TRAPEZE

Using DPVCG's outcome in TRAPEZE's compliance framework

Piero Bonatti[1], Luigi Sauro[1], Jonathan Langens[2]

[1]University of Naples Federico II

[2]Tenforce

DPVCG meeting 15.09.2021

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 883464.

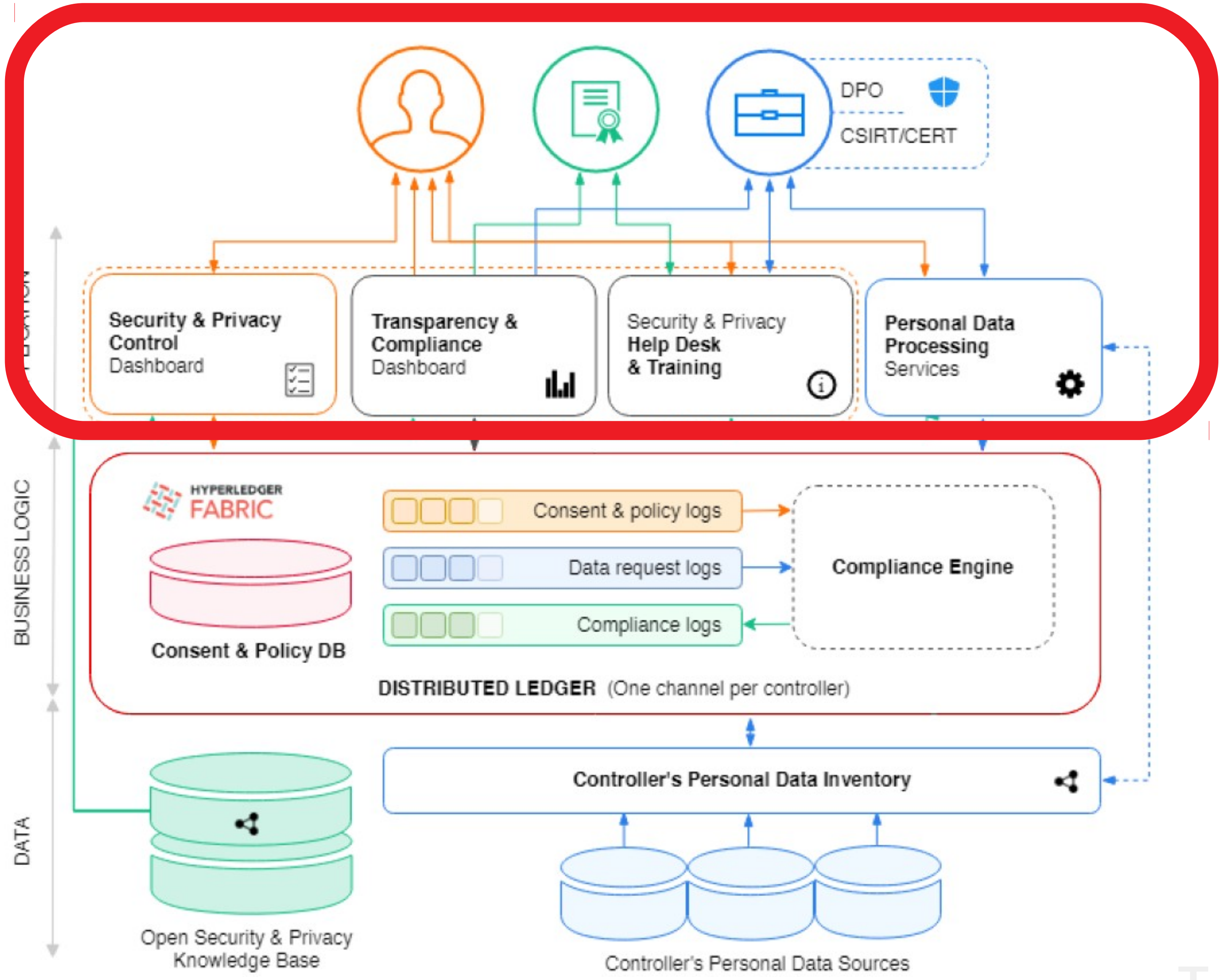
TRAPEZE

The European H2020 project TRAPEZE aims at the protection of the European data economy by weaving trust into its very foundation. The pillars:

- Control (on data processing, by both data subjects and controllers/data processors)
- Transparency (for citizens and DPA, through records of processing activities and intelligent dashboards)
- Compliance (automated, ex ante, as in access control, and ex post, for auditing)

