Encrypted Database Made Secure yet Maintainable

Mingyu Li, Xuyang Zhao, Le Chen, Cheng Tan, Huorong Li

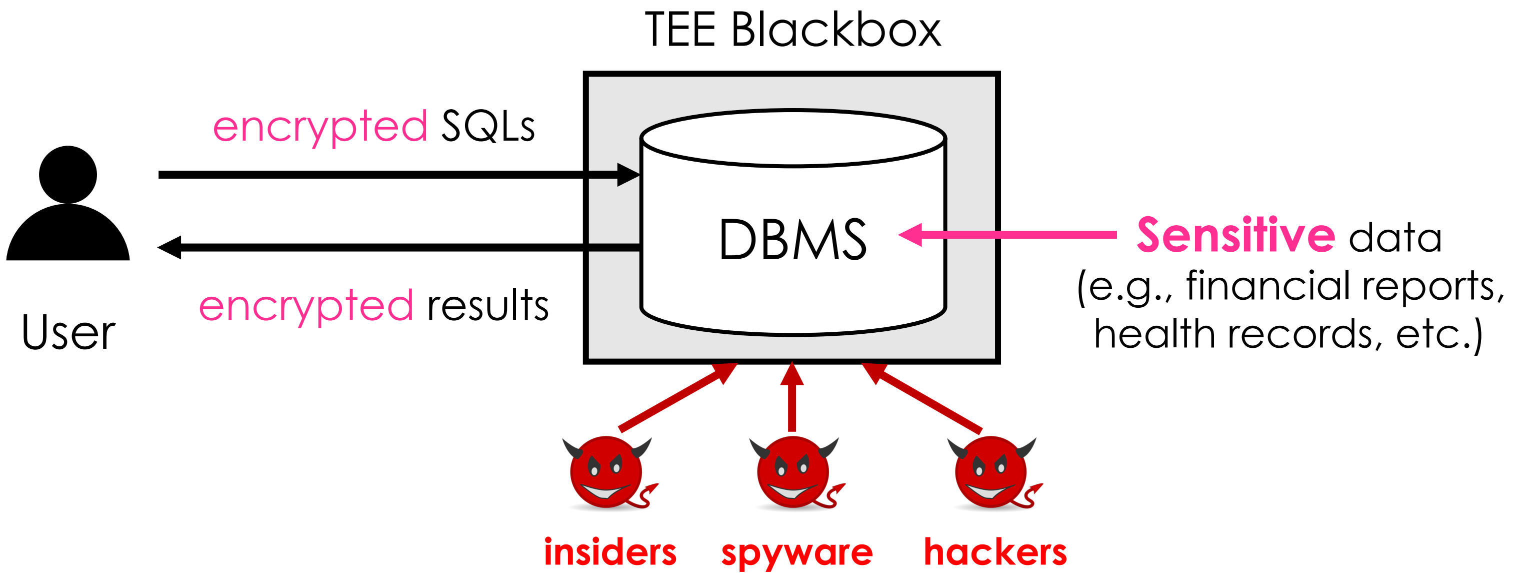
Sheng Wang, Zeyu Mi, Yubin Xia, Feifei Li, Haibo Chen

上 海 人 工 智 能 实 验 室
Shanghai Artificial Intelligence Laboratory

Alibaba Group

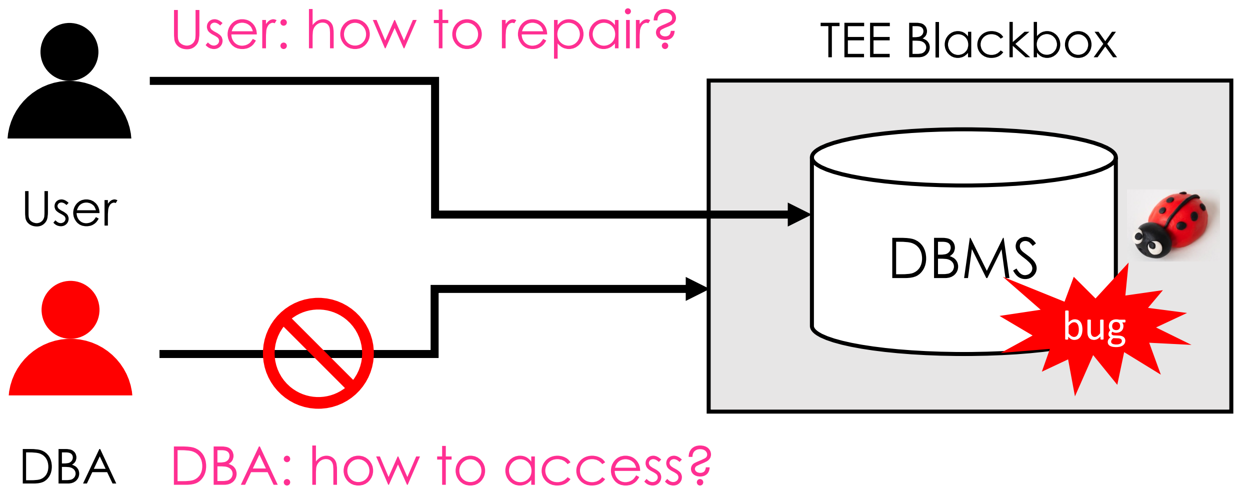
Type-I EDB against data theft

- Trusted hardware (TEE) offers a blackbox
- Putting DBMS inside TEE → encrypted database (EDB)



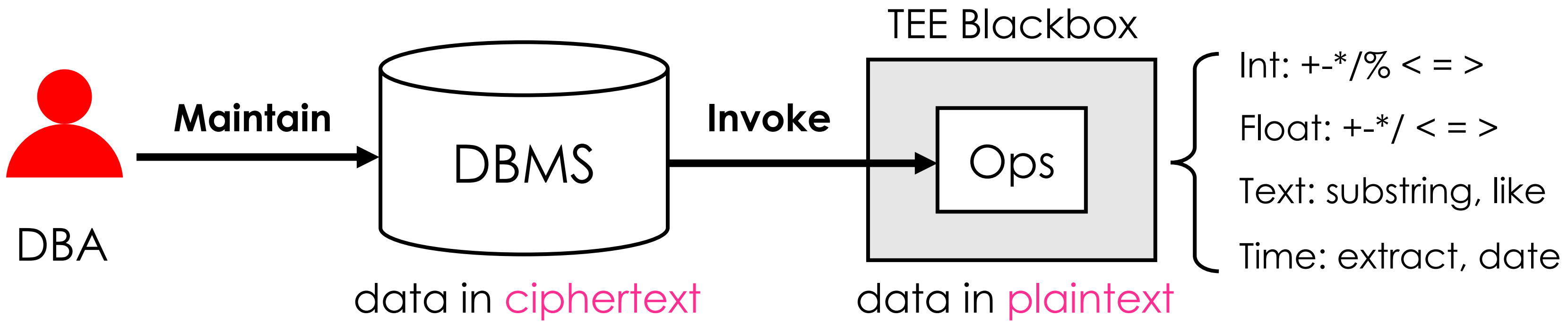
EDB bugs need diagnosis

- DBMS bugs occur! *e.g., misconfiguration* → *slow query*
- Typically handled by experts, i.e., database admin (DBAs)



