Tutorial Outline

PART I. Personal Data Management Systems (PDMS)

- Review of functionalities & addressed privacy threats
- Individual's PDMS vs (corporate) DBMS and main properties to achieve

PART II. TEE-based Data Management

- The promises of Trusted Execution Environments (TEEs)
- A review of privacy-preserving data management using TEEs

PART III. Bridging the Gap between PDMS and TEEs

- How could the main properties be achieved?
- A quick view of remaining challenges

Positioning vs traditional DB security techniques?

How to achieve trust and privacy? Lots of existing works.

- Data and queries confidentiality & integrity: Encrypt a central DB and Hash/Merkle it?
- \+ hide access patterns: ORAM or Keep DB locally and SMCize the query evaluation?
- \+ make it scalable (perf/volume): Adopt distributed/gossip style query evaluation?
- \+ make it generic (SQL, inv. search, ML, …): Avoid DP? Use a central Trusted Third Party?

Difficult combination: be confidential & fair & generic & scalable

- Local Differential privacy (e.g., RAPPOR) ➔ generic comp ? Integrity ?
- Gossip-style (e.g., Chiaroscuro/Davide) ➔ generic comp ? Integrity ?
- Homomorphic encryption (e.g., SMCQL) ➔ generic comp ?
- Somewhat homomorphic encryption (e.g., Gentry-SHE) ➔ confidentiality ? [BGC+18]
  generic compution ?

Would Trusted Execution Environments help ?

Secure Element (SE) → Trusted Execut° Environm^t (TEEs)

From secure elements, TPM, HSM, etc.

- Smart cards or TPM (in smartphones, PCs, home boxes)

… to: Trusted execution environments (TEEs)

- Specialized HW: ARM Trustzone, Intel SGX, AMD platform security, etc.
- Everywhere : Smartphones & PCs

Promise: HW level isolation and attestation

- Isolation:
  - Code executed within a TEE safe from external observation/tampering (OS, user)
- Attestation:
  - Ability to give a certificate that result produced by a specific piece of code running within TEE

Secure Element (SE) → Trusted Execut° Environmᵗ (TEEs)

Relevance in a personal cloud context

- Protect users against their own environment → non expert users are safe?
- Mutual trust without resorting to costly cryptographic mechanisms → mutual trust?

Limits of TEE security: cat and mouse race

- Side channels ➔ threat model of recent TEEs
  - Execution time (by OS/colocated programs)
  - …. memory accesses at page level (OS), byte level (memory bus)
  - → **Won't be fixed : need to be addressed in solutions**
- Attacks based on speculative execution ➔ leak secrets (secret keys of enclaves)
  - Eg. Spectre, Foreshadow.
  - → **Out of scope: need to be fixed by HW manufacturer**

Not a magic bullet that allows to execute everything safely

A Bit of History & outline of part II

Secure HW support evolution

A. Secure (<u>single</u>) database management in TEEs

Basic TEEs for dedicated personal data-oriented apps (since early 2000)

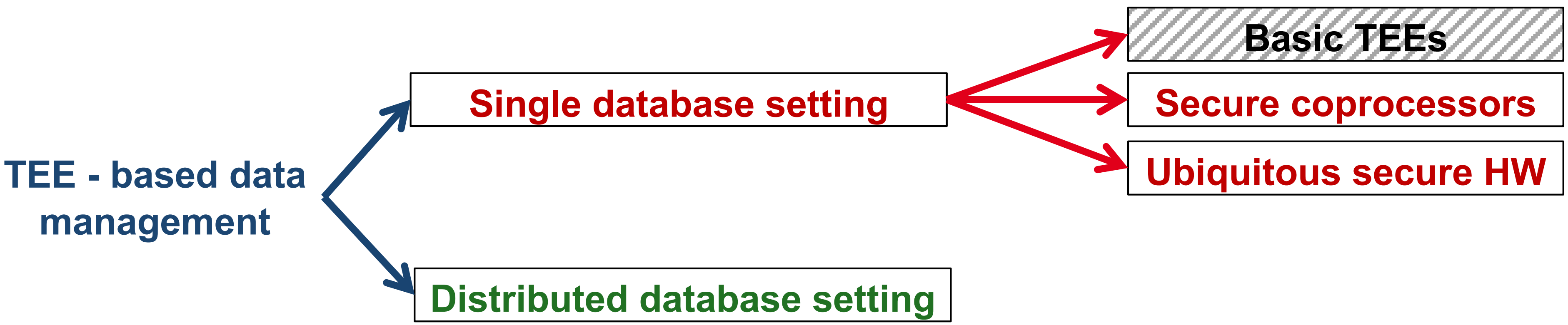
- Resource constrained devices (i.e., tamper-resistant CPUs such as smart cards or secure MCUs)
- Secure data tokens and embedded data management systems
  - **(see previous tutorials [ANSP13, ANSP14])**

Specialized secure coprocessors (since early 2010)

- Incorporate secure coprocessors to secure and scale outsourced DBs
- TrustedDB (using IBM 4764/5) or Chipherbase (using FPGA)

Ubiquitous secure HW support (recent years)