Means to this end: technical and methodological, citizen-first, innovations

TRAPEZE's architecture



DPO
CSIRT/CERT
Security & Privacy Control Dashboard
Transparency & Compliance Dashboard
Security & Privacy Help Desk & Training
Personal Data Processing Services
BUSINESS LOGIC
HYPERLEDGER FABRIC
Consent & policy logs
Data request logs
Compliance logs
Compliance Engine
Consent & Policy DB
DISTRIBUTED LEDGER (One channel per controller)
Controller's Personal Data Inventory
DATA
Open Security & Privacy Knowledge Base
Controller's Personal Data Sources

Use cases related to compliance

- Compliance of business processes:
  - Is the data usage policy of the process compliant with the GDPR?
  - Are the operations of business processes compliant with the available consent?
  - Consent re-use
- User agents:
  - Is the controller’s policy compatible with the data subjects’ privacy preferences?
  - What are the consequences of consent? (what-if queries)
- Data protection authorities:
  - Are the operations of the controller compliant with regulations and consent?

Compliance of business processes

- **Is a privacy policy compliant with the GDPR?**
  - Different from access control: *All* the operations authorized by the policy should be compliant (as opposed to single operation requests) [hereafter: *policy containment*] [query answering vs containment]
  - A *uniform* way to answer in one shot questions like: Are all mandatory obligations included in the policy? Are the necessary technical measures included in the policy? Is the legal basis appropriate to the category of processed data? Do international data transfers comply with GDPR's restrictions? …
- **Compliance with the available consent and consent re-use**:
  - Which data items can be processed by a business line? Can data still be processed after the data subject updates her consent? If the business line is *new*, which data can be directly fed to it and which require new consent?
  - Again, the question is: Do *all* the operations that the business line *may* execute comply with consent? [*policy containment*] **Note:** Late denials may cause *unnecessary* partial processing (unlawful)

