Pastis, a Model & System for Data Access on the Web

Alban Galland[1]

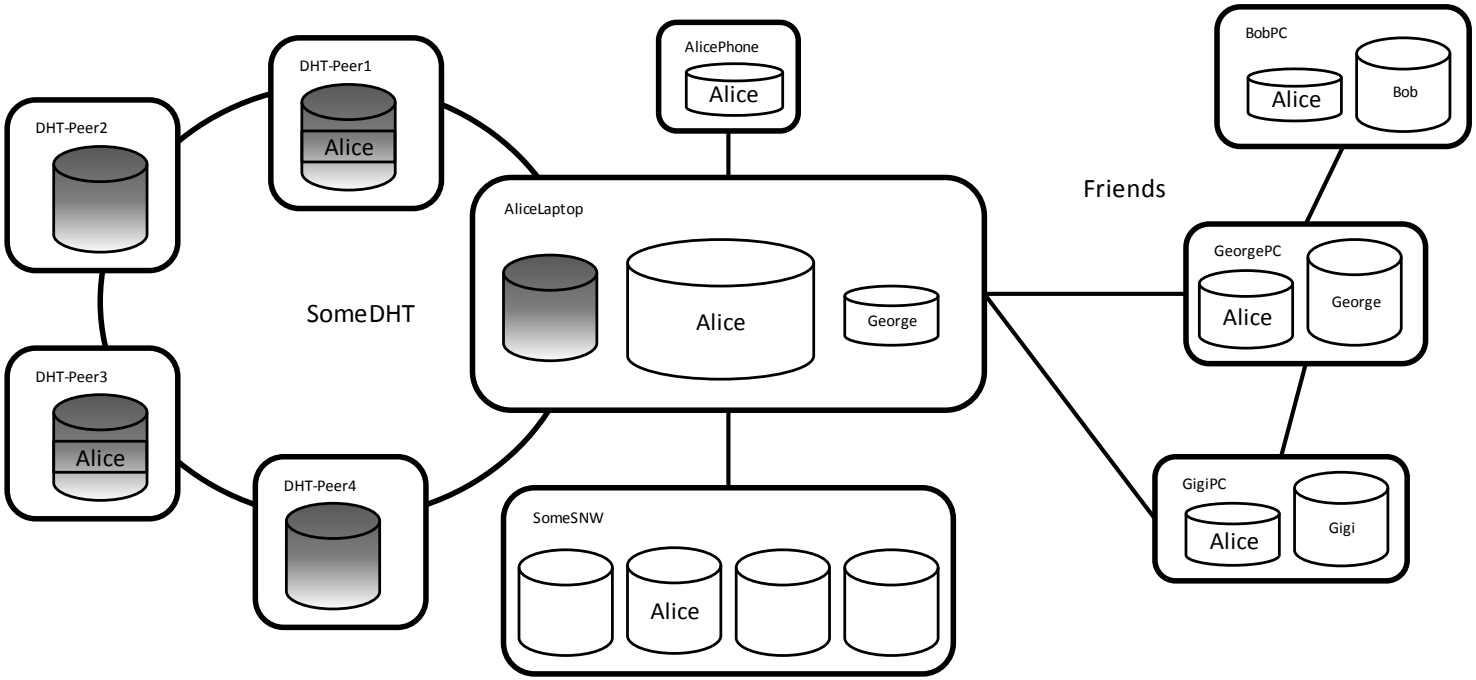
[1] INRIA Saclay & ENS Cachan

April 20th, 2010, *GDT DBWeb*

Joint work with Serge Abiteboul, Amélie Marian and Alkis Polyzotis

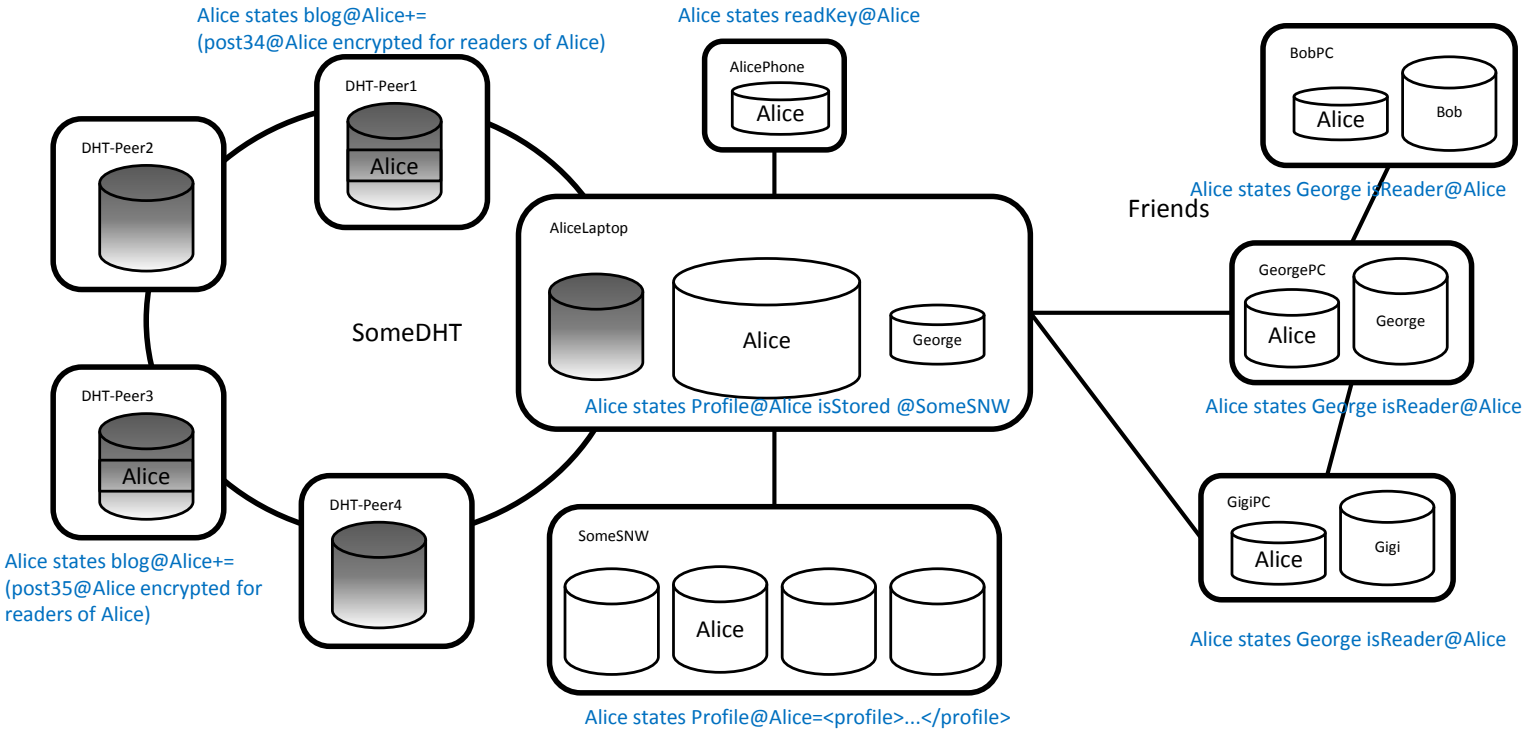
A motivating example

- The distributed knowledge base of Alice, a rockclimber:



A motivating example

- The distributed knowledge base of Alice, a rockclimber:



Goal

- Describe all kinds of distribution schemes (centralized, structured and unstructured P2P)
- Provide access control for reading and editing the data, and delegating this rights
- Execute any valid and only valid instruction (read or edit) on the data
- Enable reasoning on the knowledge (both on data and meta-data)

Contribution

- A model of distributed data with access-control and provenance
- Some constraint to guarantee properties of systems build on the model
- A system that manages distributed knowledge with privacy

Outline

Outline

Global view of the model

- Data and meta-data are all first class-citizen. They are represented as logical statement which are "valid" knowledge, enforcing read and edit rights
  - Two kinds of data statement: Document (read/write), Collection (read/append/remove)
  - Three kinds of meta-data statement: Access right, Key, Localization
- Instructions are used to request manipulation of data (get or update)

Principal

- A principal is an "agent" of the system, which may have some data, with a unique access control list for every different kind of access right.
  - A user (e.g. Alice), a sub-principal of the user which has some data with specific access right (e.g. AliceFriends)
  - A group of users (e.g. roc14, a rockClimbing group)
  - A peer (e.g. AliceLaptop, AlicePhone, SomeDHT, SomeSNW)
- A principal is authentified by an id and a pair of asymmetric keys. It is identified by the id and the public key.
- Anyone with the private key can behave as the principal itself. He *owns* the principal. In particular, he has the same rights as the principal itself. This pair of asymmetric key is immutable, so ownership is irrevocable.