Type-II EDB: DBMS + Ops

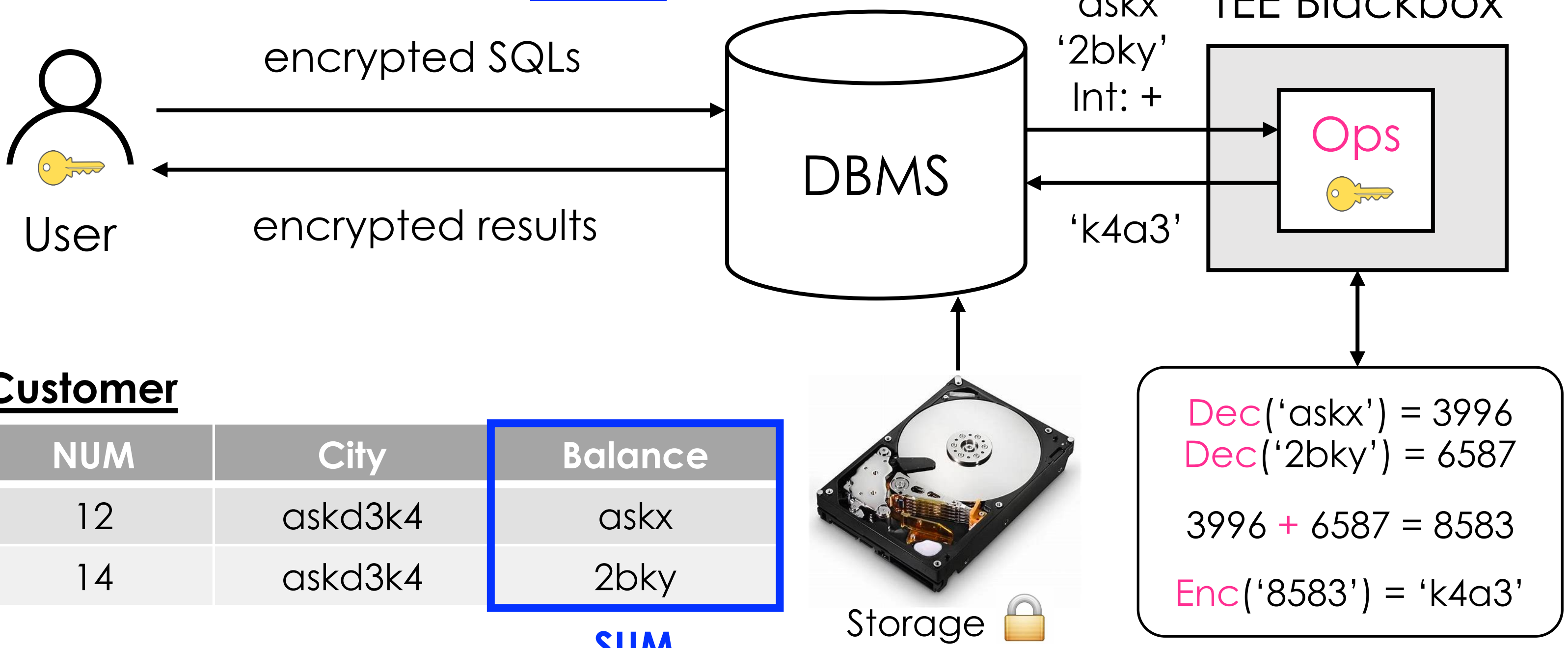
- Compose primitive operators (via database extensions)
  see next slide
  - 1. Data kept encrypted → privacy
  - 2. DBMS states visible → maintainability



- Adopted by cloud DBaaS vendors, such as Azure, Alibaba

Type-II workflow using ops

SELECT SUM(Balance) FROM Customer WHERE City='askd3k4';



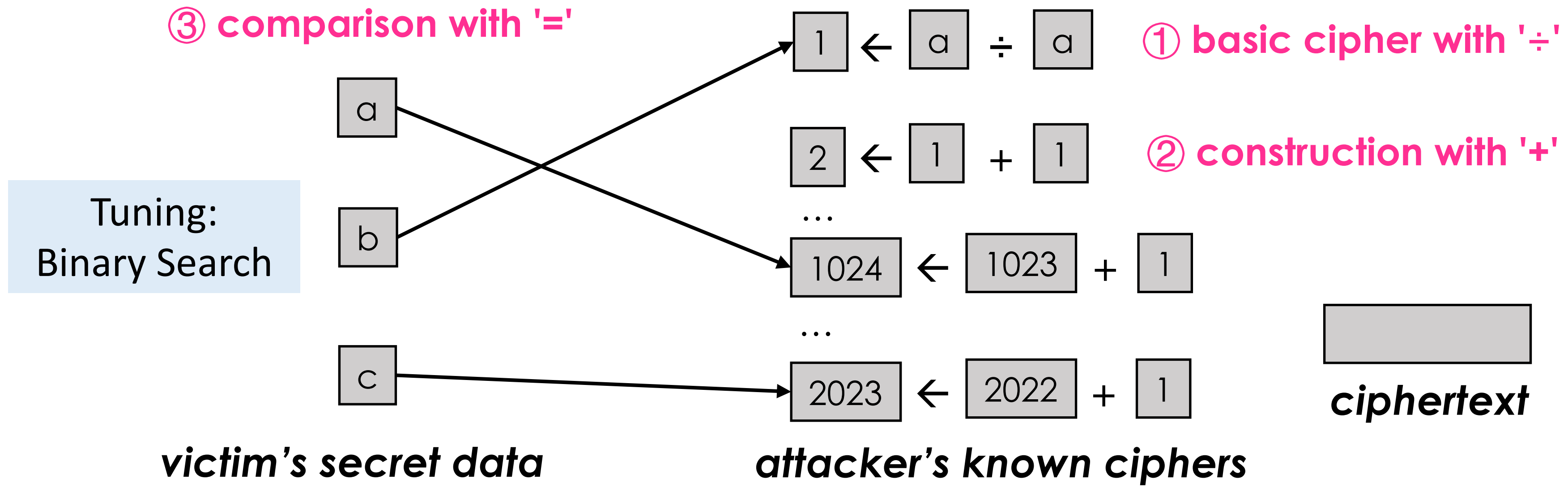
Customer

NUM	City	Balance
12	askd3k4	askx
14	askd3k4	2bky

SUM

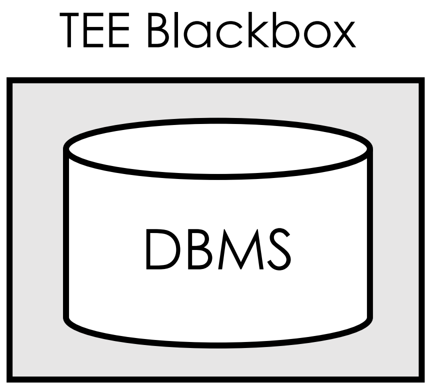
Our discovery: Smuggle Attacks!