- **Intel SGX**, ARM TurstZone, AMD SME/SEV …
- Explosion of works dealing with secure data management in TEEs (EnclaveDB, secure KVS, HardIDX, Oblix, ObliDB, …)



A Bit of History & outline of part II

Secure HW support evolution

B. Secure distributed database management in TEEs

Basic TEEs (since early 2000)

Dedicated HW, resource constrained

Specific protocols, tailored for target HW

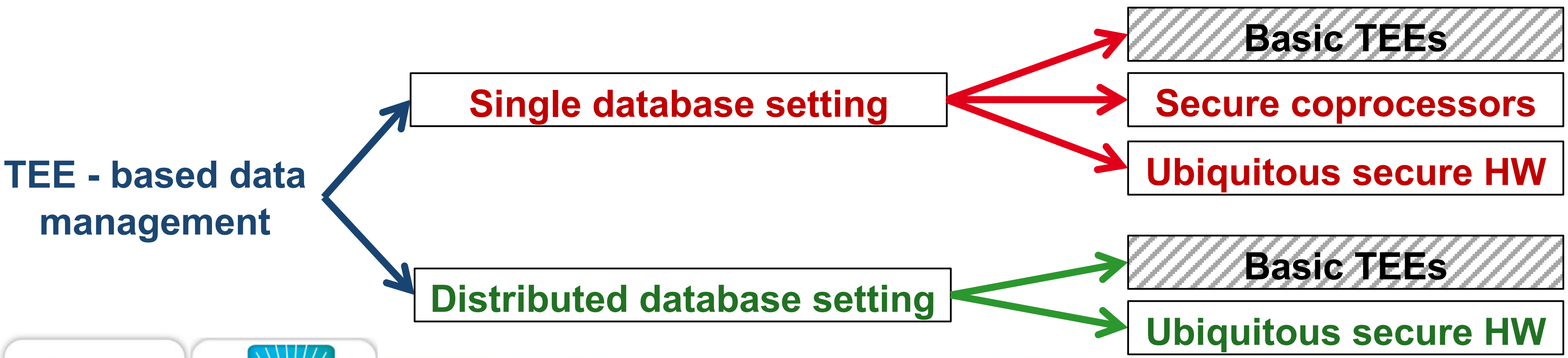
(see previous tutorials [ANSP13, ANSP14])

Ubiquitous secure HW support (recent years)

Intel SGX, ARM TurstZone, AMD SME/SEV …

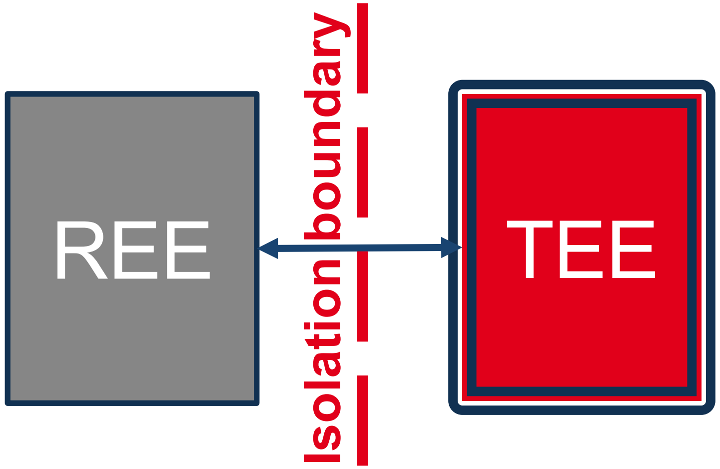
Confidentiality & integrity guarantees from multiple TEEs

Examples: VC3, M2R, lightweight-MR, Oblivious-ML, Opaque (spark SQL) …



Secure (Single) Database Management in TEEs

Common general architecture (for existing basic TEEs, secure co-CPUs/FPGAs, recent TEEs-Intel SGX): trusted vs. untrusted memory space



What to look for in details?

- HW architecture: inherent limitations of the HW (e.g., SCPU clock, size of the secure RAM, bandwidth between secure/unsecure worlds…)
- SW architecture: which modules run inside the secure HW => Objective: minimize the Trusted Computing Base (TCB) vs. efficiency (REE/TEE context switching)
- Security guarantees: access pattern leak vs. oblivious query processing