Document

- Alice states news@roc14=T
- A document is the basic form of data. It has an *unique id* inside the principal and its content is an *xml tree* with internal references to other documents.
- Access rights: read and write
- Instruction:
  - Bob writeRequest news@roc14=T to Alice
  - Bob getRequest news@roc14 to Alice

Collection

- Alice states rocks@roc14+=rocherFin@roc14
- A collection is a set of references to documents (inside or outside the principal).
- Access rights: read, append and remove
- Instruction:
  - Bob removeRequest rocks@roc14-=rocherReine@George to Alice
  - Bob getRequest rocks@roc14 to Alice

Localization

- Alice states alldocuments@roc14 isStored @Facebook
- A localization is a meta-data specifying where a knowledge (or a type of knowledge) is stored.
- Access rights: readWhere, writeWhere
- Instruction:
  - Bob removeWhereRequest news@roc14 isStored @Facebook to Alice
  - Bob getWhereRequest news@roc14 to Alice

Access right

- Alices states Bob isReader@roc14
- An access right is a meta data specifying that a principal has a given access right on the principal: read, append, remove, write, readRights, readWhere, writeWhere, own
- Access right: readRights, own
- Instruction:
  - Bob revokeRequest George isReader@roc14 to Alice
  - Bob getRequest isReader@roc14 to Alice

Key

- Alices states readKey@roc14
- A key is a meta data specifying a pair of asymmetric keys for a given access right on a principal.
- Access right: own, own
- The logical statement do not care about the value of the key, but the implementation of the logical statement in the system has to contain it.

Factification (1)

- The factification is the transformation of an instruction into a statement. It is easy to check that a statement is valid.

Bob writeRequest news@roc14=T to Alice

$\Rightarrow$ Alice states news@roc14=T requester Bob at 2010/04/01 10:00:00GMT

- It is important to keep trace of the provenance to be sure that nothing weird happen outside the system.

Factification (2)

- Alice states news@roc14=(T encrypted for readers of roc14) requester Bob at 2010/04/01 10:00:00GMT
- To enforce edit access right, the statement is signed with the key corresponding to the needed access right.
- To enforce read access right, the statement data is encrypted if needed.
- The statement keep trace of the *performer* of the factification with a signature and of the id of the *requester*. The performer has to keep trace of the instruction of the requester. Moreover, the statement keep trace of the local *time* of factification.

Provenance

- The exchange of knowledge keep the full trace of the previous exchange, by piling up signatures of the principal which send the data
  - Bob says Alice says Alice states new@roc14=T to Bob to George

Outline

System properties

- We are interested by the following properties of system
  - Well-formedness: the data is syntactically correct
  - Soundness: only valid instructions (read or edit) are executed in the system.
  - Completeness: any valid instruction (read or edit) is correctly executed.
- Nothing prevents a participant to do something illegal such as giving a document to some unauthorized party. But then, the unauthorized party cannot prove that he obtained the information legally.

Well-Formedness

- Well-formedness: the data is syntactically correct
  - The sequence of exchange of data is well-formed with respect to sender and receiver.
  - All the signatures are correct with respect to data and use the correct type of key.
  - The owner key are correct with respect to the id of the principal.
- We assume that all the data in our system is well-formed (since the non-well-formed data is rejected).

Soundness(1)

- A system is *(data-privacy)* sound if a principal can read and edit only the content of data he has access to according to access rights.
- Soundness can also be more restrictive: right-privacy and docId-privacy
- Problem: what does "according to access rights" means? We need some form of consistency.

Soundness(2)

- A principal follows *sound-rule* if
  - he factifies only when he has a proof that the requester has the edit right.
  - when sending knowledge to another principal, he encrypts the information with the corresponding key, unless he has a proof that the recipient has the read right.
- A system is monotone if it only allows adding knowledge.

Theorem

When all principals in a monotone well-formed system respect sound-rule, the system is guaranteed to be (data-privacy) sound. Moreover, if some principals does not obey the rule, their coalition will not get more access-right than the union of their access-rights.

Completeness

- A system is *complete* if any valid instruction (read or edit) will be correctly executed.
- To reach completeness, we need
  - Awareness: a principal should be able to know about the identifier of data he has access to.
  - Reachability: a principal should be able to find the corresponding data
  - Read-denial-free: a principal should be able to read the corresponding data
  - Update-denial-free: a principal should be able to edit the corresponding data
- To guarantee these properties, we need some consistency of knowledge, e.g. using a concurrency control mechanism.

Verification with provenance

- If some peers misbehave, we want to detect misbehavior as soon as it reach a "good" peer. This verification is done using the trace of provenance.
- The verification can be done by each peer, by some authority or by the principal corresponding to the data depending of visibility of access control.
- The verification can be systematic, randomly distributed, or guided by the detection of a problem.

Outline

Distribution schemes

- @Home: one trusted peer hosts all the data of the principal
- @Host: one untrusted peer hosts all the data of the principal, encrypted
  - @DHT: a set of untrusted peer hosts redundantly the data of the principal
- @Friends: each principal hosts his own knowledge and some data of other principals, he is interested in.