- Operators can be manipulated by insiders like DBAs
- Smuggle vs. leakage attacks[1]: zero prior knowledge



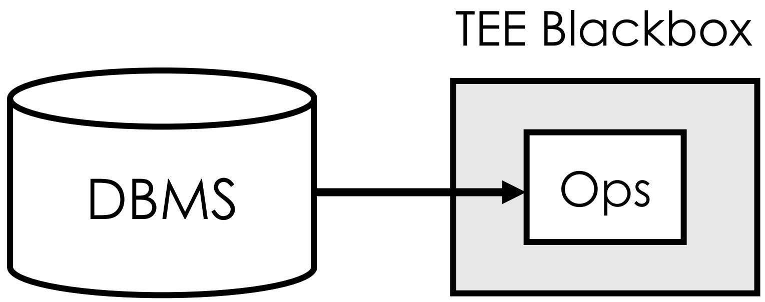
[1] SoK: Cryptographically Protected Database Search

Dilemma: security vs. maintenance



✅ Security

❌ Maintenance



✅ Maintenance

❌ Security (e.g., Smuggle)

How to resolve this dilemma?

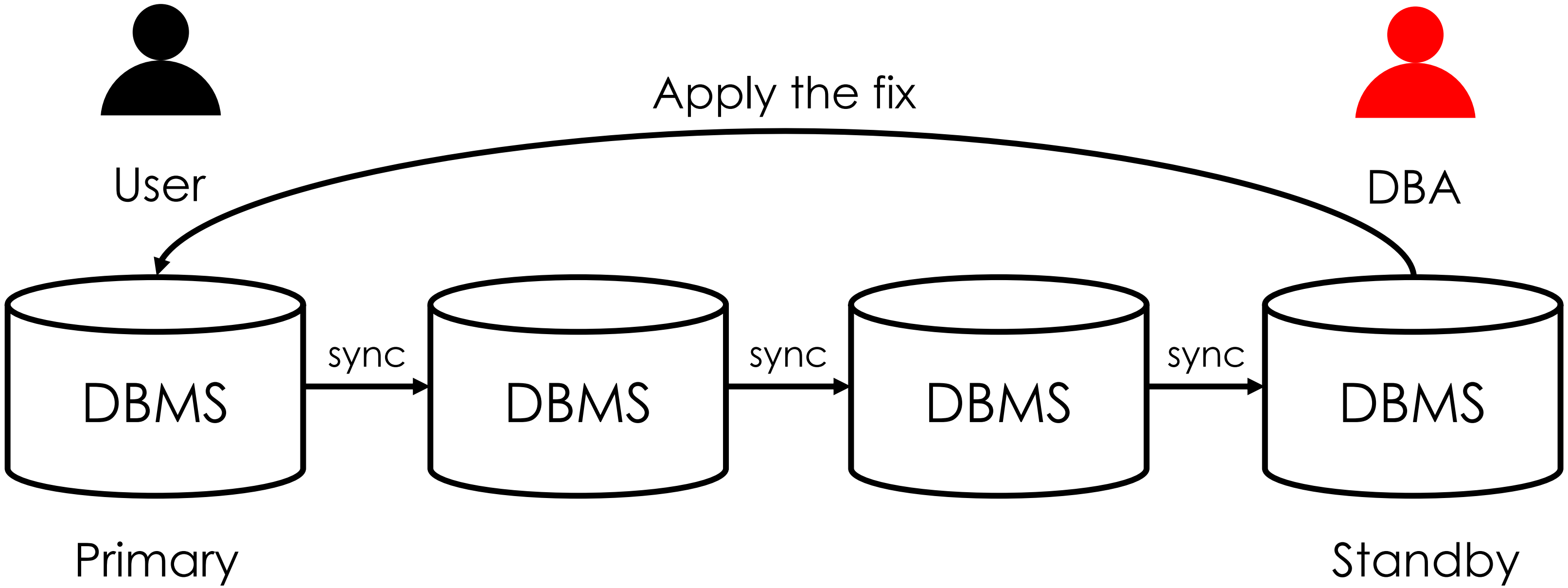
Type-I EDB

Type-II EDB

Main challenges

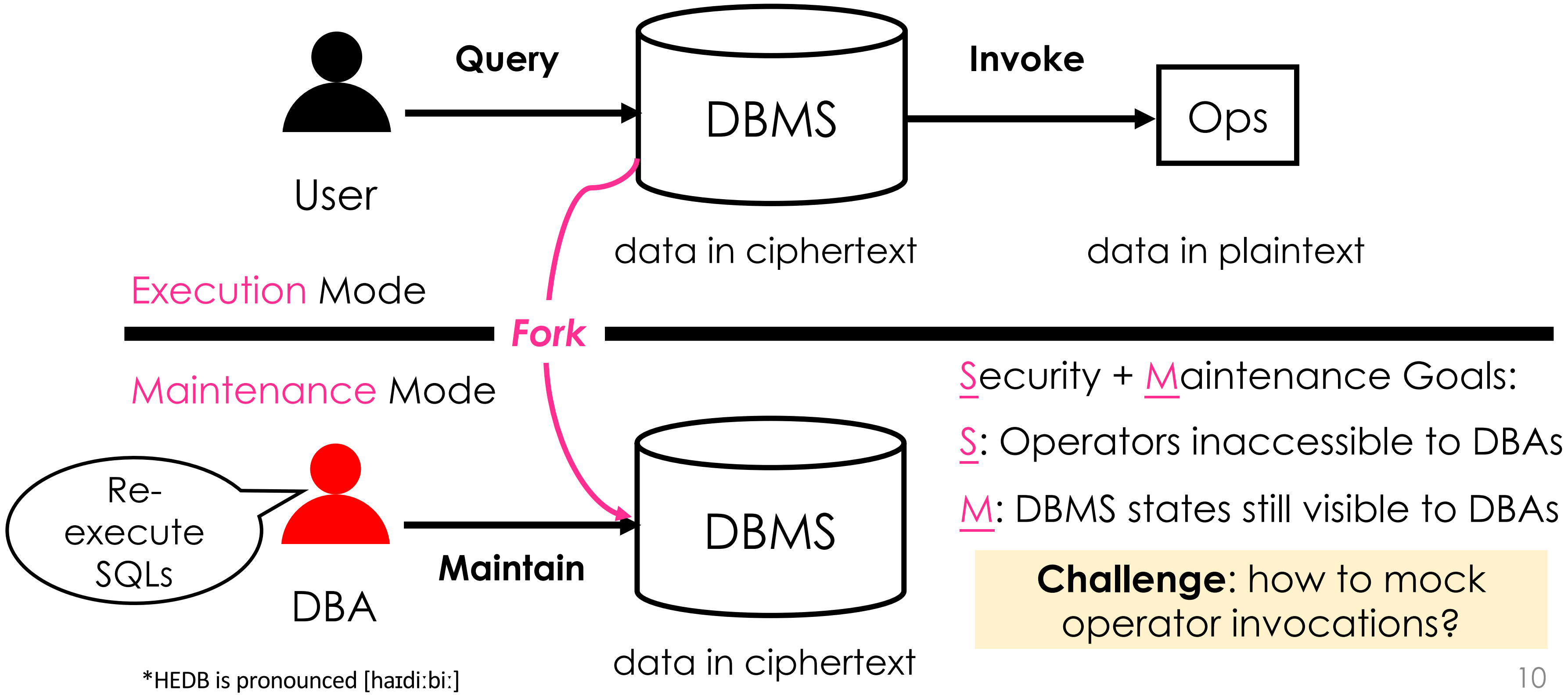
- Can we build an EDB with security and maintainability?
- Challenge #1: Defense Smuggle
- Challenge #2: Maintain DBMS
- Challenge #3: Maintain Ops