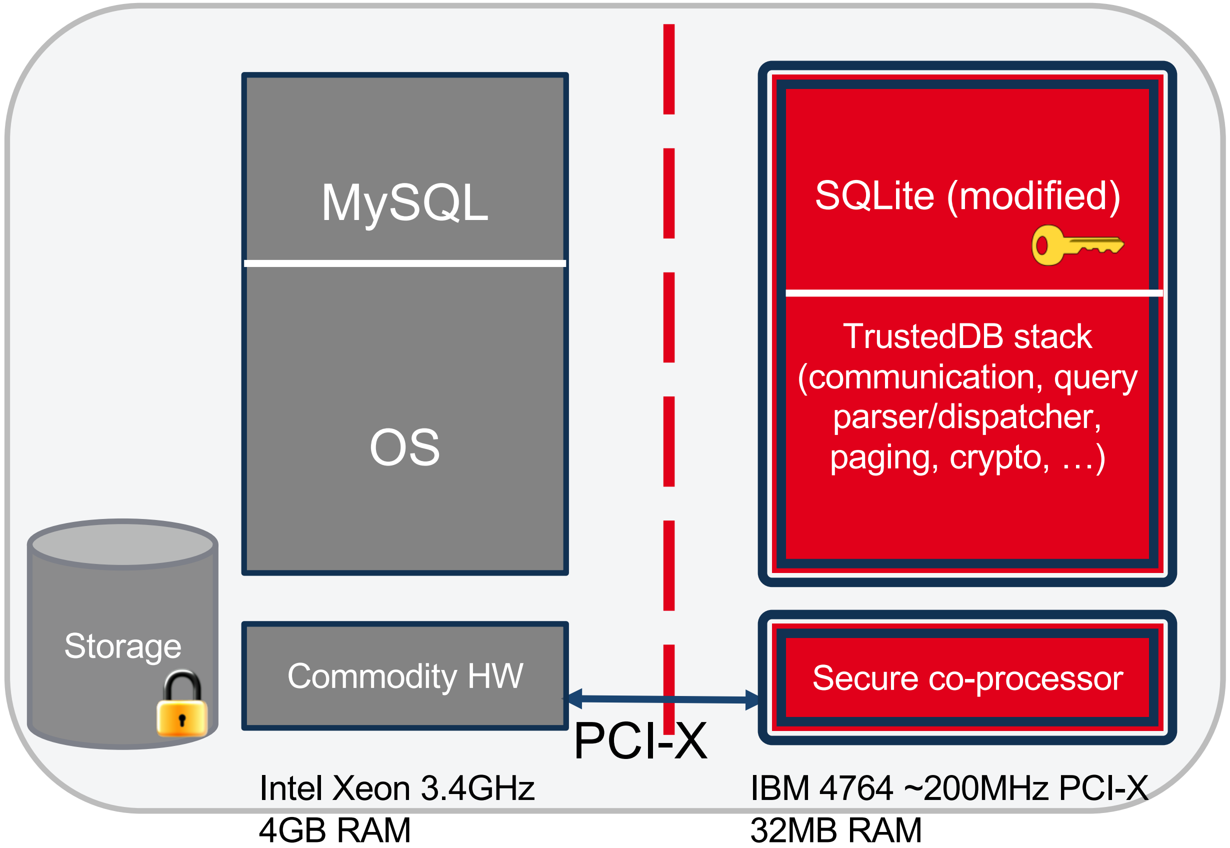
Adversary: untrusted, curious and controls the system

- Assumption: TEE isolation cannot be bypassed by an attacker controlling the system

Specialized Secure Coprocessors - TrustedDB

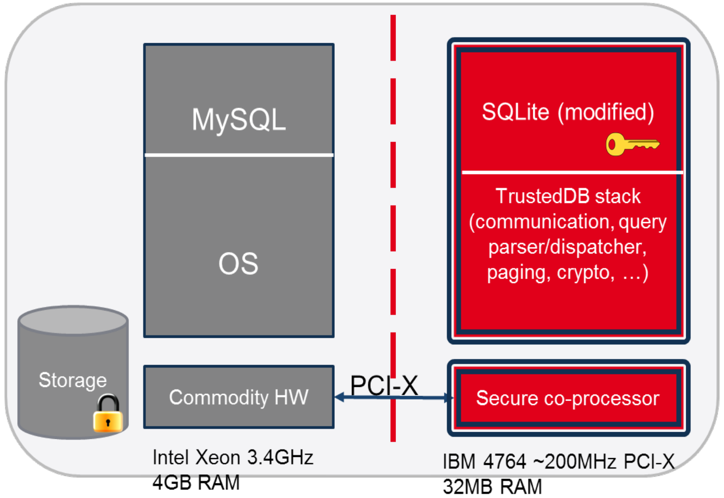
TrustedDB [BS11]

- Relational DB query processing with data confidentiality
- Split query processing: public data (MySQL) + private data (SQLite)

