Personal Database Security and Trusted Execution Environments: A Tutorial at the Crossroads
Personal data...
... at the crossroads of Business and Privacy

From the business perspective...
Personalized services (e.g., personalized searches, pay-as-you-xxx),
... and needed optimizations (e.g., energy consumption, network ...),
Various features improving business
... like targeted ads, improved CRM, increased time spend in social medias and games, etc.

Ultimate profiling

Source: crackedlabs.org
Personal data…
… at the crossroads of Business and Privacy
to societal concerns…

Silent over-collection of personal data
  Eg: corp. (Alexa, Fortnite), gov. (Health Data Hub)

Recurrent/massive leaks & attacks
  Eg: Yahoo, Equifax, Cambridge Analytica…

Anonymised datasets often not anonymous
  Eg.: 15 fields is enough [RHM19]

Uses considered questionable
  Eg: Social medias (Visa, Insurance)
      Personal reports (Pipl, Intelius…),

Discriminatory uses of personal data
  Eg: criterias in targeted ads,
      e-justice, recruiting process
      23andMe vs. GINA, …

© banksy
Personal data…
… at the crossroads of Business and Privacy

… more advocacy of privacy issues & more acceptance by economic actors

Legislation
GDPR, Facial recognition forbidden in SF, California Consumer Privacy Act (CCPA), With fines applied

More acceptance
Symptoms of a crisis of consciousness (e.g., Time well spent)
From “privacy is no longer the social norms”
… to “private is the future”
Privacy-based marketing campaigns
Current trend: give their personal data (agency) back to individuals

Act I: the right to Data portability
... the right to retrieve its own data

Act II: Personal Data Mg¹ Systems (PDMS)
... the tool to manage its own data
Is this enough to change the situation? …

Individual’s agency

Let individuals freely decide about the new usages of their data all along their life cycle

Rather than: services in exchange of personal data

Secured decentralized architectures

Offer individuals the ability to securely control the raw data produced on their side

Rather than: centralizing everything in a few hands

TWO prerequisites!
Is this enough to change the situation? …

Individual’s agency

Let individuals freely decide about the **new usages** of their data all **along their life cycle**

Rather than: services in exchange of personal data

Major steps of personal data life-cycle escape today individual’s control

Architectural considerations of a the PDMS platform are paramount

Secured decentralized architectures

Offer individuals the ability to **securely control** the raw data produced on their side

Rather than: centralizing everything in a few hands

Layman citizen … as security expert?

Emergence of Trusted Execut° Env't (high-end servers & edges)

The primary topic of this tutorial!
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
10 years history of Personal Data Management Systems

Since 2008 – FreedomBox@Columbia (Eben Moglen)
   Free individuals from state control
   PDMS = Low-cost open HW + open SW

Since 2010 – PDS@Inria [AAB+10], MiloDB [ABP+14], PDMS [ABB+19]
   Manage (specific) personal folders at hand, enforce privacy policies
   PDMS = Tamper resistant HW (smart card or TEEs) + embedded DBMS

2012 – OpenPDS@MIT [MSW+14], 2016 – DataBox-BBCBox@Nottingham [MZC+16]
   Manage your data locally, externalize only safe answers
   PDMS = SW running on user’s device (smartphone, tablet)

Since 2013 – Gov. [MyDex, MesInfos] & commercial initiatives [NextCloud, Cozy, …]
   Collect personal data from different data silos & provide transversal Apps
   PDMS = Online SW with Apps (terminology shift: PDS → personal cloud)

Since 2018 – Solid PODs and Inrupt (Tim Berner Lee)
   To re-decentralize the Web of personal data, give agency to individuals
   PDMS = Personal Online Data store (PODs)
10 years history of Personal Data Management Systems

Since 2008 – FreedomBox@Columbia (Eben Moglen)
    Free individuals from state control
    PDMS = Low-cost open HW + open SW

Since 2010 – PDS@Inria [AAB+10], MiloDB [ABP+14], PDMS [ABB+19]
    Manage (specific) personal folders at hand, enforce privacy policies
    PDMS = Tamper resistant HW (smart card or TEEs) + emulated

2012 – OpenPDS@MIT [MSW+14], 2016 – DataBox@UCommons [MZC+16]
    Manage your data locally, externalize only safe answers
    PDMS = SW running on user’s device (smartphone, tablet)

Since 2013 – Commercial initiatives [NextCloud, Cozy, …]
    Collect, connect data silos & provide transversal Apps
    PDMS = Online SW with Apps (terminology shift: PDS → personal cloud)

Since 2018 – Solid PODs and Inrupt (Tim Berner Lee)
    To re-decentralize the Web of personal data, give agency to individuals
    PDMS = Personal Online Data store (PODs)

What are their functionalities?
What are the privacy threats considered?