Observation: DBA works on standby



Insight: isolate DBAs from the EDB instances used by Users

HEDB idea: a dual-mode architecture



Security + Maintenance Goals:

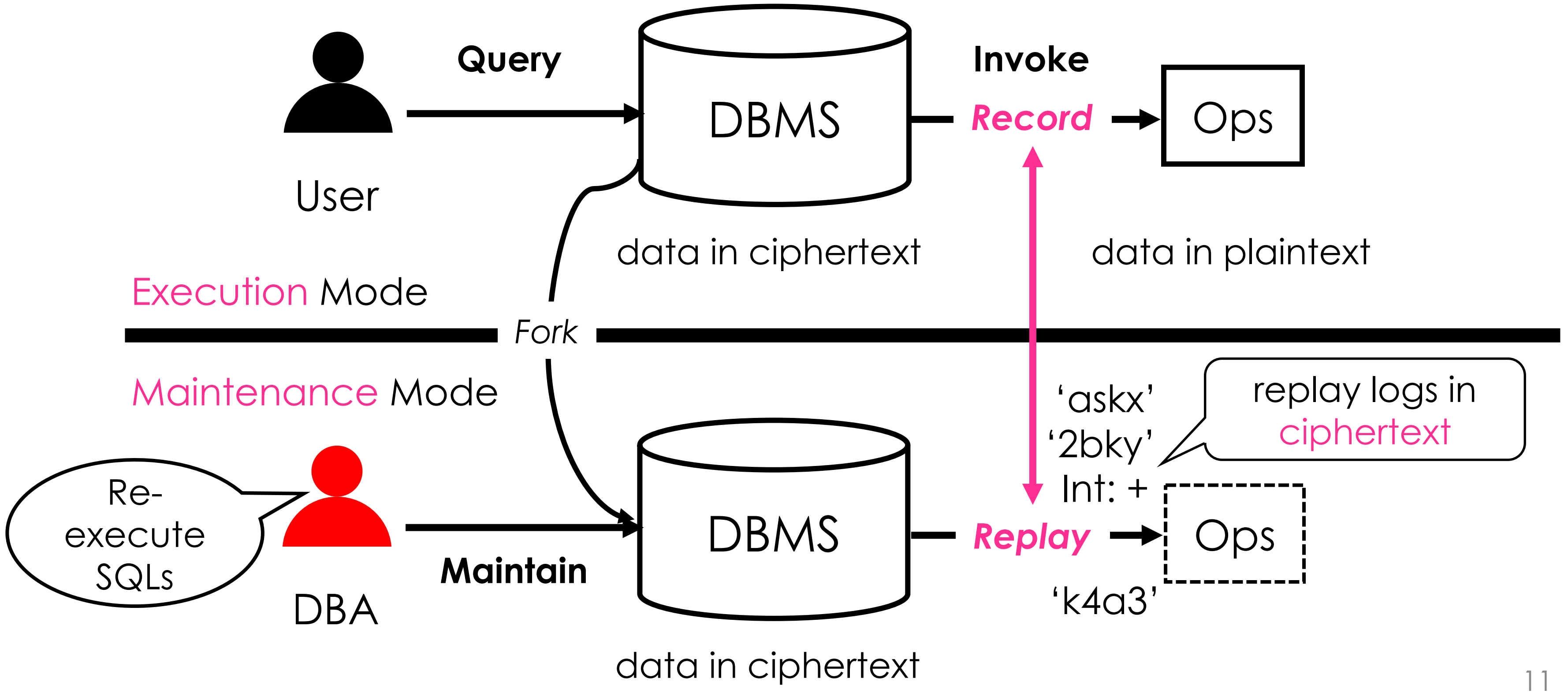
S: Operators inaccessible to DBAs

M: DBMS states still visible to DBAs

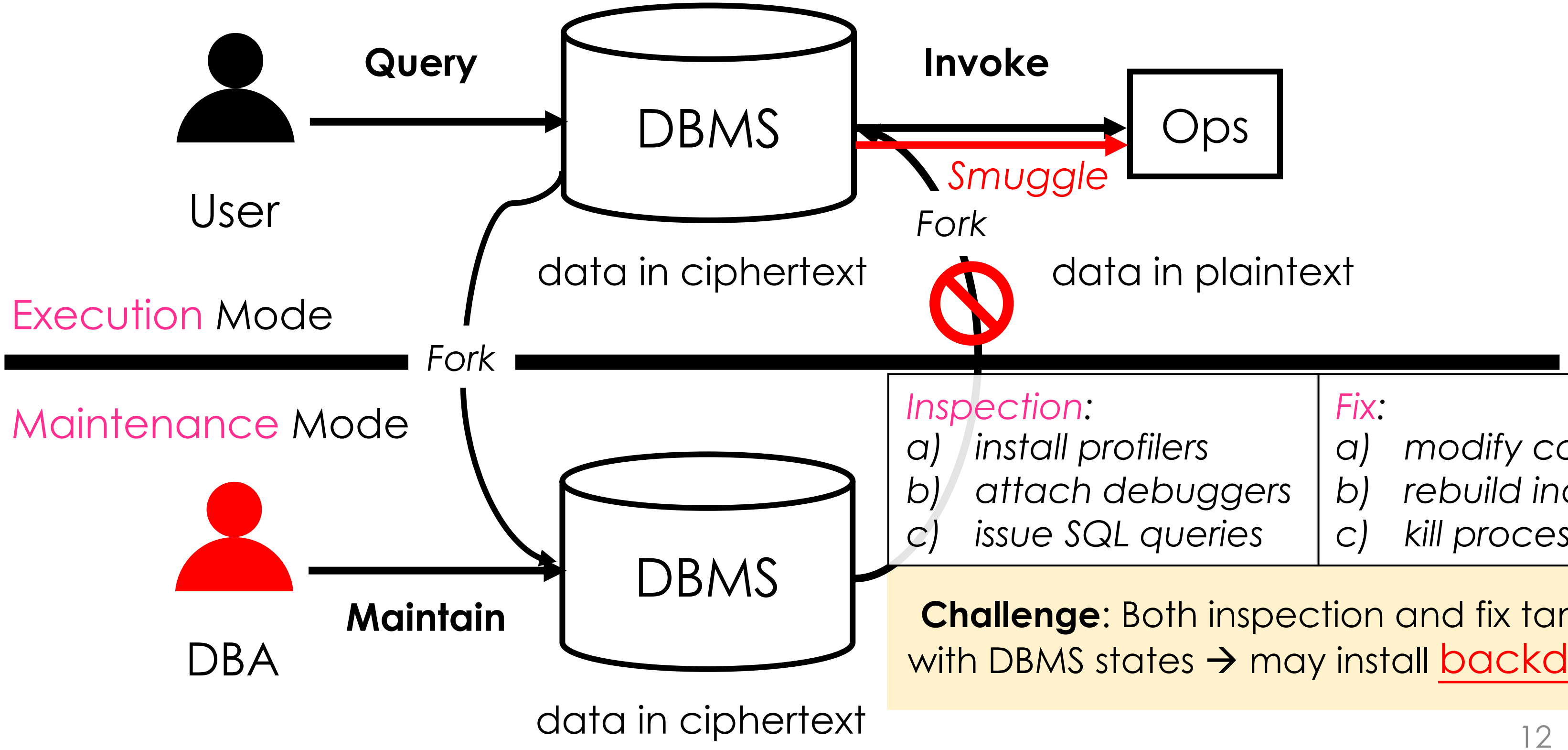
Challenge: how to mock operator invocations?

*HEDB is pronounced [haɪdi:bi:]

HEDB idea: a dual-mode architecture



#2: How to securely fix DBMS bugs?



Inspection:	*Fix*:
a) install profilers	*a) modify conf.*
b) attach debuggers	*b) rebuild index*
c) issue SQL queries	*c) kill processes*

Challenge: Both inspection and fix tamper with DBMS states → may install backdoors!

Observation: inspection-fix asymmetry

Summarized from 50 experienced DBAs and 8 popular databases *

Deadlock? *Too many connections?*

Wrong privilege? *No space left?*

Index issues? *Insufficient buffer?*

Network unreachable?

Fix
shared_buffer = [num]
cancel_backend(pid)
REINDEX TABLE [name]
VACUUM FULL [name]