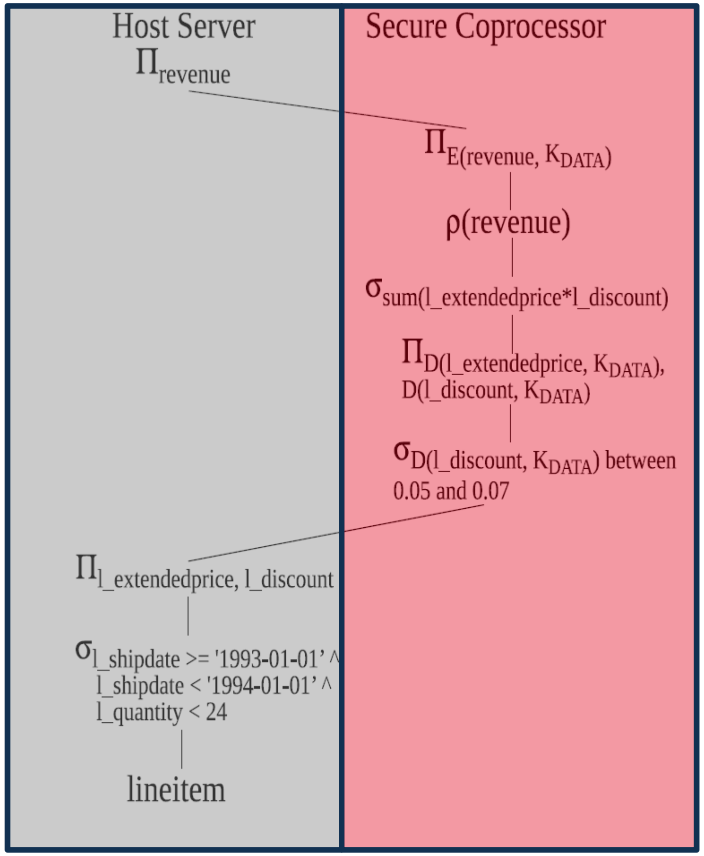


TrustedDB [BS11]

Query evaluation




```
SELECT SUM(l_extendedprice * l_discount) as revenue
FROM lineitem
WHERE     l_shipdate >= '1993-01-01'        AND
          l_shipdate < '1994-01-01'  AND
          l_discount between 0.05 and 0.07   AND
          l_quantity < 24
```



Specialized Secure Coprocessors – Cipherbase

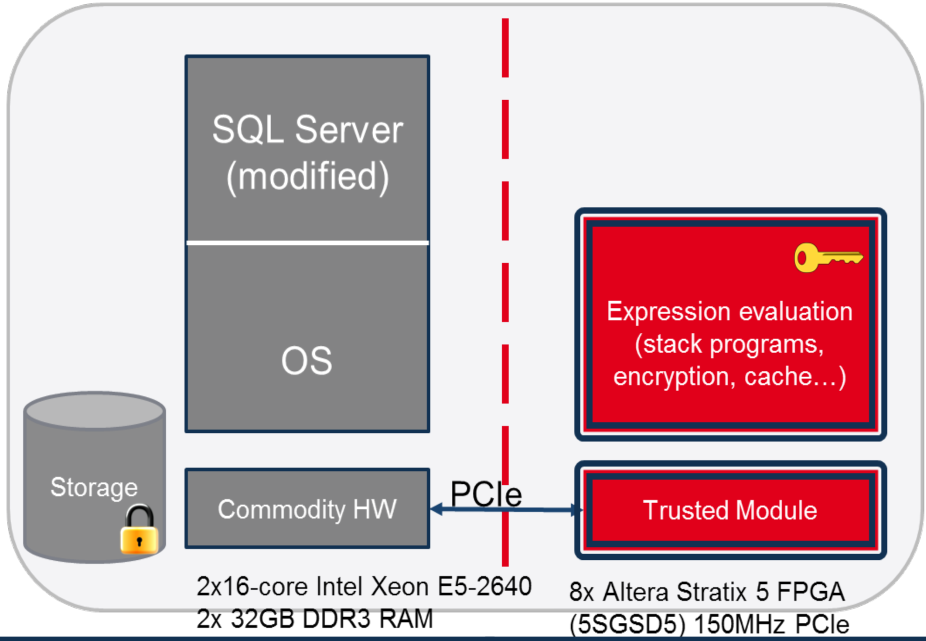
[AEK14, AEJ+15]

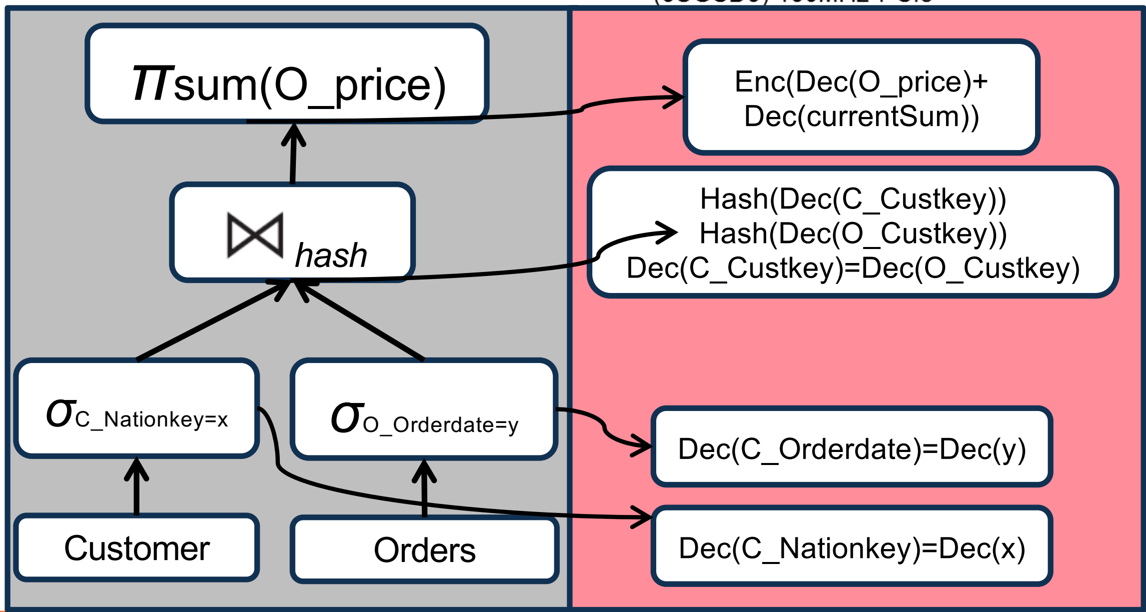
Relational DB query…

with data confidentiality

Database processing

- Mostly done in the REE (by modified SQL server), i.e., whenever the value semantics is not needed
- Large number of fine-grained TM accesses for expression evaluations





