Méthodes formelles
et sécurité
16 march 2021

Guillaume Scerri

PETRUS team, Inria & UVSQ
https://team.inria.fr/petrus/

Decentralized personal data management: Challenges for formal methods

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…
- **Anonymized datasets often not anonymous**
  - Eg.: 15 fields is enough [RHM19]

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…

Anonymized datasets often not anonymous

Eg.: 15 fields is enough to deanonymize [RHM19]

Uses considered questionable

Eg.: Social medias (Visa, Insurance)

Personal reports (Pipl, Intelius…),

…


THE WALL STREET JOURNAL.
New York Insurers Can Evaluate Your Social Media Use—If They Can Prove Why It's Needed

Discriminatory uses of personal data

Eg.: criterias in targeted ads,

e-justice, recruiting process

23andMe vs. GINA, …

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

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

The New York Times
San Francisco Bans Facial Recognition Technology

More acceptance

Symptoms of a crisis of consciousness (e.g., Time well spent)
From "**privacy is no longer the social norms**" (2019)
… to "**private is the future**" (2019)
Privacy-based marketing campaigns

Pop-up Cafés in UK

Chicago

New York

Toronto

Los Angeles

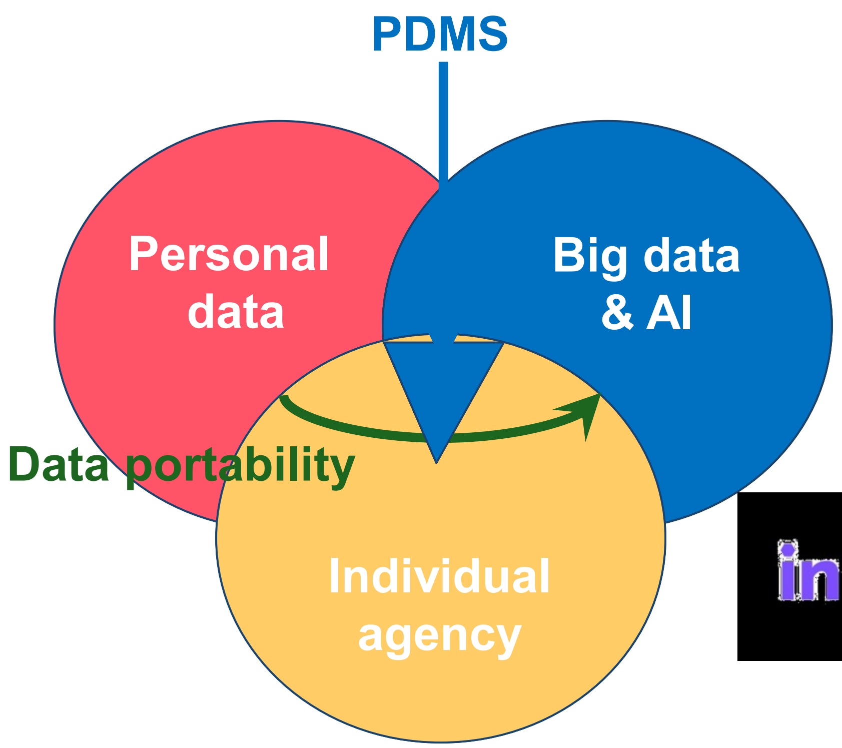
Current trend: give their personal data (agency) back to individuals

Act I: the right to Data portability

… the right to retrieve its own data

Act II: Personal Data Mg^{t} Systems (PDMS)

… the tool to manage its own data



©Tim Berners-Lee

Is this enough to change the situation? …

TWO prerequisites !

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

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 Executº Env^{t} (high-end servers & edges)

Outline

I. Functionalities and properties of a PDMS

Review of functionalities & assumptions

Informal properties and challenges for formalization

II. Architecture, techniques and threat model

The promises of Trusted Execution Environments (TEEs)

A review of privacy-preserving data management using TEEs

Main classes of architectures for a PDMS

Online personal cloud

→ Advanced functionality, strong trust assumptions

- **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, code transparency (community checks), PEN tests (Cozy)

cozy.io

No-knowledge personal cloud

→ Increased security, minimalist functionality

- **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

➔ ‘formal’ security is lost, more functionality

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)

Functionality: data store and backup in a dedicated hardware plug

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

➔ Security at the price of functionality, advanced processing on untrusted devic

Tamper-resistant home cloud

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

Trust model: secure HW + embedded SW are trusted

Unifying properties?

Objective for a PDMS

(1) Provide the main of set of functionalities (personal data life-cycle):

Data collection, storage and recovery, personal (cross-)computations, collective computations, and data dissemination.

(2) Address the threats identified:

Data snooping, Data leakage, Secondary usages, Over-priv. Apps, Failures, Ransomware, …

Is there anything new?

Specificities of (individual's) PDMS?

Derive properties towards extensive and secure PDMSs?

5 main functionalities ➔ 5 main properties [ABB+19]

Very far from classical DBMS: distributed, no expert in the loop, …

Expected PDMS functionalities & properties: <u>Storage & Recovery</u>

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: 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

Challenges: Fairly close to usual protocol properties but

- Inhenrently stateful
- Might involve complex primitives

Expected PDMS functionalities & properties:

Administration and data dissemination

Individual's PDMS

Non-expert owner, highly dynamic setting, untrusted environment

Single owner interacts with myriads of 3rd parties

… cannot apprehend the potential net effects

… and administration is performed from a untrusted devices by a non-expert…

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

Challenges:

- Modelling non expert user (can't expect user to write AC rules…)
- Modelling partial trust in devices

Expected PDMS functionalities & properties: <u>Data Collection</u>

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

Challenges:

- Modelling trust for untrusted code
- Modelling code composition

Expected PDMS functions & properties: Personal computations

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)

Challenges:

- Model guarantees for partially trusted code
- Model attestations

Expected PDMS functions & properties: Collective computations

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 MPC)

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

Challenges:

- Find weaker models than secure multiparty computations
- Deal with integrity of complex computations

The goal:

An Extensive & Secure PDMS

provides the expected set of functionalities
to cover the complete data life-cycle
- *data collection,*
- *storage and recovery,*
- *personal 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.*

Challenges: formalize complex definitions involving
- Imprecise human behavior
- Untrusted/partially trusted code
- Statefulness

How do we build a PDMS?