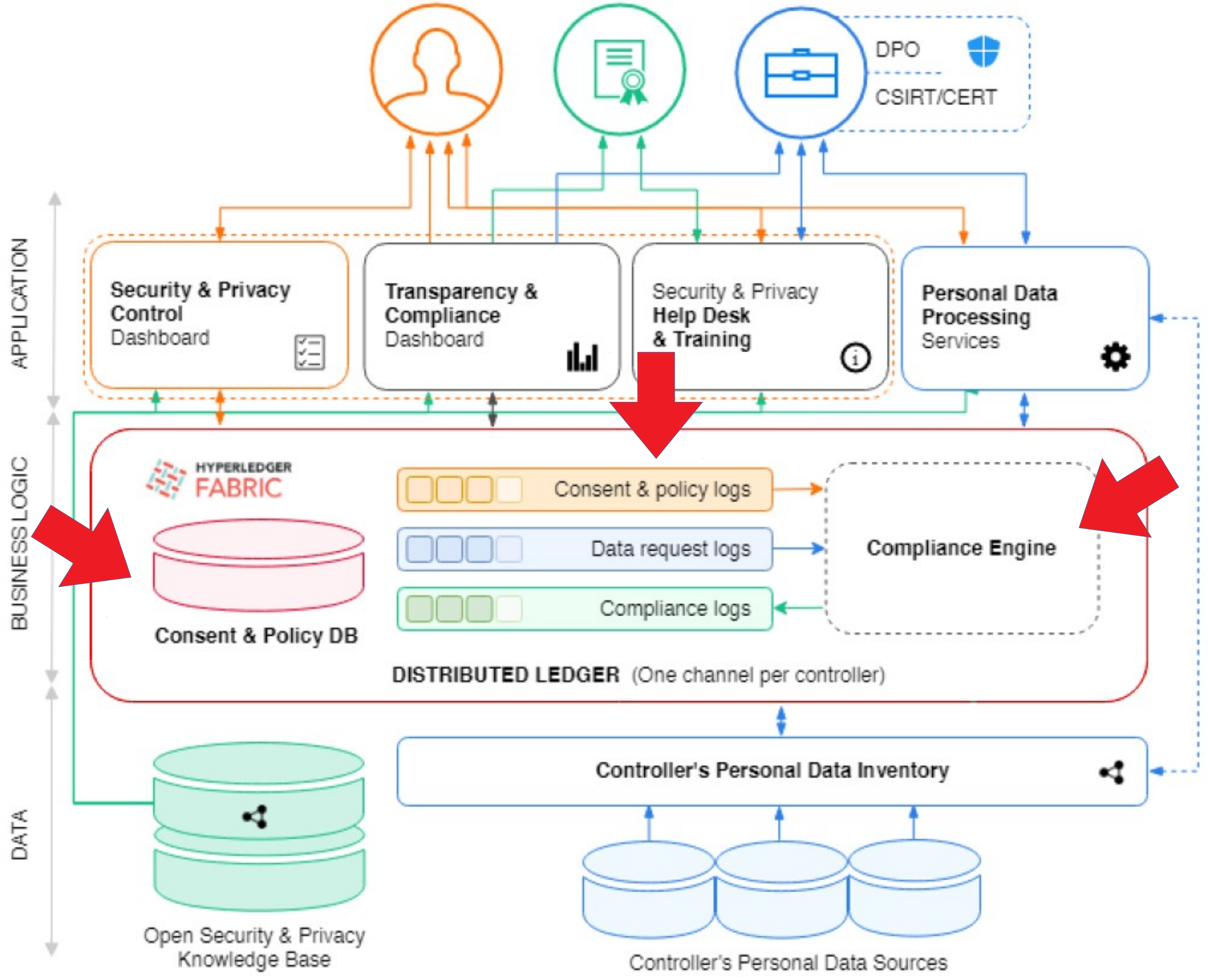
User agents

- **Is a privacy policy compliant with the data subject’s privacy preferences?**
    - E.g. “*analytics can be carried out (only) on anonymized data*”. Does the privacy policy allow operations that violate this constraint? [again: *policy containment*]
    - Opportunities for automated opt in/out

DPA auditing

- Data Protection Authorities are interested both in
  - Compliance of the controller’s processes with the GDPR
  - Compliance of the controller’s processes with data subjects’ consent
  - [*policy containment*]

TRAPEZE's architecture



DPO
CSIRT/CERT
APPLICATION
BUSINESS LOGIC
DATA
Security & Privacy Control Dashboard
Transparency & Compliance Dashboard
Security & Privacy Help Desk & Training
Personal Data Processing Services
HYPERLEDGER FABRIC
Consent & Policy DB
Consent & policy logs
Data request logs
Compliance logs
Compliance Engine
DISTRIBUTED LEDGER (One channel per controller)
Controller's Personal Data Inventory
Open Security & Privacy Knowledge Base
Controller's Personal Data Sources

Data Usage Policies: a unified view

- *Data usage policies* include:
  - the privacy policies of the *data controllers* (those who collect and process data)
  - the consent of *data subjects* and their privacy preferences
  - the objective part of data protection regulations such as the GDPR, that are high-level data usage policies
- Conceptually, data usage policies are *classes* of possible operations, described by properties such as:
  - Purpose, data categories, processing, storage location & duration, associated obligations, legal bases, …

Requirements on data usage policies & related languages and algorithms

- An *expressive language* for encoding GDPR concepts, privacy policies, and consent
  - In a way that enables *automated compliance checking*
    - Privacy policies vs GDPR
    - Privacy policies vs consent
- *Usability*: For policy authors (controllers' employees and data subjects)
- *Reliability*: Strong guarantees on the correctness of compliance checking (violations are expensive)
- *Interoperability*: different stakeholders, *sticky policies*
  - Not only standard formats. Everyone must understand the policies in the same way
- *Scalability*: There exist challenging real-time scenarios (e.g. telco providers)

Main challenge

- Satisfying all requirements together
  - Expressiveness vs scalability
  - Expressiveness vs usability

- The next slides illustrate how TRAPEZE addresses the requirements

W3C DPVCG Vocabularies

DPVCG = Data Privacy Vocabulary Community Group of the W3C

With DPVCG’s vocabularies we address

- *Expressiveness*
  - Extensive coverage of GDPR’s concepts (data categories, purposes, legal bases, obligations, technical meaures, …)
  - Plus common sub-terms that occur in many application domains (e.g. commonly used data categories and purposes)
- *Interoperability*
  - DPVCG provides a standardized representation of the above concepts

Example 0: DPV in OWL2

Axiomatizing DPVCG's vocabularies is easy. For each term:

1. Declare its type

```
Declaration( Class( CreditCapacity ) )
Declaration( ObjectProperty( hasPurpose ) )
```

2. Declare its logical properties

```
SubClassOf( CreditCapacity  Credit )
ObjectPropertyRange( hasPurpose Purpose )
DisjointClasses( Purpose Processing Recipient … )
…
```