Specialized Secure Coprocessors - Conclusion

The good

- Rich functionality (DBMS-like) with good performance (much better than cryptographic-based solution)
- Strong data confidentiality guarantees
- Do not have to trade functionality or confidentiality for performance

The tradeoffs

- TCB vs. performance vs. SW portability
- Smaller (TCB) is better
  - E.g., TCB of Cipherbase < TCB of TrustedDB
- Specificity of secure HW and platform can impose specific data processing optimizations => this can impact the code portability
  - E.g., TrustedDB requires less SW engineering but is less portable than Cipherbase

…and the issues

- Variety and availability of secure HW and its specificity (RAM and cache size, CPU clock, bus speed, …) => (partially) solved by the new generation of secure HW (e.g., Intel SGX)
- TrustedDB and Cipherbase leak access patterns (*intrinsic to the REE/TEE architecture*) => need oblivious query processing

Outline of part II (TEE)

TEE - based data management

- **single database setting**
  - Basic TEEs (see [ANS14])
  - Secure coprocessors (TrustedDB, Cipherbase)
  - **Ubiquitous secure HW** (EnclaveDB, HardIDX, secure KVS, Oblix/ObliDB)
- **distributed database setting**
  - Basic TEEs (see [ANS14])
  - **Ubiquitous secure HW** (VC3, M2R, ObliviousML, Opaque…)

Ubiquitous Secure HW Support – 1. Efficient Data Processing

- **Modern HW, e.g. Intel SGX, democratize the access to trusted execution technologies**
  - Main CPU chip offers TEE capabilities through enclaves (special CPU mode enabled via new instructions) => ubiquitous access to TEE and strong (HW) integration between REE/TEE
  - Yet, performance considerations remain critical for minimizing the enclave related overheads
- **Main overhead sources with SGX enclaves [WAK18] [PVC18]**
  - Memory encryption and integrity checking: unavoidable but low overhead
  - Enclave transitions (ECALL/OCALL): high overhead
  - Enclave paging (related to a limited enclave size): high overhead

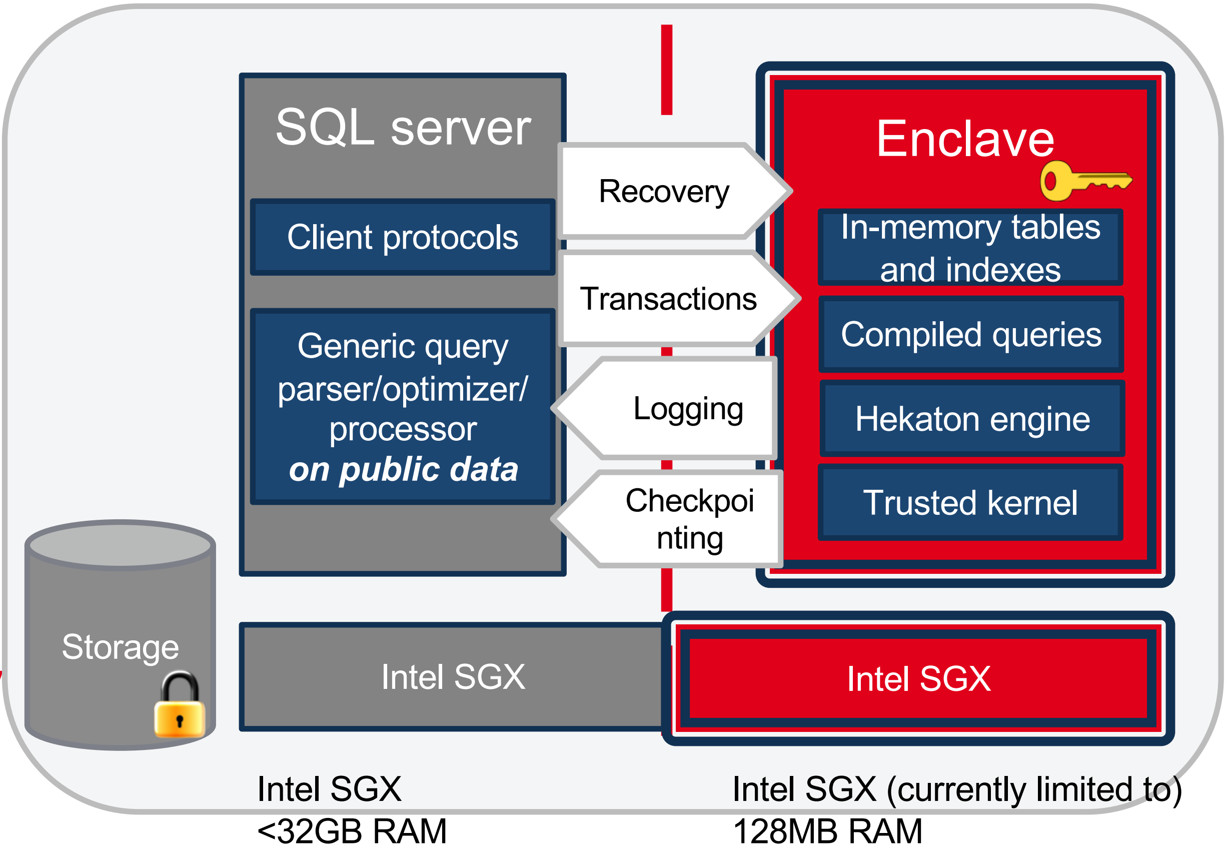
It requires carefully redesigning (data-oriented) apps

