with at most 26% slower
  - Deployed in large systems.
- Still not a panacea
  - Some queries are too complicated: computation + sorting.
  - Information leakage is inevitable.

SQL



Azure SQL

Migrate, modernize, and innovate on the modern SQL family of cloud databases

Azure Cosmos DB

Build or modernize scalable, high-performance apps

Azure SQL Database

Build apps that scale with managed and intelligent SQL database in the cloud

Azure Database for PostgreSQL

Fully managed, intelligent, and scalable PostgreSQL

Azure SQL Managed Instance

Modernize SQL Server applications with a managed, always-up-to-date SQL instance in the cloud

Azure Database for MySQL

Fully managed, scalable MySQL Database

SQL Server on Azure Virtual Machines

Migrate SQL Server workloads to the cloud at lower total cost of ownership (TCO)

Azure Cache for Redis

Accelerate apps with high-throughput, low-latency data caching

Azure Database Migration Service

Accelerate your data migration to Azure

Azure Managed Instance for Apache Cassandra

Modernize Cassandra data clusters with a managed instance in the cloud

Azure Database for MariaDB

Deploy applications to the cloud with enterprise-ready, fully managed community MariaDB

ACID

- Atomicity, consistency, Isolation, Durability.
- My own story: A small project containing only 3 KVTs gave me a huge punishment in performance.
    - Students, parents, students’ classes.
    - Some complex operations require me to read all tables, lock all tables, update accordingly and then free all the locks.
    - This process is surprisingly slow with features like hot data push, i.e., I can only access the part of table in my browser, so hitting a cold cache is extremely harmful.
- My lesson:
    - Schema is important.
    - It does not harm to use relational databases.

SQL Tips

- Join order matters.
- Plan ahead in your schema design.
- It never hurts to have multiple DBs.