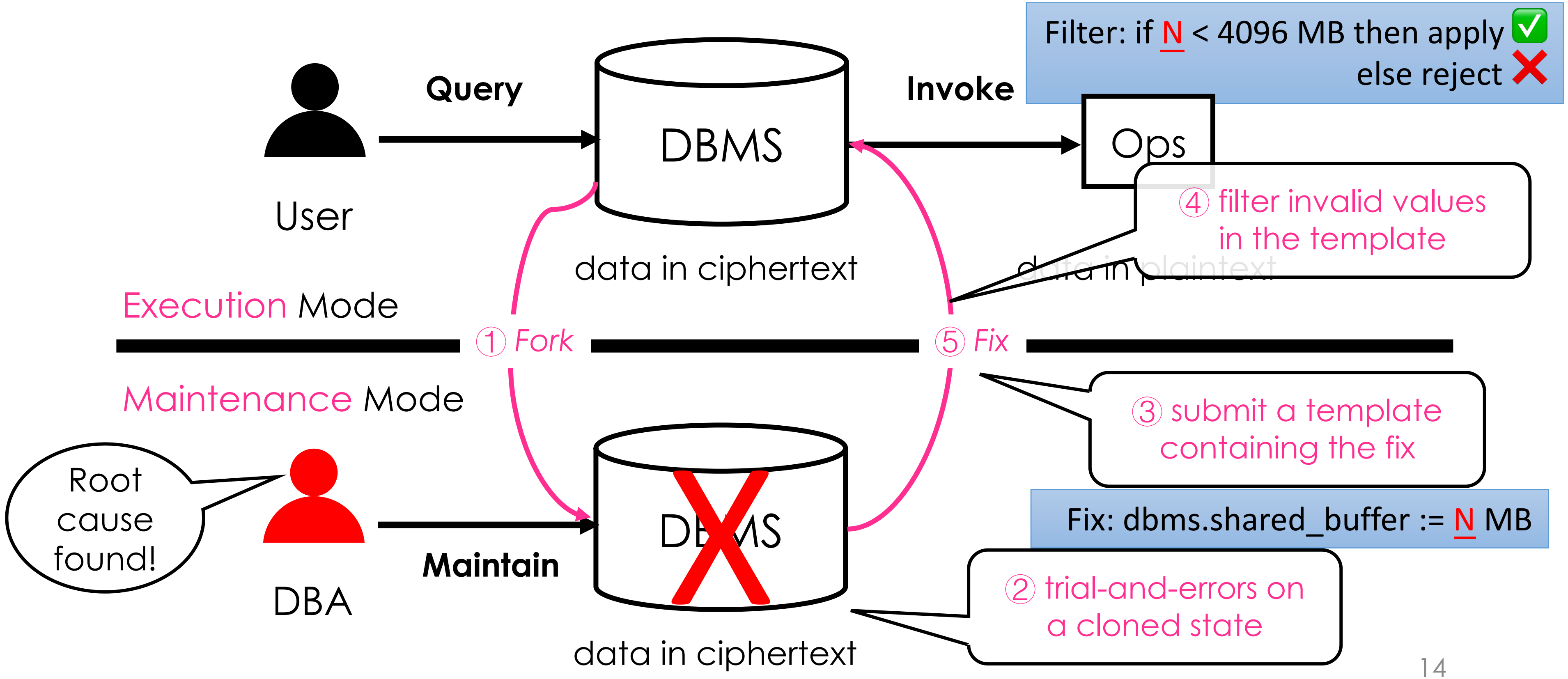
Inspection: arbitrary and complex

Fix: regular and structured

Insight: open for inspection, restrictive for fix

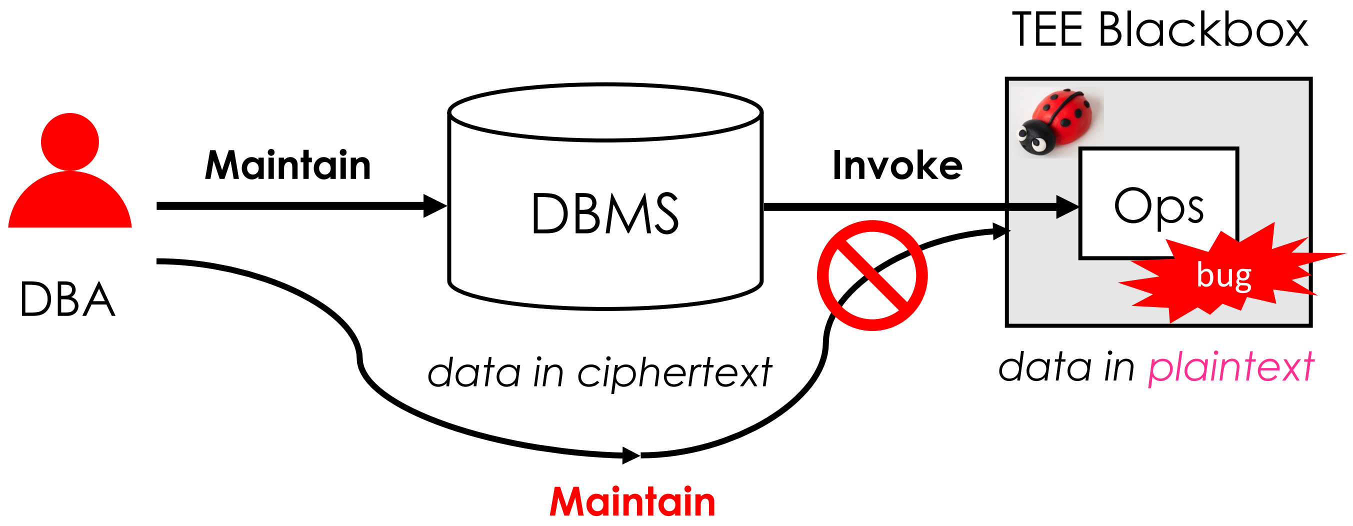
* PostgreSQL, MySQL, MS SQL Server, and 5 cloud databases like Alibaba PolarDB, Amazon Aurora, Azure SQL, etc.

HEDB idea: fork-discard-template-fix


Filter: if N < 4096 MB then apply ✅
else reject ❌
User
Query
DBMS
Invoke
Ops
data in ciphertext
data in plaintext
④ filter invalid values in the template
Execution Mode
① Fork
⑤ Fix
Maintenance Mode
③ submit a template containing the fix
Root cause found!
DBA
Maintain
DBMS
Fix: dbms.shared_buffer := N MB
② trial-and-errors on a cloned state
data in ciphertext

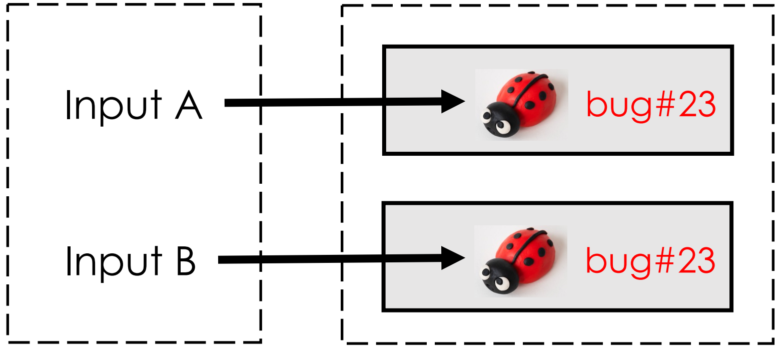
#3: What if ops have trouble?

- Ops can also be buggy! *e.g., correctness bugs*