Ubiquitous Secure HW Support – EnclaveDB [PVC18]

High performance DB engine…
with security using Intel SGX

Important assumption: <u>all sensitive data loaded in enclave memory</u>
- No need for expensive SW encryption/integrity checks
- In-memory enclave data minimizes the leakage of sensitive information
- Also minimizes the number of costly IN/OUT enclave transitions
- Smaller TCB (Heckaton engine) using precompiled procedures

➔ Focus on secure and efficient DB logging and recovery
- Efficient protocol for checking integrity and freshness of the DB log
- Low overhead (~40%) compared with classical industry in-memory DBs



Ubiquitous Secure HW Support – Indexing/KVS

- **HardIDX [FBB+18]: secure and efficient B-tree indexing using SGX**
  - Leverage SGX enclaves to secure outsourced data searches while maintaining high query performance
  - Several order of magnitude lower query processing time than with traditional compared with the best known searchable encryption schemes…
  - … with similar level of confidentiality protection
- **eLSM [TCL+19]: authenticated KVS with TEE enclaves**
  - Focuses on optimizing update-oriented workloads…
  - … and ensuring query authenticity: integrity, completeness and freshness
  - Modifies the classical LSM-tree to cope with SGX enclave constraints
- **Both HardIDX and eLSM leak the access patterns**

Ubiquitous Secure HW Support – 2. Advanced Security

- **TEEs do not protect accesses outside the secure enclave**
  - **Loading everything inside the enclave is not always an option**
  - **Known side channel attacks with Intel SGX: OS can observe the enclave data accesses at the granularity of pages**
- **Access patterns in the workflow can reveal information (e.g., order, frequency distribution) for disk resident data**

Example:

1. Query Alice's age
2. Query number of people who commited tax fraud
3. If record retrieved in 1 is also retrieved in 2, Alice commited tax fraud

Oblivious Query processing

Idea: make sure memory access patterns are data independent (except for query input/output size) [**AK13**]

Ensures that the only leakage from a query is the the size of input output, even if the adversary observes memory.

- i.e. semantic security for queries

Relevant here: Adversary is assumed to control all memory external to secure hardware.

Oblivious Query processing using ORAM (Opaque [ZDB+17])

Problem: Memory accesses outside enclave leaked

Idea: Use existing cryptographic primitives: store data in an oblivious RAM

ORAM = Using a small private memory, and a large external encrypted memory, ensures that accessing two times the same item or two different items looks the same for the adversary.

Opaque: Uses ORAM with private memory within the enclave, and external RAM as external memory

Advantage: Can reuse an existig DBMS adding an ORAM layer for memory accesses

Problem: each memory access costs O(log²(|DB|) – in practice ~x50

Can we do better? From [AK2013] to ObliDB [EZ17]

ORAM is expensive and too general.

Idea: Do not store all data in an ORAM, implement specific algorithms that make sure data access is independent, only use (expensive) ORAM when no oblivious algorithms exits.

- **Example: Use linear scans instead of using indexes for selection.**
  - **More complex for joins, aggregates**

- **Advantage: smaller overhead w.r.t. no security**
- **Problem: cannot reuse existing DBMS with little modification,**
  - **everything needs to be reimplemented, choose right algo for right size of database**

A closer look at indexes? Oblix [MPC+18]

Assume index does not fit within enclave

- **i.e. loading the whole index within enclave and reading it impossible**

Oblix: use ORAM, but is it enough ?

- **Recent attacks : memory accesses within enclave are not entirely private (at page level)**
  - /!\ ORAM assumption of perfectly protected computing environment with private memory does not hold !
- **Specifically important problem for indexes as sucessive searches performed on the same index leak more and more data…**

Idea: memory accesses within the enclave (before accessing external ORAM) must be data independent !

- **i.e. make programs running inside the enclave oblivious**
- **→ Doubly oblivious schemes**

What if query code cannot be trusted (Ryoan [HZX18])

Problem: TEE do not ensure that malicious code cannot leak data on purpose

Ryoan: Distributed services for a data provider

- Uses sandboxing + TEEs + countermeasures for executing a service while protecting both code and data
- Code provider and data provider distinct
- Uses labels to ensure intended workflow is respected and result only disclosed to data provider

Problem: No memory outside enclave, what about leakage for memory within enclave?