Example 0: DPV in OWL2 (II)

Leveraging OWL2 *annotations*:

Declaration(<*annotation*> <*term declaration*>)

Annotations are human readable text, that can be used to:

- Describe the term’s semantics, as clearly as prescribed by GDPR’s requirements on informed consent
- Double check that the formalization reflects the intended meaning of terms
- Translate the formal OWL2 policy into natural language

OWL2 PL Profile (I)

PL = Policy Language. Profile = restricted use of OWL2 operators

- *Expressiveness*
  - OWL2 PL can assemble DPVCG’s terms into privacy policies, consent, and GDPR requirements
    - In a machine-understandable way
  - OWL2 is *class-oriented* which is paramount for:
    - Expressing policies with different granularity
      - This enables compliance checking of privacy policies (finer-grained), w.r.t. the GDPR (high-level)
      - Fine tuning of policies, e.g. *you can track my position in Europe/Italy/Naples/My university/My office* …
      - Data re-use for similar purposes, e.g. *all kinds of CommercialInterest*
    - Capturing compliance checking [policy inclusion] correctly
    - Instance-oriented languages like RDF(S) not adequate
      - Their semantics has no relationship with compliance checking → wrong results + no granularity handling

OWL2 PL Profile (II)

- *Interoperability*
  - OWL2 is a standard
  - Through OWL2 ontologies the meaning of new or local domain-specific terms can be communicated to other parties – *no need for changing the compliance checker's code*
    - Also good for *extensibility*
  - Ontologies + OWL2's formal, *unambiguous* semantics → policies are understood in the same way by all peers/software agents
    - Crucial for compliance

Example 1: adding domain specific terms

In order to add a new domain-specific purpose *RecommendArtEvents* it suffices to add 2 lines to the core ontology for DPVCG's vocabularies:

```
Declaration( Class( RecommendArtEvents ) )

SubClassOf( RecommendArtEvents ServiceProvision )
```

Here *ServiceProvision* is the term of DPVCG's vocabulary of purposes that is more closely related to *RecommendArtEvents*.

The above two lines let the reasoner (compliance cheker) conclude that consent to use data for *ServiceProvision* purposes (or any superclass thereof) allows also the recommendation of art events.

OWL2 PL Profile (III)

Reliability

- *Correctness*: No false positives: a data usage policy P actually complies with restriction R (either consent or GDPR) if the compliance checking algorithm says so
  - Mitigates the risk of violations/sanctions/loss of reputation
- *Completeness*: No false negatives: if the compliance checking algorithm says that P does not comply with R then this is actually the case
  - No unnecessary loss of data usage opportunities
- OWL2 addresses also this requirement:
  - *Formal semantics* → correctness and completeness criteria for the compliance algorithm
  - (RDF(S) not adequate for this purpose)

OWL2 PL Profile (IV)

Scalability

- Reasoning in OWL2 PL is much easier than full OWL2 reasoning (polynomial time)
- We use a *specialized* reasoning algorithm for compliance checking
  - Correct and complete for OWL2 *PL*
- Performance of a sequential Java implementation:
  - Most common policies: between 410 and 570 μsec per compliance check

OWL2 PL Profile (V)

Usability / Adoption

- The *specialized* algorithm for compliance checking is *simple*
- It can be implemented in many languages by adopters (including our partners)
  - So far: Java, Ethereum’s smart contracts, Javascript
- No dependencies on complicated, possibly proprietary, black-box technology

External policy formats

Usability / Adoption

- Several companies/organizations are already familiar with JSON, ODRL, …
- Strategy: *Encode policies with external formats (JSON, ODRL, …) and convert them to the internal OWL2 format for compliance checking and explanations*
- The usability of a streamlined ODRL profile has been successfully tested by a partner of previous project SPECIAL
- In TRAPEZE we have defined a JSON encoding, JSON PL, with unambiguous translation into OWL2 PL
  - Vocabulary neutral, unlike ODRL; therefore compatible with DPVCG and other proposals
  - Natively supported by Ember.js, Angular, React, Vue.js and any framework written in Javascript
  - Very familiar to developers

Example 2: usage policy in JSON (I)