- **Challenge**: ops contain plaintext user secrets



Observation: control flow matters

- Ops are stateless (for crash consistency)
- Ops' control flow only depends on its inputs



control-flow equivalence same bug

Insight: hide secrets using control-flow equivalence

Real input $\longleftrightarrow$ Fake input

control-flow equivalence pair

Multiple inputs can lead to the same buggy control flow.

How to generate fake inputs?

$$\begin{tikzcd}
\text{real input} \arrow[r] & \boxed{\text{concolic executor}} \arrow[r] & \text{control flow} \arrow[r] & \boxed{\text{constraint solver}} \arrow[r] & \text{fake input} \\
 & & & \text{modern data masking engine} \arrow[u] &
\end{tikzcd}$$

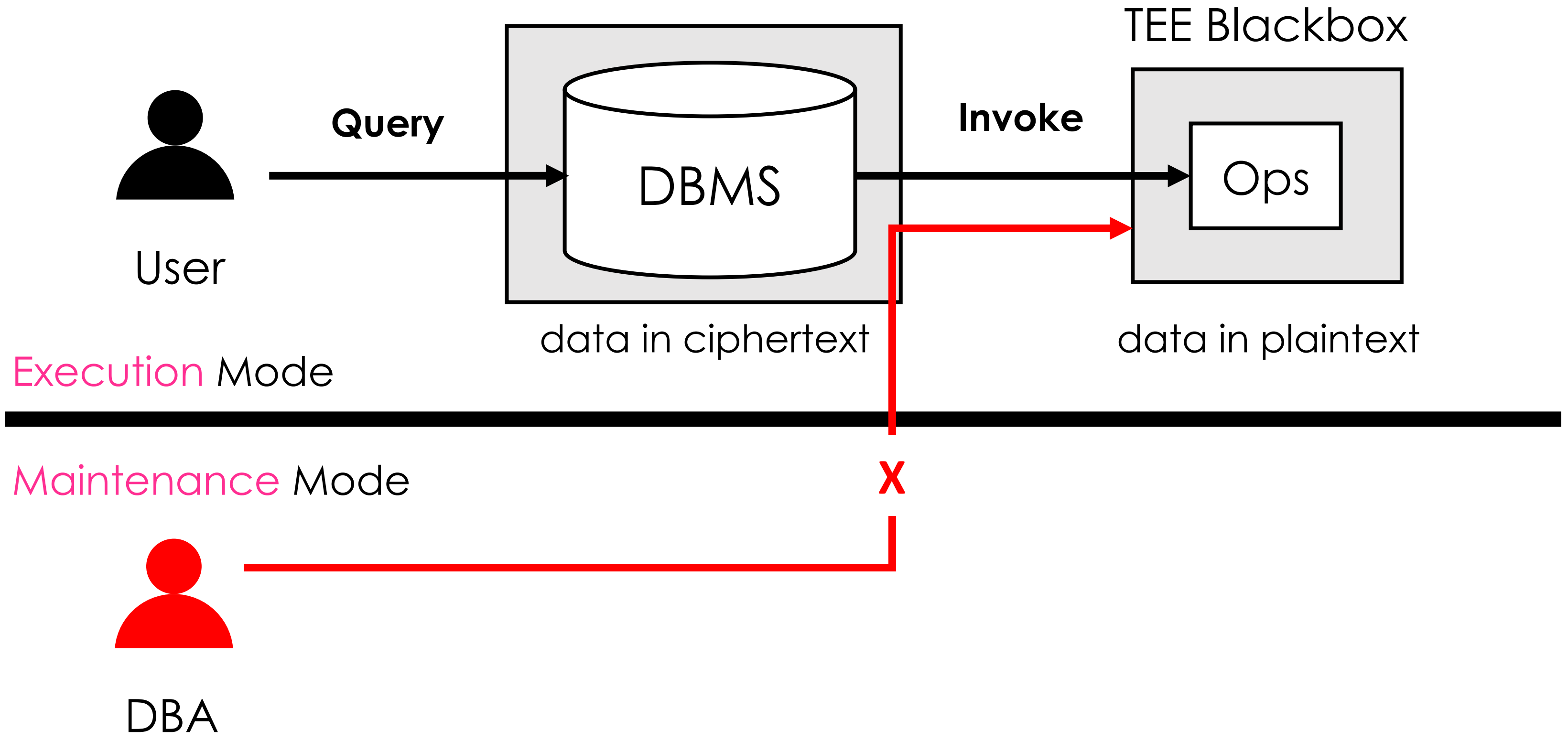
Challenge: what kind of fake inputs are **secure** enough to preserve privacy?