New HW since 3/19
Main classes of architectures for a PDMS

Online personal cloud

E.g., Cozy, Digi.me, NextCloud, BitsAbout.Me, Perkeep

Functionality:
- Data collectors for everything (banks, energy, health, geolocation, ‘likes’ graphs, ...)
- Personal (cross-)computation (1 individual) features for App developers
- Backup (full retention: Perkeep)

Trust model:
- Personal cloud provider & Apps considered fully honest
- Security standards, PEN tests (Cozy), code transparency (community checks)

No-knowledge personal cloud

E.g., MyDex, SpiderOak, Digi.me

Functionality:
- Secure data store, personal data encrypted (encryption keys managed at client side)
- Secure backup and point in time recovery

Trust model:
- Personal cloud provider is untrusted (but the client device is not)
- Considered attacks: data snooping and secondary usages (server), ransomware (client)
Main classes of architectures for a PDMS

Online personal cloud ➔ Advanced functionality, strong trust assumptions
E.g., Cozy, Digi.me, NextCloud, BitsAbout.Me, Perkeep
Functionality:
- Data collectors for everything (banks, energy, health, geolocation, ‘likes’ graphs, ...)
- Personal (cross-)computation (1 individual) features for App developers
- Backup (full retention: Perkeep)
Trust model:
- Personal cloud provider & Apps considered fully honest
- Security standards, PEN tests (Cozy), code transparency (community checks)

No-knowledge personal cloud ➔ Increased security, minimalist functionality
E.g., MyDex, SpiderOak, Digi.me
Functionality:
- Secure data store, personal data encrypted (encryption keys managed at client side)
- Secure backup and point in time recovery
Trust model:
- Personal cloud provider is untrusted (but the client device is not)
- Considered attacks: data snooping and secondary usages (server), ransomware (client)
Main classes of architectures for a PDMS (cont.)

Home (or edge) cloud software

E.g., OpenPDS [MSW+14], Databox [MZC+16]

Functionality:

- Trusted storage on end-user device or at the edge (1 store per IoT device)
- Personal computation provided safe answers and aggregated views, never raw data
- Data dissemination rules to share computed results

Trust model: user device and SW must be trusted

Home cloud plugs (dedicated)

E.g., FreedomBox, CloudLocker

Functionality: data store and backup in a dedicated hardware plug

Trust model: Plug code must be trusted (dedicated => limited attack surface)

Tamper-resistant home cloud

E.g., PDS [AAB+10], PlugDB [ANSP14, ALSP+15, LASP+17, ABB+19]

Functionality: (simple) store, share, compute (local/global) in a secure HW device

Trust model: secure HW + embedded SW are trusted
Main classes of architectures for a PDMS (cont.)

Home (or edge) cloud software ➔ ‘formal’ security lost, more functionality

E.g., OpenPDS [MSW+14], Databox [MZC+16]

Functionality:
- Trusted storage on end-user device or at the edge (1 store per IoT device)
- Personal computation provided safe answers and aggregated views, never raw data
- Data dissemination rules to share computed results

Trust model: user device and SW must be trusted

Home cloud plugs (dedicated)

E.g., FreedomBox, CloudLocker

Functionality: data store and backup

Trust model: Plug code must be trusted ➔ Security at the price of functionality, advanced processing on untrusted device

Tamper-resistant home cloud

E.g., PDS [AAB+10], PlugDB [ANSP14, ALSP+15, LASP+17, ABB+19]

Functionality: (simple) store, share, compute (local/global) in a secure HW device