@Home

- @Home: one trusted peer hosts all the data of the principal
  - The trusted peer owns the principal. It does all the factification and stores the data.
  - When receiving a "get" request, the trusted peer check the access control and send the data "in clear" if the requester has read access.
- Example: Facebook, your web site
- Problem: you have to fully trust one peer.

@Host

- @Host: one untrusted peer hosts all the data of the principal
  - The peers with the edit access rights do factification and encrypt the data with respect to the read access rights. They use time to live to avoid denial of update of documents.
  - The untrusted peer stores the data encrypted. It may use access control list if it can read it to control the distribution, but it is not mandatory.
  - The peers with the read access rights have to decrypt the data.
- Example: Mozilla weave
- Problem: you have to trust the peer for serving the data. You can't (cheaply) avoid denial of update for collections and denial of answers.

@DHT

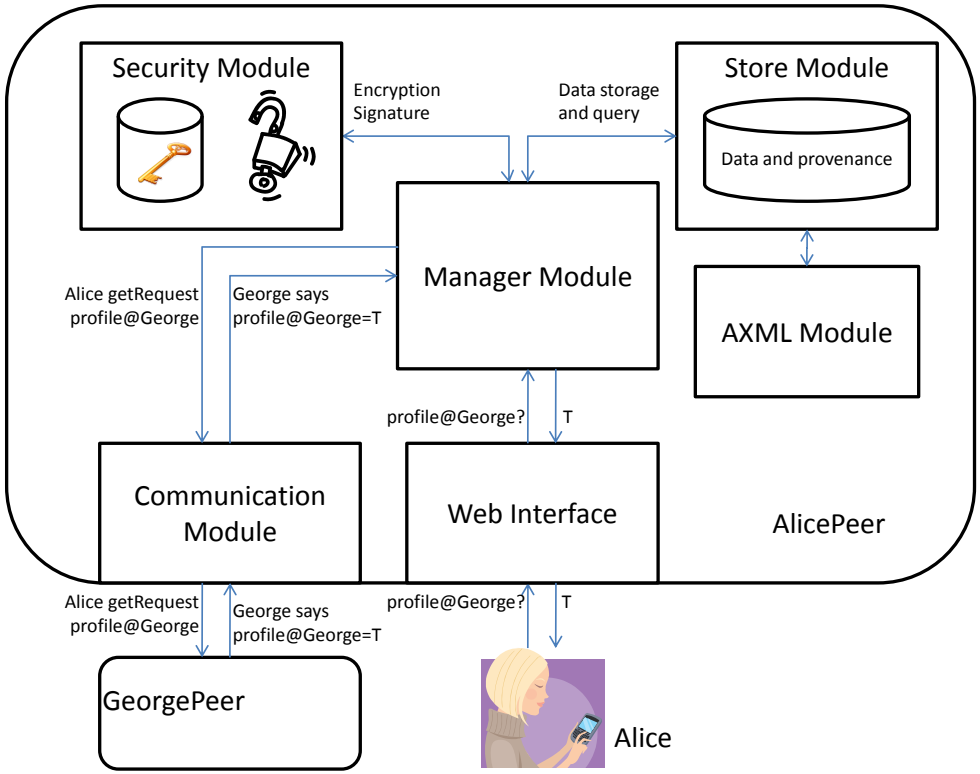
- @DHT: a set of untrusted peer hosts redundantly the data of the principal
  - Same organization as @Host, but exploiting redundancy to overcome previous problems.
  - To avoid denial of update, the peers do rounds of mutual certification of the list of items in collections.
- Example: an untrusted dht (e.g. PAST system)

@Friend

- @Friend: a set of trusted peer caches the data they care about
  - The friends do not get more access right than they have. So the factification is done by peer with edit access right.
  - The friends are trusted to check access control before sending data in clear.
- Example: a trusted network of friends
- Problems: as previously seen, proving soundness and completeness here is difficult. Moreover, localization is also more challenging.

The Pastis system

- The architecture of the system:



Outline

Query-based access control

- High level specification: *use queries to define access control*
- Different kinds of query specification: datalog, xquery...
- Semantic problem: Does the evaluation of the access control by a central omniscient authority give the same result as the distributed one (Evaluation of the queries on the local data by each peers)?
  - In general case, undecidable (comparison of datalog programs)
  - Decidable case we know about are not very expressive

Outline

Conclusion

- A model and a system for a distributed knowledge base with access rights
- Directions for future work:
  - Query processing based on distributed datalog evaluation
  - Study of scenarios of distribution and verification of properties
  - Building some wrapper (Facebook, OpenSocial...) to demonstrate private data-integration on the web