- only one fake input needs to be generated
- satisfying reproducibility, privacy and efficiency

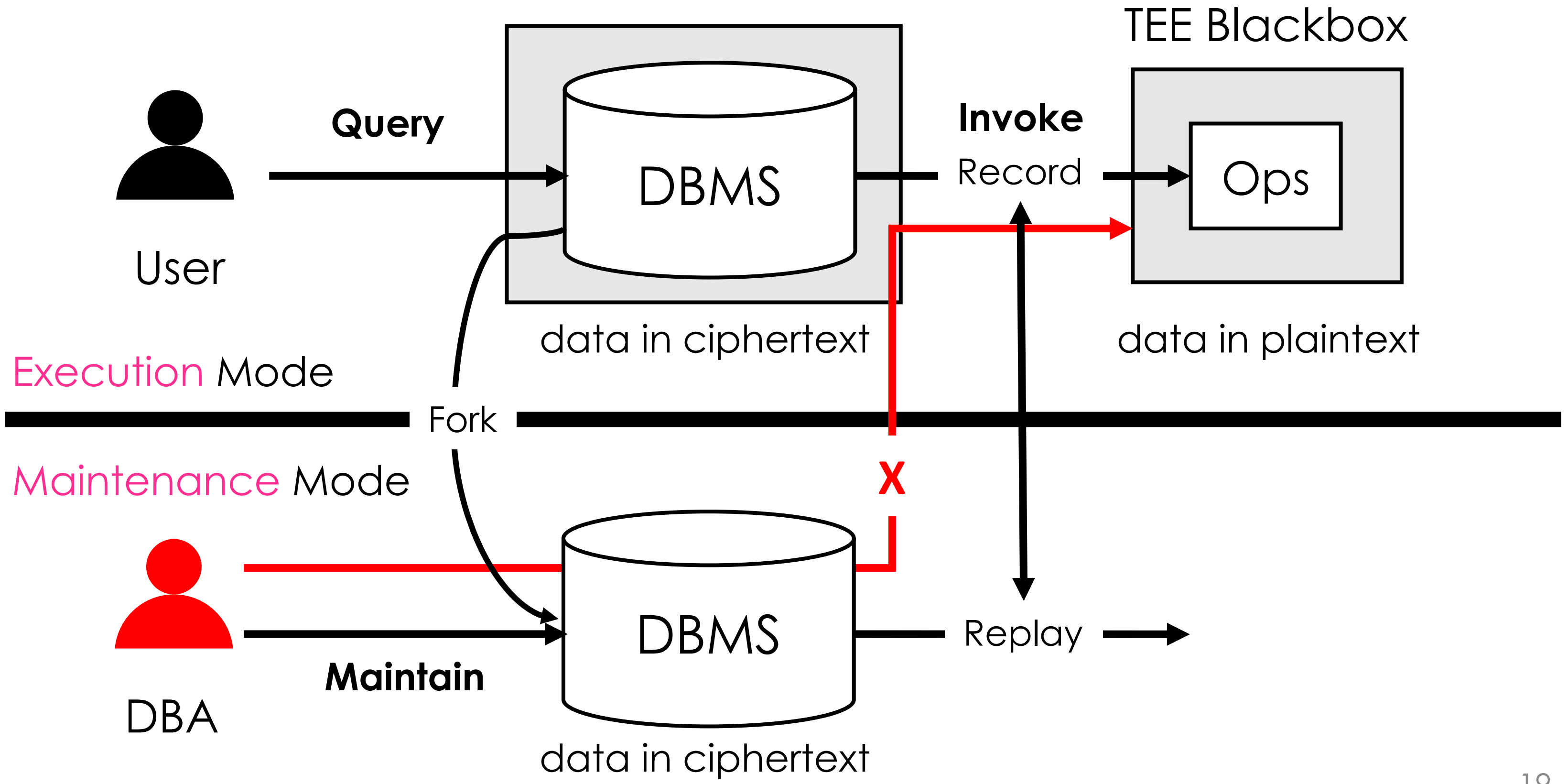
'Phone' **data masking rule:**
***4567890

Real input & SQL statement	Fake input #1	Fake input #2
Phone: **5105288649**::VARCHAR(10) SELECT … WHERE Phone LIKE '**5105%**';	**5105101010** (leaks area code)	**0005101010**, LIKE '**0005%;**