Trust model: secure HW + embedded SW are trusted
# Synthesis: functionalities

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Online personal cloud</th>
<th>Zero-knowledge personal cloud</th>
<th>Home cloud software</th>
<th>Home cloud plug</th>
<th>Tamper resistant home cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Storage</strong></td>
<td>Regular DBMS</td>
<td>Files/KVS store</td>
<td>Files/KVS/DBMS at user-side</td>
<td>Files/KVS in the Plug</td>
<td>Embedded DBMS</td>
</tr>
<tr>
<td><strong>Backup</strong></td>
<td>Regular DBMS</td>
<td>Encrypted archive, Pt-in-time recovery</td>
<td>Replication / offline store</td>
<td>Replication / offline store</td>
<td>Replication / offline store</td>
</tr>
<tr>
<td><strong>Data collection</strong></td>
<td>Web scrapping</td>
<td>By users / Apps</td>
<td>By users / Apps</td>
<td>By users</td>
<td>By users / Apps</td>
</tr>
<tr>
<td><strong>Personal computations</strong></td>
<td>Linked/ transversal queries</td>
<td>Apps level</td>
<td>Safe answer, local data aggregation</td>
<td>Apps level</td>
<td>Simple transversal queries</td>
</tr>
<tr>
<td><strong>Distributed computations</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Simple distributed SQL statistics at large scale</td>
</tr>
<tr>
<td><strong>Data dissemination</strong></td>
<td>[synchro.]</td>
<td>At Apps level</td>
<td>Privileges for 3rd parties and Apps</td>
<td>[synchro.]</td>
<td>Privileges for 3rd parties and Apps, Secure AC</td>
</tr>
</tbody>
</table>

1. The whole personal cloud data life-cycle must be covered!
## Synthesis : functionalities (cont.)

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Online personal cloud</strong></td>
</tr>
<tr>
<td><strong>Storage</strong></td>
<td>Regular DBMS</td>
</tr>
<tr>
<td><strong>Backup</strong></td>
<td>Regular DBMS</td>
</tr>
<tr>
<td><strong>Data collection</strong></td>
<td>Web scrapping</td>
</tr>
<tr>
<td><strong>Personal computations</strong></td>
<td>Linked/ transversal queries</td>
</tr>
<tr>
<td><strong>Distributed computations</strong></td>
<td>[X]</td>
</tr>
<tr>
<td><strong>Data dissemination</strong></td>
<td>[synchro.]</td>
</tr>
</tbody>
</table>

1- The whole personal cloud data life-cycle must be covered !
2- Distributed computations are poorly covered…
Less useful? No, Big-Data perspectives !
More difficult? Yes, efficient and secure (solutions in the tamper resistant context)
### Synthesis: trust

<table>
<thead>
<tr>
<th>Considered threats</th>
<th>Online personal cloud</th>
<th>Zero-knowledge personal cloud</th>
<th>Home cloud software</th>
<th>Home cloud Plug</th>
<th>Tamper resistant personal server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data snooping</td>
<td>Data snooping</td>
<td>Data snooping</td>
<td>Data snooping</td>
<td>Data snooping</td>
<td>Data snooping</td>
</tr>
<tr>
<td>Data leakage</td>
<td>Data leakage</td>
<td>Data leakage</td>
<td>Data leakage</td>
<td>Data leakage</td>
<td>Data leakage</td>
</tr>
<tr>
<td>2ndary usages</td>
<td>2ndary usages</td>
<td>2ndary usages</td>
<td>2ndary usages</td>
<td>2ndary usages</td>
<td>2ndary usages</td>
</tr>
<tr>
<td>Client Failure</td>
<td>Client Failure</td>
<td>Client Failure</td>
<td>Client Failure</td>
<td>Client Failure</td>
<td>Client Failure</td>
</tr>
<tr>
<td>Ransomware</td>
<td>Ransomware</td>
<td>Ransomware</td>
<td>Ransomware</td>
<td>Ransomware</td>
<td>Ransomware</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trust model</th>
<th>Fully-honest personal cloud &amp; Apps</th>
<th>Semi-honest or Malicious personal cloud Trusted Apps Trusted client</th>
<th>Trusted personal cloud Trusted client Untrusted Apps</th>
<th>Trusted personal cloud Trusted Plug Trusted Apps</th>
<th>Trusted personal cloud Semi-honest infra. Untrusted Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privacy and security measures</td>
<td>Security stds, Business model Open source</td>
<td>client-side encrypted ‘no-knowledge’ store</td>
<td>Safe answers Separated stores Local audit</td>
<td>Closed platform (dedicated device), physical ownership</td>
<td>Secure HW small TCB secure distributed protocols</td>
</tr>
</tbody>
</table>

1- different privacy threats considered, all must be circumvented
### Synthesis: trust (cont.)

