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: 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 ➞ 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?

Disclaimer: work done by the tutorial presenters
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)

Disclaimer: work done by the tutorial presenters
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)
Building blocks from [BPS+16, BBB+17]

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 + 3\textsuperscript{rd} party accessing the result
- Data task runs computation code (\[\text{\scriptsize{\text{\_}}},\text{\scriptsize{\_}}\]) + result declassification by the Core

---

**Diagram:**

- CORE
  - Trusted Computing Base
  - Audit data
  - Energy bill
  - Power meter measurements
  - TCP/IP DNS

- Energy supplier
  - Core (proven code)
  - Isolated data task
  - Untrusted module/app
  - Protected databases
  - Code isolation
  - Attestation

- Personal computation data task
  - Apps
  - Internet

- No access!

- Other data task
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

Control several data tasks used simultaneously in a computation, e.g., such as in πBox[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 *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 + 3\textsuperscript{rd} 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]
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]
- Reusable 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
- Secure area
- Secure Element
- User smartphone with TZ

(b) Mobile device
- Smartphone / Tablet with TrustZone and Secure Element
- Core modules
- Decision making
- Personal comp°
- Collect. Comp°
- Data collector
- Untrusted apps

(c) Cloud based
- Cloud Server with SGX
- SGX
- Secure area
- Rich area
- User smartphone with TZ

- Code isolation
- Attestation
- Confidentiality
- Peripheral isolation
- TrustZone processor
- Secure Element
- Intel SGX
- 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 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 solutions lead to various and irreconcilable choices.

There is a stringent need for an Extensible and 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 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 solutions lead to various & irreconcilable choices.

There is a stringent need for an Extensible & Secure Personal Data Management System (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 specifics of the PDMS context.

A first step is to arrive at a reference logical architecture based on TEEs, providing 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!

Let’s make it happen!
Thanks!

Questions?
References (1)


References (2)


References (3)


References (4)