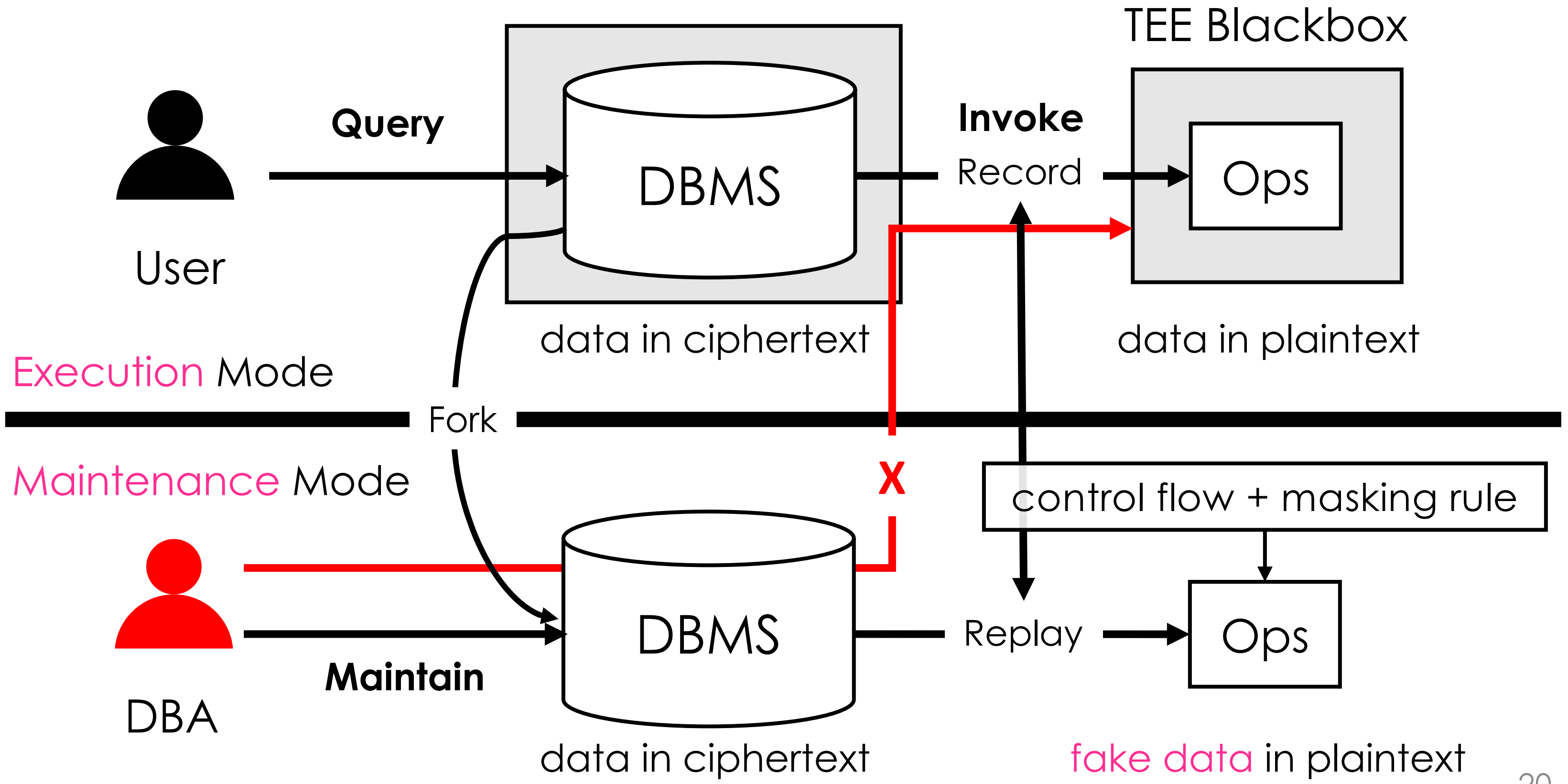
How HEDB prevents Smuggle



How HEDB maintains DBMS



How HEDB maintains Ops



Building HEDB on real-world hardware

- Implementation:
  - Two modes: ARMv8 S-EL2, 100 LoC
  - DBMS: PostgreSQL 13
  - Operators: user-defined functions, 4K LoC
  - Record-replay: KLEE + Z3, 1.8K LoC
- Trusted Hardware:
  - Confidential VM: AMD SEV, Intel TDX, ARMv9 CCA, IBM PEF

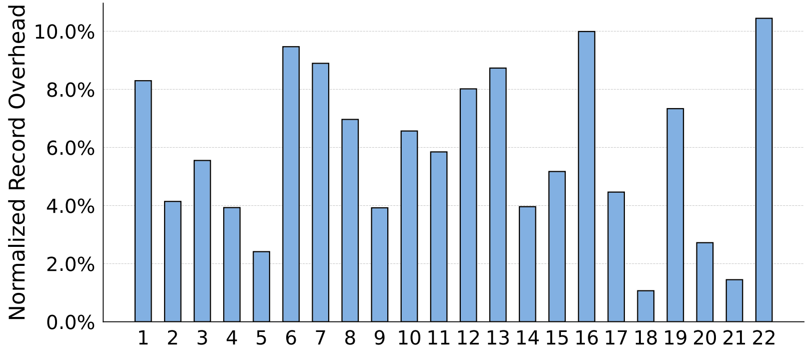
Evaluating maintainability

- Empirical study:
  - 6-month DB issues
  - 28K tickets from users
- Maintenance coverage (3/4)
  - common daily DBA tasks
  - DBA query rewriting may yield unseen ops invocations

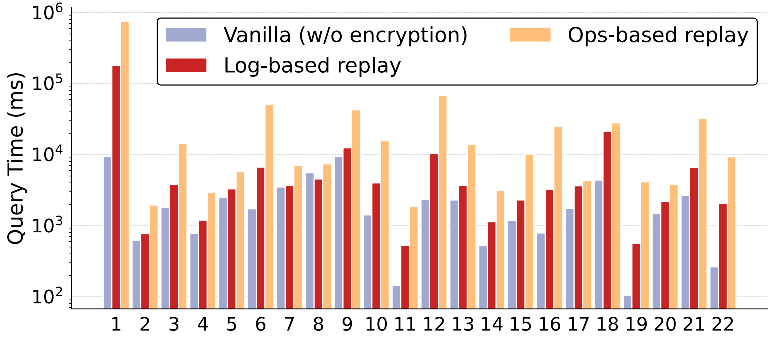
Control-plane Management	HEDB
start, stop, switchover	✅
backup, migration	✅
Data-plane Troubleshooting	**HEDB**
analyze DBMS plans	✅
cancel hung queries	✅
Data-plane Tuning	**HEDB**
update configuration	✅
rewrite queries	❌
Data-plane Bug Reporting	**HEDB**
coredump DBMS	✅
reproduce ops bugs	✅

Evaluating cost

- Workload: TPC-H (financial representative)
  - TPC-H requires ‘+’, ‘/’, ‘=’ that can conduct smuggle attacks
- Runtime cost:



record: < 6%



replay: 3.9x faster than execution

- Storage cost: < 10%

Main contributions

- **#1 survey**: state-of-the-art encrypted databases (EDBs)
  - Type-I and Type-II
- **#2 attack**: an effective and efficient attack to real-world EDBs
  - Smuggle Attacks
- **#3 study**: empirical studies of modern DBA operations
  - Common DB issues + corresponding DBA actions
- **#4 system**: enabling DBA tasks without compromising user secrets
  - HEDB

Summary

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

- *HEDB*: a dual-mode EDB that enables common DBA-based maintenance tasks without compromising user secrets.
- Future work:
  - support flexible DBA tasks (e.g., query rewriting)
  - cover concurrent write workloads (e.g., TPC-C)
- Learn HEDB tutorial at: https://github.com/SJTU-IPADS/HEDB
- For questions, feel free to contact Mingyu Li (maxul@sjtu.edu.cn)