<table>
<thead>
<tr>
<th>Representative Personal Cloud approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Online personal cloud</strong></td>
</tr>
<tr>
<td><strong>Considered threats</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Trust model</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Privacy and security measures</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

1. Different privacy threats considered, all must be circumvented
2. Unifying the solutions is not trivial (if not impossible)

Wide spectrum of architectural choices…

… but different – irreconcilable – trust models and security measures
Personal Data Management: anything new?

Objective:
(1) provide the set of functionalities
(2) address all threats

Decades of research in
…. secure data collection, storage, backup, queries!

Next:
Specificities of (individual’s) PDMS vs (corporate) DBMS
…. and derived properties for an extensive and secure PDMS
In [ABB+19]: 5 properties are defined…
Expected PDMS functionalities & properties: Data Collection

Corporate DBMS
A basic operation using wrappers/APIs
Well-known & predefined wrappers/APIs
… audited and patched by the admins

Individual’s PDMS
Primary data directly fed into user’s PDMS
Secondary data needs data scrapping
Huge set of scrappers
…with untrusted code (e.g., Weboob)
…accessing sensitive data (credentials)
…in an untrusted environment!

Property: A PDMS enforces *piped data collection* iff:
1- the only PDMS data, accessible to the data collector, is the credentials;
2- the credentials/collected data cannot be leaked outside the PDMS.

The only external channel provided to the data collector is with a single data provider
… and the code is suitably isolated not to leak data elsewhere
Expected PDMS functions & properties:  

**Corporate DBMS**  
Computations on corporate data  
Set of (trusted) applications selected, … audited and patched by admins

**Individual’s PDMS**  
Apps crossing several data from individual  
For the PDMS owner or an external service (e.g., Pay as you drive).  
Apps ‘move’ to data but…  
Apps are untrusted (user’s viewpoint)  
→ local data must not leak  
Computations are untrusted (service viewpt)  
→ results must be attested

**Property: A PDMS enforces bilaterally trusted computations iff:**  
1- the data computation can only access the expected data from the PDMS;  
2- only the final result – not the raw data – can be exposed to a 3rd party;  
3- it provides a proof that the result was produced by the expected code.  
‘Bilateral’ → guarantees to the owner and the 3rd party willing to execute code  
To owner: minimal collection principle is fulfilled, raw data cannot leak  
To 3rd party: code remotely sent has been computed (it may include any verification on data)
Expected PDMS functions & properties: **Collective computations**

**Corporate DBMS**
- Not common → practical solutions
  - e.g., few Hospitals run a collective query
  - A trusted party may be used (by contract)
  - SMC usable [BEE+17] (few participants)

**Individual’s PDMS**
- Common → new solutions are needed
  - e.g., Big Data and IA (recommendations,
    participative studies, community learning…)
  - Mutual confidentiality & integrity are critical
  - At a very large scale
  - (no trusted party nor SMC)

---

Property: A PDMS enforces *mutually trusted collective computations* iff:

1. the data computation can only access the required participant data;
2. only the final result – not the raw data – can be exposed to a 3rd party or any participant;
3. it provides a proof that the result was produced by the expected code on the expected set of participants.

‘Mutual’ → guarantees also hold between the participants
Definition of an Extensive and Secure PDMS (ES-PDMS)

An Extensive & Secure PDMS

provides the expected set of functionalities to cover the complete data life-cycle:
- data collection,
- storage and recovery,
- cross-computations,
- collective computations,
- data dissemination.

and is compliant with their respective security properties counterparts,
- piped data collection,
- mutual data at rest protection,
- bilaterally trusted personal computation,
- mutually trusted collective computation,
- controlled data dissemination.

How do we get there?
The field of TEE-based secure data management is rapidly developing
➤ let’s take a closer look…
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 computation ?

Would Trusted Execution Environments help?
Secure Element (SE) $\rightarrow$ Trusted Execut° Environment (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° Environment (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/collated 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. Secure (single) 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 TrustZone, 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
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 TrustZone, AMD SME/SEV …
- Confidentiality & integrity guarantees from multiple TEEs
- Examples: VC3, M2R, lightweight-MR, Oblivious-ML, Opaque (spark SQL) …

A Bit of History & outline of part II

- Single database setting
- Distributed database setting
- Secure coprocessors
- Basic TEEs
- Ubiquitous secure HW

TEE - based data management

Secure HW support evolution
Secure (Single) Database Management in TEEs

Common general architecture (for existing basic TEEs, secure co-CPU/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)