Outline of part II (TEE)

TEE - based data management

- single database setting
  - **Basic TEEs (see [ANS14])**
  - Secure coprocessors (TrustedDB, Cipherbase)
  - Ubiquitous secure HW (EnclaveDB, HardIDX, secure KVS, Oblix/ObliDB)
- **distributed database setting**
  - **Basic TEEs (see [ANS14])**
  - **Ubiquitous secure HW (VC3, M2R, ObliviousML, Opaque…)**

TEE-based distributed databases

- Problem statement: How can we perform collaborative computation securely, without giving all data to a trusted third party?
- Single user/database/query code but outsourced computation => obtain confidentiality/integrity guarantees from multiple TEEs
  - Difficulty: obtain integrity/confidentiality from multiple TEEs
  - VC3, M2R (and also: lightweight mapreduce [PGF+17], Oblivious-ML [OSF+16]…)
- Multiple user/db and trusted (validated) query code
  - Difficulty: provide trust to multiple users (close to MPC problem) [LAP+19]

Distributing computation among several TEEs (1)

VC3 [SCF+15]: map reduce framework

Goal: Distribute computation among enclaves, keep data/computation secret, provide integrity guarantees to controler

Difficulties:

- **Establishing trust between multiple TEEs, and a controler**
- **Without sacrificing efficiency**
- **(Distributing tasks without disclosing code)**

Trust obtained via attestation (between TTEs and to the controler) + secure channels between enclaves

Problems: communication flow might leak information + single controler

Distributing computation among several TEEs (2)

[OCF+15], M2R [DSC+15]: map reduce framework

Goal: Address leakage via communication flow

Difficulty: must break the link between data/input of mapper and output of mapper/ input of reducer. Cannot have a single enclave processing all data.

Solution: add « anonymity of inputs » via shuffling, distribute shuffling between multiple enclaves, while keeping strong guarantees.

Problem: single controler

Distributing trust between parties [LAP+19]

Difficulty: **No single authority can guarantees good execution**

Using attestion and a monitoring enclave, ensure:

- **All participants actually execute the computation within TEE**
  Using attestations, ensure everybody executes same monitor
- **All participants agree on computation**
  Propagating attestions between participants
- **Data never leaves TEEs and only result is disclosed**
  Isolation property
- **Side channel attacks distributed by distribution of data**

Problems:

Need to (re)implement all DB algo in this framework

Distributing while minimizing potential leakage non-trivial

Back to the PDMS context

- TEEs undeniably grew to be a first class of solutions towards privacy-preserving data management
  - And the PDMS context makes no exception (on the contrary)
- Can we claim that current TEE-based solutions fundamentally address the ***extensible and secure PDMS*** problem?
- Hard to say as:
  - Majority of TEE-based data management consider the classical enterprise/outsourced DBMS context (but a lot can be reused).
  - The case of large scale distributed computations is mostly considered for single data provider, and single controller (but a lot of good ideas).

➔ Focus on the specificities of the PDMS context: next part

Thanks !

Questions ?

References (1)

[AAB+10] T. Allard, N. Anciaux, L. Bouganim, Y. Guo, L. L. Folgoc, B. Nguyen, P. Pucheral, I. Ray, S. Yin. Secure personal data servers: a vision paper. PVLDB, 3(1), 25-35, 2010.

[ABB+19] N. Anciaux, P. Bonnet, L. Bouganim, B. Nguyen, P. Pucheral, I. S. Popa, G. Scerri. Personal data management systems: The security and functionality standpoint. Inf. Syst., 80:13–35, 2019.

[ABD+19] M. Acosta, T. Berners-Lee, A. Dimou, J. Domingue, L-D. Ibá, K. Janowicz, M-E. Vidal, A. Zaveri: The FAIR TRADE Framework for Assessing Decentralised Data Solutions. WWW 2019

[ABP+14] N. Anciaux, L. Bouganim, P. Pucheral, Y. Guo, L. L. Folgoc, S. Yin. Milo-DB: a personal, secure and portable database machine. Distributed and Parallel Databases, 32(1):37–63, 2014.

[AEJ+15] A. Arasu, K. Eguro, M. Joglekar, R. Kaushik, D. Kossmann, R. Ramamurthy: Transaction processing on confidential data using cipherbase. ICDE 2015: 435-446

[AEK+14] A. Arasu, K. Eguro, R. Kaushik, R. Ramamurthy: Querying encrypted data. SIGMOD Conference 2014: 1259-1261

[AK13] A. Arasu, R. Kaushik: Oblivious Query Processing. ICDT 2014.

[ALS+15] N. Anciaux, S. Lallali, I. Sandu Popa, P. Pucheral: A Scalable Search Engine for Mass Storage Smart Objects. PVLDB 8(9): 910-921 (2015)

[ANS13] N. Anciaux, B. Nguyen, I. Sandu Popa: Personal Data Management with Secure Hardware: How to Keep Your Data at Hand. MDM (2) 2013: 1-2

[ANS14] N. Anciaux, B. Nguyen, I. Sandu Popa: Tutorial: Managing Personal Data with Strong Privacy Guarantees. EDBT 2014: 672-673

References (2)

[BBB+17] R. Bahmani, M. Barbosa, F. Brasser, B. Portela, A.-R. Sadeghi, G. Scerri, B. Warinschi: Secure Multiparty Computation from SGX. Financial Cryptography 2017: 477-497