The set of policies of a controller or a data subject is defined with a *policy set*

```
{   “@policySet” :  [ policy 1, policy 2, …],
    “@ontologies” : [   DPV ontology,   domain specific ontology, … ],
    “@context” :  [
            “@base”: “default namespace”,
            “prefix 1” : “namespace 1”,
             …  ]
}
```

Example 2: privacy policy in JSON (II)

Sample policy: *For marketing purposes, demographic and location data are/can be given to ACME and processed by an automated decision making system with human involvement*

```
{
    "hasDataCategories": ["Demographic", “Location”],
    "hasPurpose": "Marketing",
    "hasRecipient": "acme:ACME",
    "hasProcessing": { "@class": "AutomatedDecisionMaking",  "hasHumanInvolvement": "true" }
    …  // if this is a privacy policy add legal bases and obligations, protection measures, info about storage…
}
```

Example 2: privacy policy in JSON (III)

Where do the DPV’s fit in: properties

```
{
    "hasPersonalDataCategories": ["Demographic", “Location”],
    "hasPurpose": "Marketing",
    "hasRecipient": "acme:ACME",
    "hasProcessing": { "@class": "AutomatedDecisionMaking",  "hasHumanInvolvement": "true" }
    …
}
```

Example 2: privacy policy in JSON (III)

Where do the DPV’s fit in: classes

```
{
    "hasPersonalDataCategories": ["Demographic", “Location”],
    "hasPurpose": "Marketing",
    "hasRecipient": "acme:ACME",
    "hasProcessing": { "@class": "AutomatedDecisionMaking",  "hasHumanInvolvement": "true" }
    …
}
```

Example 2: privacy policy in JSON (III)

Getting the expressiveness of RDF(S): instances

```
{
    "hasPersonalDataCategories": ["Demographic", “Location”],
    "hasPurpose": "Marketing",
    "hasRecipient": "acme:ACME",
    "hasProcessing": { "@class": "AutomatedDecisionMaking",  "hasHumanInvolvement": "true" }
    …
}
```

where ACME is defined in the acme ontology as an instance:

```
    Declaration( NamedIndividual( ACME ) )
```

Other uses of the usage control language & DPV

(not strictly related to automated compliance)

- *Record of processing*: A mandatory document summarizing all personal data processing
  - Same as the machine understandable privacy policy – alignment is guaranteed
- *Consent request generation and privacy policy verbalization*: Turning the machine understandable privacy policy into natural language
  - Leveraging annotations (already exploited in SPECIAL for data transfers across borders)
  - See also the GUIs developed by TU Berlin (that use PL as internal format)

Other uses of the usage control language & DPV (II)

(related to automated compliance)

- *Personal data processing logs*: entries are described with the usage control language
  - Additional measures (e.g. blockchains) to make logs tamper-proof
- *Labeling software/processes and resources*: (with purposes, contents, etc.)
  - In order to derive automatically the description of each operation on personal data
  - Which fosters the alignment of privacy policies and system behavior

Ongoing work on OWL2 PL

Negation/exceptions (related to dynamic consent)

- In dynamic consent, the user opts-in for a specific use of her data, then she introduces exceptions
  - Either on the fly or ex post, through the privacy dashboard
  - e.g. "*You can track my location*" followed by "*don't track my location in Rome*"
- Exceptions make sense in general to geto more compact policies
  - e.g. "*CommercialInterest* is ok with the exception of *SellDataToThirdParties*"
- Compliance checking with negation in general is coNP-hard
- We introduced a restricted use based on histories of the form P1±P2±...±Pn
  - For example P1-P2 where P1="*You can track my location*" and P2="*you can track my location in Rome*"
  - P2 acts as a *prohibition*

Ongoing work on OWL2 PL (II)

Negation/exceptions (continued)

- It is important to know whether the current operation overlaps any prohibitions
- This requires a careful axiomatization of disjoint classes
  - e.g. “*SellDataToThirdParties*” and “*SellProductsToDataSubject*”
  - So if the former is prohibited, the latter is still allowed because it does not overlap
- DisjointClasses axioms are also very useful to debug policies
- They highlight contradictions, such as misplaced classes like
  - “hasPersonalData”: “SellDataToThirdParties”
- Default disjointness does not always work, eg:
  - “hasPersonalData”: {@intersection: [Anonymized, Demographic] }
  - “hasPersonalData”: {@intersection: [Blurred, Location] }
- We encourage the DPVCG to specify which classes are disjoint

For further information please contact me at

piero.bonatti@unina.it

or visit

https://trapeze-project.eu/