Diagram:
- MySQL
- OS
- SQLite (modified)
- TrustedDB stack (communication, query parser/discharger, paging, crypto, …)
- Commodity HW
- Secure co-processor
- PCI-X
- Intel Xeon 3.4GHz 4GB RAM
- IBM 4764 ~200MHz PCI-X 32MB RAM

Storage
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

\[ \sigma_{C_{Nationkey}=x} \pi_{\text{sum}(O_{price})} \]

\[ \sigma_{O_{Orderdate}=y} \]

\[ \text{hash} \]

\[ \text{Dec}(C_{Nationkey})=\text{Dec}(x) \]
\[ \text{Dec}(C_{Orderdate})=\text{Dec}(y) \]
\[ \text{Dec}(C_{Custkey})=\text{Dec}(O_{Custkey}) \]
\[ \text{Enc}(\text{Dec}(O_{price})+\text{Dec}(\text{currentSum})) \]
\[ \text{Hash}(\text{Dec}(C_{Custkey})) \]
\[ \text{Hash}(\text{Dec}(O_{Custkey})) \]
\[ \text{Dec}(C_{Custkey})=\text{Dec}(O_{Custkey}) \]

Customer

Orders

SQL Server (modified)

OS

Commodity HW

Storage

PCIe

2x16-core Intel Xeon E5-2640
2x 32GB DDR3 RAM
8x Altera Stratix 5 FPGA
(5SGSD5) 150MHz PCIe

Expression evaluation
(stack programs, encryption, cache...)
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
  - Distributed database setting

- Basic TEEs (see [ANS14])

- Secure coprocessors (TrustedDB, Cipherbase)

- Ubiquitous secure HW (EnclaveDB, HardIDX, secure KVS, Oblix/ObliDB)

- Basic TEEs (see [ANS14])

- Ubiquitous secure HW (VC3, M2R, ObliviousML, Opaque…)

PETRUS team – VLDB’19
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: **all sensitive data loaded in enclave memory**
- 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 (Hekaton 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
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 committed tax fraud
3. If record retrieved in 1 is also retrieved in 2, Alice committed 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 $O(\log^2(|DB|))$ – in practice $\sim x50$
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 successive 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

- Basic TEEs (see [ANS14])
  - Secure coprocessors (TrustedDB, Cipherbase)
  - Ubiquitous secure HW (EnclaveDB, HardIDX, secure KVS, Oblix/ObliDB)

- Single database setting

- 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 controller

Difficulties:
   Establishing trust between multiple TEEs, and a controller
   Without sacrificing efficiency
   (Distributing tasks without disclosing code)

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

Problems: communication flow might leak information + single controller
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 controller
Distributing trust between parties [LAP+19]

Difficulty: No single authority can guarantees good execution

Using attestation 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 attestations 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

TEE\text{s} undeniably grew to be a first class of solutions towards privacy-preserving data management  
\begin{itemize}
    \item And the PDMS context makes no exception (on the contrary)
\end{itemize}

Can we claim that current TEE-based solutions \textbf{fundamentally} address the \textit{extensible and secure PDMS} problem?

Hard to say as:
\begin{itemize}
    \item Majority of TEE-based data management consider the classical enterprise/outsourced DBMS context (but a lot can be reused).
    \item The case of large scale distributed computations is mostly considered for single data provider, and single controller (but a lot of good ideas).
\end{itemize}

\rightarrow \textbf{Focus on the specificities of the PDMS context: next part}
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
- Towards a reference logical PDMS architecture
- Reusing building blocks from the existing TEE-based solutions
- A quick view of remaining challenges
Definition of an Extensive and Secure PDMS (ES-PDMS)

Extensive & Secure

Provides the expected set of functionalities to cover the complete data life-cycle:
- data collection,
- storage and recovery,
- cross-computations,
- collective computations,
- data dissemination,

and is compliant with their respective security properties counterparts:
- piped data collection,
- mutual data at rest protection,
- bilaterally trusted personal computation,
- mutually trusted collective computation,
- controlled data dissemination.