[BEE+17] J. Bater, G. Elliott, C. Eggen, S. Goel, A. Kho, J. Rogers: SMCQL: secure querying for federated databases. PVLDB 2017

[BGC+18] V. Bindschaedler, P. Grubbs, D. Cash, T. Ristenpart, V. Shmatikov: The tao of inference in privacy-protected databases. PVLDB 2018

[BPS+16] M. Barbosa, B. Portela, G. Scerri, B. Warinschi: Foundations of Hardware-Based Attested Computation and Application to SGX. EuroS&P 2016: 245-260

[BS11] S. Bajaj, R. Sion: TrustedDB: a trusted hardware-based database with privacy and data confidentiality. SIGMOD Conference 2011: 205-216

[DSC+15] T. T. A. Dinh, P. Saxena, E. Chang, B. C. Ooi, C. Zhang: M2R: Enabling Stronger Privacy in MapReduce Computation. USENIX Security 2015

[EZ17] S. Eskandarian, M. Zaharia: An oblivious general-purpose SQL database for the cloud. CoRR, abs/1710.00458, 2017

[FBB+18] B. Fuhry, R. Bahmani, F. Brasser, F. Hahn, F. Kerschbaum, A.-R. Sadeghi: HardIDX: Practical and secure index with SGX in a malicious environment. Journal of Computer Security 26(5): 677-706 (2018)

[HZX18] T. Hunt, Z. Zhu, Y. Xu, S. Peter, E. Witchel: Ryoan: A Distributed Sandbox for Untrusted Computation on Secret Data. ACM Trans. Comput. Syst. 35(4): 13:1-13:32 (2018)

References (3)

[LAP+19] R. Ladjel, N. Anciaux, P. Pucheral, G. Scerri. Trustworthy Distributed Computations on Personal Data Using Trusted Execution Environments. TrustCom, 2019.

[LAS+17] S. Lallali, N. Anciaux, I. Sandu Popa, P. Pucheral: Supporting secure keyword search in the personal cloud. Inf. Syst. 72: 1-26 (2017)

[LSB19a] J. Loudet, I. Sandu Popa, L. Bouganim: SEP2P: Secure and Efficient P2P Personal Data Processing. EDBT 2019.

[LSB19b] J. Loudet, I. Sandu-Popa, L. Bouganim. DISPERS: Securing Highly Distributed Queries on Personal Data Management Systems. PVLDB 2019

[LWG+13] S. Lee, E.L. Wong, D. Goel, M. Dahlin, V. Shmatikov, πbox: A platform for privacy-preserving apps, in: NSDI, 2013.

[MPC+18] P. Mishra, R. Poddar, J. Chen, A. Chiesa, R. A. Popa: Oblix: An Efficient Oblivious Search Index. S&P 2018.

[MSW+14] Y-A. de Montjoye, E. Shmueli, SS. Wang, AS. Pentland: OpenPDS: Protecting the Privacy of Metadata through SafeAnswers. PLoS ONE 9(7) 2014

[MZC+16] R. Mortier, J. Zhao, J. Crowcroft, L. Wang, Q. Li, H. Haddadi, Y. Amar, A. Crabtree, J. Colley, T. Lodge, T. Brown, D. McAuley, C. Greenhalgh: Personal Data Management with the Databox: What's Inside the Box? ACM CoNEXT Cloud-Assisted Networking workshop, 2016

[OCF+15] O. Ohrimenko, M. Costa, C. Fournet, C. Gkantsidis, M. Kohlweiss, D.Sharma: Observing and Preventing Leakage in MapReduce. CCS 2015.

References (4)

[OSF+16] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, M. Costa: Oblivious Multi-Party Machine Learning on Trusted Processors. USENIX Security 2016.

[PGF+17] R. Pires, D. Gavril, P. Felber, E. Onica, M. Pasin: A lightweight MapReduce framework for secure processing with SGX. CCGrid 2017

[PVC18] C. Priebe, K. Vaswani, M. Costa: EnclaveDB: A Secure Database Using SGX. IEEE Symposium on Security and Privacy 2018: 264-278

[RHM19] L. Roche, J. M. Hendrickx, Y-A. de Montjoye: Estimating the success of re-identifications in incomplete datasets using generative models. Nature Communications 2019

[SCF+15] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, M. Russinovich: VC3: Trustworthy Data Analytics in the Cloud Using SGX. S&P 2015

[TAP17] P. Tran-Van, N. Anciaux, P. Pucheral: SWYSWYK: A Privacy-by-Design Paradigm for Personal Information Management Systems. ISD 2017

[TCL+19] Y. Tang, J. Chen, K. Li, J. Xu, Q. Zhang: Authenticated Key-Value Stores with Hardware Enclaves. CoRR abs/1904.12068 (2019)

[WAK18] N. Weichbrodt, P.-L. Aublin, R. Kapitza: SGX-perf: A Performance Analysis Tool for Intel SGX Enclaves. Middleware 2018

[ZDB+17] W. Zheng, A. Dave, J. G. Beekman, R. A. Popa, J. E. Gonzalez, I. Stoica. Opaque: An oblivious and encrypted distributed analytics platform. NSDI 2017