The field of TEE-based secure data management is rapidly developing

Outline

I. Functionalities and properties of a PDMS

- Review of functionalities & assumptions
- Informal properties and challenges for formalization

II. Architecture, techniques and threat model

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

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° Environm^{t} (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:

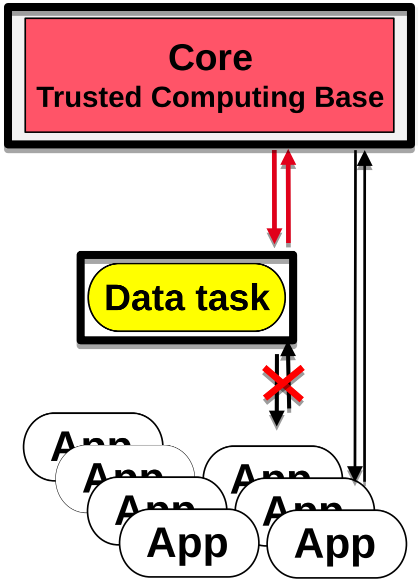
- **Side channels ➔ threat model of recent TEEs**
  - Execution time (by OS/colocated programs)
  - …. memory accesses at page level (OS), byte level (memory bus)
  - → **Won't be fixed : need to be addressed in solutions**
- **Attacks based on speculative execution ➔ leak secrets (secret keys of enclaves)**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

Not a magic bullet that allows to execute everything safely

Logical Architecture : Three Layers [ABB+19]

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)



Satisfying *bilaterally trusted data computations* property

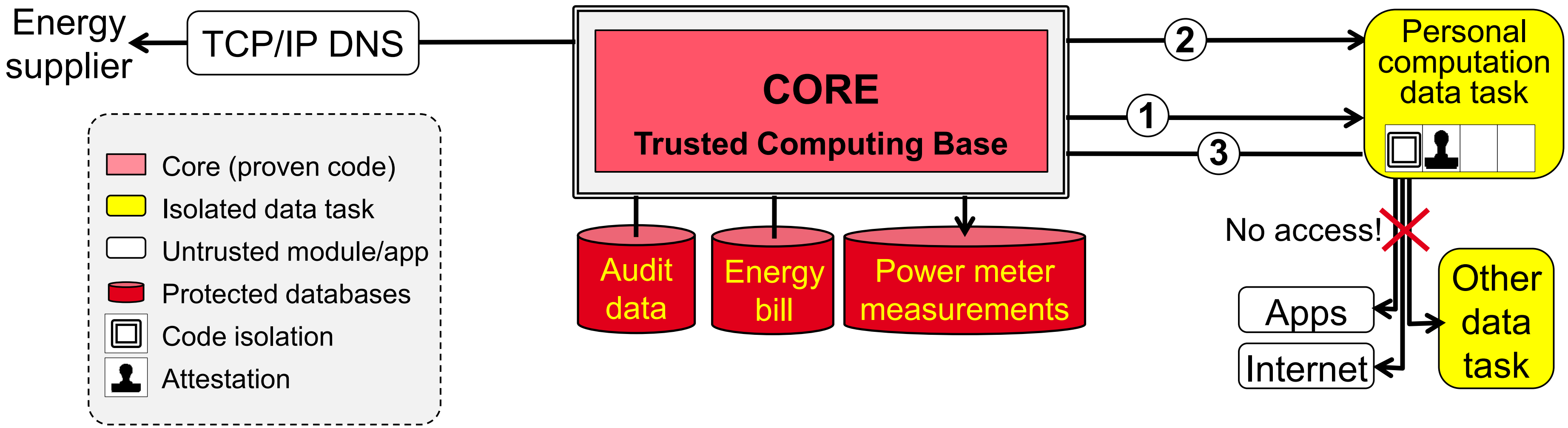
Assumptions

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**



Challenges for formal methods

- **Large number of moving parts (even for just one property)**
  - **Composition (of stateful processes, with partially trusted code)**
    → Not easy (shared secrets, shared data, …)
- **Core not that simple**
  - **Prove code of essentially a full DBMS**
- **Model properties of TEEs**
  - **In particular how to execute untrusted code in a safe way?**

A quick zoom on executing code in TEEs

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 average age of people who voted for Macron
3. If record retrieved in 1 is also retrieved in 2, Alice voted for Macron

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 existing DBMS adding an ORAM layer for memory accesses

Problem: each memory access costs $O(\log^2(|DB|)$ – in practice ~x50

What about memory access within enclave? Oblix [MPC+18]

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 (Oblix): 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?

Conclusion: challenges

Formally establish properties of a PDMS

- Weak models of humans
- Stateful processes
- Guarantees on partially trusted/untrusted code

Design a minimal and proven Core engine

- Minimal (in code size & complexity) proven 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?

Build protocols and proofs around that Core

- Need better composition results
- Need good model of untrusted code executed in enclaves
- Complex primitives (e.g. ORAM)

Thanks !

Questions ?