ES-PDMS problem ➔ typical tension extensibility vs. security

TEEs are a prime opportunity to alleviate this tension

... but one needs to consider the atypical context of the Personal Cloud

Layman PDMS owner (and no DBA/DSA)

Completely open and hence untrusted system environment

Apps ‘move’ to data

Large scale collective computations among many individuals unknown to each other

Unclear legal responsibility of the PDMS owner w.r.t. third party data

...
Logical Architecture: Three Layers [ABB+19]

ES-PDMS problem $\Rightarrow$ tension extensibility – security

In practice: interesting processing is App-specific $\Rightarrow$ privacy violations are App-specific…

$\Rightarrow$ How to avoid data leaks and launch advanced data processing?
Logical Architecture: Three Layers [ABB+19]

ES-PDMS problem ➔ tension extensibility – security

In practice: interesting processing is App-specific ➔ privacy violations are App-specific…

➔ How to avoid data leaks and launch advanced data processing?

Three-layer architecture

Core (limited and secure)
- Trusted Computing Base (TCB) – small and (ideally) proven
- Data Storage, Policy enforcement, Communication

Data tasks (advanced and isolated/sandboxed)
- Untrusted code – potentially large
- Deal with (complex) app specific data management

Applications (Apps)
- No trust assumptions can be made (today)
- Manipulate results (but not raw data)
Logical Architecture: Three Layers [ABB+19]

ES-PDMS problem ➔ tension extensibility – security

In practice: interesting processing is App-specific ➔ privacy violations are App-specific… ➔ How to avoid data leaks and launch advanced data processing?

Reuse building blocks from the existing TEE-based solutions to implement the core
- e.g., EnclaveDB[PVC18], HardIDX[FBB+18], secure KVS[TCL+19], Oblix[MPC+18]…
- e.g., for secure and efficient DB logging/recovery, base query processing and indexing, protection against side-channels attacks…

Reuse building blocks from the existing TEE-based solutions to secure the Data tasks (see next)

Deal with (complex) app specific data management

Applications (Apps)

No trust assumptions can be made (today)
Manipulate results (but not raw data)
Satisfying *bilaterally trusted data computations* property

**Assumptions**
Execute any arbitrarily complex but untrusted computation code with access to some (large amounts of) PDMS raw data

**Requirements**
Computation code only accesses required raw data, only the result is shared and attested

- Manifest: collection rules + computation code + 3rd party accessing the result
- Data task runs computation code (__,__) + result declassification by the Core

Building blocks from [BPS+16, BBB+17]
Satisfying \textit{bilaterally trusted data computations} property

Assumptions
Execute any arbitrarily complex but untrusted computation code with access to some (large amounts of) PDMS raw data

Requirements
Computation code only accesses required raw data, only the result is shared and attested
- Manifest: collection rules + computation code + 3\textsuperscript{rd} party accessing the result
- Data task runs computation code \((\square, \square)\) + result declassification by the Core

Control several data tasks used simultaneously in a computation, e.g., such as in \pi Box [LWG+13]

Provide secure sandboxing inside a TEE enclave, e.g., with Ryoan [HZX18]
Remaining Challenges: Architectural level

Design a minimal and provable Core engine
- Minimal (in code size & complexity) set of modules
- Algebra of operators that cannot be delegated to Data tasks
- Support different data models vs. deal with data models/optimizations at Data task level?
- Design provably secure components

Establish a continuum of solutions (modular approach)
- How to map our logical architecture on different physical ones?
- How to avoid (re-)proving Core modules on different HW targets (SE, TrustZone, SGX...)?
Remaining Challenges: Architectural level (cont.)

Full (both ways) Data task isolation ➜ stateless data tasks

- TEEs offer code/data confidentiality inside the enclave but assume code is trusted!
- In the PDMS context, the Data tasks are untrusted ➜ require another level of isolation
- Enforcing reverse isolation requires additional in-enclave sandboxing (e.g., Ryoan [HZX18]) ➜ Stateless data tasks, i.e., no access to persistent storage and one-shot at input data

Joint (asymmetrical) data management between Core and Data tasks ➜ require redesigning traditional data access methods

- Core can only support basic, generic data management operations (e.g., in the spirit of a KVS) and has access to storage
- Data tasks implement advanced data processing techniques but require to map them to the Core API for persistence
Satisfying \textit{mutually trusted collective computations} property

\textbf{Assumptions}

- Distributed computation code is arbitrary (and untrusted), with access to $n$ PDMS raw data
- The number of participants can be huge
- Confidentiality of a (small) subset of PDMS can be broken by their owners (colluding)

