Log Design for Accountability

Denis Butin, Marcos Chicote and Daniel Le Métayer

Context

Data subjects share more & more PII

Stronger privacy guarantees, more transparency needed

A common strategy to support privacy: Privacy Impact Assessments

- Modern analytic approach to mitigate privacy risks
- Done before deployment
- No guarantees to users about actual running system

Motivations for a complementary approach

- Runtime / a posteriori verifications needed!
- "Proven trust" instead of "blind trust"
- Data controllers should be accountable to data subjects
- Practical requirements?

Motivations for a complementary approach

- Need to provide the means to check that policies were complied with
- Method: check PII handling event logs against policies, automatically
- Duality — if PIA done right (*implies* design choices), accountability possible (*depends* on design)

What is meant by *accountability*?

- Obligation to accept responsibility for actions
- Attributability: who did what?
- Non-repudiable evidence that cannot be falsified
- Transparent use of information
- Article 29 Working Party Opinion: *showing how responsibility is exercised, and making this verifiable.*

Enabling accountability in practice

- Accountability does not emerge spontaneously
- Feasibility of comprehensive a posteriori verification?
- Depends directly on technical architecture!

Ingredients for practical accountability

Need to define:

- Obligations to be met $\Longrightarrow$ Policy language
- Compliance checking evidence $\Longrightarrow$ Log architecture
- Compliance checking procedure $\Longrightarrow$ Log analyser

Policy languages

- Lengthy text-formatted privacy policies seldom read by data subjects. . .
- . . . Usage policy languages allow data handling details to be standardised, set and matched
- On both sides: data subject (preferences), data controller (policies).
- Examples: P3P, EPAL, PPL

Primelife Policy Language (PPL)

- Access and data usage policy language, developed by SAP (European project PrimeLife)
- Extends XACML with usage control features ; uses SAML protocol language
- Symmetric architecture (data subject side / data controller side) yields *Sticky Policies* (agreements)

Primelife Policy Language (PPL)

- Automated matching of
  - Data subject (Data Handling Preferences) &
  - Data controller (Data Handling Policies)
- Wide range of obligations possible (trigger + action)
- Authorizations
  - Use for a specific purpose
  - Downstream (third party) usage

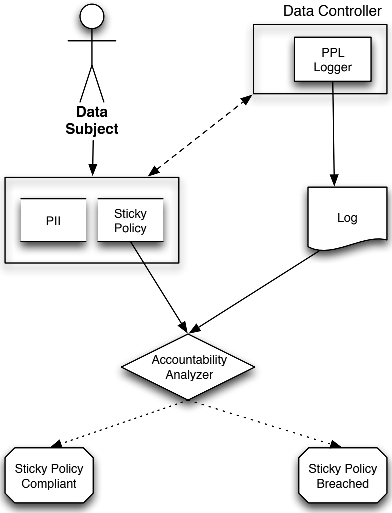
Primelife Policy Language (PPL)

- Trigger examples: At time / Periodic / On PII deletion / On PII access for purpose ...
- Action examples: Delete PII / Encrypt PII / Notify data subject / log ... (usually before a set deadline)
- Only informal specification available until our work

PII event logging

- Data controller must provide evidence that agreements met
- Audit possible through inspection of a log against the corresponding sticky policy
- Structure of logs conditions auditability, hence accountability
- Deciding what to include in logs — not a trivial task (tension with minimisation needs)

Architectural overview



Contribution: PPL formalisation / PPL log analyser

- Relevant events precisely defined (syntax) / ambiguities identified
- Compliance properties described (semantics)
- Tool built for automated compliance checking — Haskell implementation
- Policy matching supported
- Reasoning over compliance can be generalised

Log design guidelines

- Importance of explicitness — avoid ambiguity by reflecting causal relationships
- Accountability definitions shape log structure & vice versa
- Support break-glass situations (exceptional / emergency usage)
- Provide links between formal specifications and human verification

Conclusion

- Log architecture must be considered from the design phase on
- Suitable log structure supports privacy through accountability
- General, policy language independent principles can be derived
- Future work: formal framework for verification of accountability architectures (formal methods): characterise trusted policy engine components