\textbf{Requirements}

- Computation code only accesses needed raw data, only the result is shared, it is attested
  - \textit{Global Manifest}: collection rules + computation code + 3\textsuperscript{rd} party accessing the result
  - \textit{Local validity}: each participant checks \textit{locally} that all others behave honestly
  - \textit{Global integrity}: certify through attestation the output of each data task for incremental integrity checks
  - \textit{Side-channel attacks resiliency}: (1) circumscribe leakage and (2) prevent targeting

Building blocks from [LAP+19, LSB19a]
Satisfying *mutually trusted collective computations* property

**Assumptions**

- Distributed computation code is arbitrary (and untrusted), with access to $n$ PDMS raw data.
- The number of participants can be huge.
- Confidentiality (🔒) of a (small) subset of PDMS can be broken by their owners (colluding).

**Requirements**

- Computation code only accesses needed raw data, only the result is shared, it is attested.
  - Global Manifest: collection rules + computation code + 3rd party accessing the result.
  - Local validity: each participant checks locally that all others behave honestly.
  - Global integrity: certify through attestation the output of each data task for incremental integrity checks.
  - Side-channel attacks resiliency: (1) circumscribe leakage and (2) prevent targeting.

Building blocks from [LAP+19, LSB19a]

Reusuable building blocks to achieve integrity and confidentiality from multiple TEEs, e.g., VC3, M2R...
- address some limitations: single vs. multiple controllers.

Protect data provider nodes from computation code providers (and vice-versa), e.g., based on Ryoan [HZX18].
Remaining Challenges: Distributed data management level

Guarantees against confidentiality attacks from covert (colluding) adversaries in large PDMS networks*

Which network overlay to allow for efficient, scalable and secure node selection and distributed data indexing?

How to efficiently prevent a (large colluding) attacker to influence a query execution?

How to efficiently execute a query while minimizing the private data leakage risks?

*DISPERS: Securing Highly Distributed Queries on Personal Data Management Systems [LSB19b]

Leakage through communication patterns

Making the dataflow of a distributed computation data independent is costly (broadcasts…)

Find appropriate, efficient solutions to anonymize the communications between peers (e.g., in-network proxies, differentially private data exchanges, mixer nodes, …)

Confidentiality guarantees on the final result in a fully distributed setting

Make sure that the final result is ‘sufficiently’ anonymous (e.g., the selected set of participants is sufficiently large and divers)

Integrate privacy guarantees as part of the manifest (e.g., to let the nodes decide if the exposure risk is compliant with the user required privacy level)
Other remaining challenges: Administration level

Make the net effects of sharing policy viewable and easy to adjust

- How to integrate (faithful) visualization tools in Data tasks with peripheral isolation?
- How to validate net effects of decisions when the set of permissions is huge (e.g., [LWG+13])?
- How to identify suspicious grants (first step [TAP17])?
- Visualize the complete life-cycle of personal data while it may undergo transformations?
- How to visualize time or location-based contextual rules?

Design a trusted reference monitor to enforce ‘effects’ rather than ‘decisions’

- Define a model to capture the effects of policies in a simple/basic logic
- Visualization tools need to be executed as Data tasks providing peripherals isolation
- Materialize / maintain these effects in an efficient way

Design ‘zero-knowledge’ grants

- Model side-channels leaking personal data within (acceptable) sets of permission
- Design countermeasure, e.g., based on replayed enclaves on a “what if” basis
Which possible concrete PDMS instances?

(a) Home box
- Home box with TrustZone
- TrustZone
- Secure area
- Secure Element
- User smartphone with TZ

(b) Mobile device
- Smartphone/Tablet with TrustZone and Secure Element

(c) Cloud based
- Cloud Server with SGX
- SGX

Logical architecture:
- Data collector
- Personal comp°
- Collect. Comp°
- Core modules
- Decision making
- Untrusted apps

Decision making
- Personal comp°
- Collect. Comp°
- Core modules
- Decision making
- Untrusted apps

Core isolation
- Attestation
- Confidentiality
- Peripheral isolation
- TrustZone processor
- Secure Element
- Intel SGX

Smartphone/Tablet with TrustZone and Secure Element

- Secure area
- Rich area
- TrustZone
- SE

Core (proven code)
- Isolated data task
- Untrusted module/App
- Protected database
Conclusion

Personal Data Management Systems arrive at a rapid pace and provide us with novel tools to manage our own personal data. But current existing solutions lead to various and irreconcilable choices.

There is a stringent need for an Extensible & Secure PDMS (ES-PDMS).

I.e., provide at least the minimum set of functionalities to cover the complete data lifecycle with trustworthy security guarantees.

TEEs offer a great opportunity to reach the ES-PDMS goal. But we need to look more closely at the specificities of the PDMS context.

A first major step is to arrive at a reference logical architecture based on TEE. A way out of the dilemma between the delegation of personal data to providers and the myth of the owner’s self-capacity to secure her own data.

But the journey has only begun and many challenges remain to be addressed!

… and the most critical ones primarily concern our research community.
Conclusion

Personal Data Management Systems arrive at a rapid pace and provide us with novel tools to manage our own personal data
But current existing solution lead to various & irreconcilable choices

There is stringent need for an Extensible & Secure PDMS (ES-PDMS)
I.e., provide at least the minimum set of functionalities to cover the complete data life-cycle with trustworthy security guarantees

TEEs offer a great opportunity to reach the ES-PDMS goal
But we need to look more closely at the specificities of the PDMS context

A first major step is to arrive at a reference logical architecture based on TEE
A way out of the dilemma between the delegation of personal data to providers
… and the myth of the owner’s self-capacity to secure her own data

But the journey has only began and many challenges remain to be addressed!

… and the most critical ones primarily concern our research community
Conclusion

Personal Data Management Systems arrive at a rapid pace and provide us with novel tools to manage our own personal data. But current existing solution lead to various & irreconcilable choices.

There is stringent need for an Extensible & Secure PDMS (ES-PDMS) to provide at least the minimum set of functionalities to cover the complete data lifecycle with trustworthy security guarantees.

TEE offers a great opportunity to reach the ES-PDMS goal. But we need to reflect more closely at the specificities of the PDMS context. A first major step is to arrive at a reference logical architecture based on TEE. A way out of the dilemma between the delegation of personal data to providers and the myth of the owner’s self-capacity to secure her own data.

But the journey has only began and many challenges remain to be addressed!

And the most critical ones primarily concern our research community.
Thanks!

Questions?
References (1)


References (2)


References (3)


References (4)


DomYcile home box (based on PlugDB)

--- D District ---
8,000 households
262 counties
10,000 patients

Usage 1
Effectiveness control

Usage 2
Medical / social care coordination

Usage 3
Supervision

Sigfox / GPRS
**Expected PDMS functionalities & properties:**

**Storage & Recovery**

**Corporate DBMS**
- Managed by experts (DBA/DSA)
  - DBA/DSA have granted access to all data/keys
  - Use of HW Security Module to store keys
- Classical Backup & Recovery

**Individual’s PDMS**
- Managed by non-experts (PDMS owners)
  - PDMS may host other user’s data
  - → no granted access to full PDMS content
  - Master key may be lost

---

**Property 2: A PDMS enforces** *mutual data at rest protection* **iff:**

1. the PDMS protects data & backup archive in confidentiality and integrity;
2. the secret protecting the backup archive is recoverable;
3. the secret is only accessible to a PDMS of the owner, providing all security properties.

‘*mutual*’: PDMS stores raw data from others → protection also operates against its owner
The PDMS enforces these properties automatically → no administrator attacks
**Expected PDMS functionalities & properties:**

### Administration and data dissemination

<table>
<thead>
<tr>
<th>Corporate DBMS</th>
<th>Individual’s PDMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managed by experts (DBA/DSA)</td>
<td>Non-expert owner, highly dynamic setting, untrusted environment</td>
</tr>
<tr>
<td>Rely on rich access control policies</td>
<td>Single owner interacts with myriads of 3rd parties</td>
</tr>
<tr>
<td>… like RBAC, MAC, TBAC, etc.</td>
<td>… cannot apprehend the potential net effects</td>
</tr>
<tr>
<td>Standard security measures</td>
<td>… and administration is performed from a potentially untrusted devices by a non-expert…</td>
</tr>
<tr>
<td>… and audit processes to enforce it</td>
<td></td>
</tr>
</tbody>
</table>

**Property 5: A PDMS enforced controlled data dissemination iff:**

1. integrity & confidentiality of interactions between the PDMS and its owner are guaranteed, when decisions regarding data dissemination are made;
2. the decisions are enforced by the PDMS and cannot be circumvented.

This property ensures that all decisions are faithfully captured (point 1) … and that the effects of these decisions are enforced (point 2)

Audit (point 1) is provided to help lay owners to understand all the effects of their decisions