Polymorphic Encryption and Pseudonymisation (PEP)

ECP/PI.lab, Den Haag
Bart Jacobs and the PEP team
bart@cs.ru.nl
Feb. 22, 2017

Outline

Introduction

A PEP crash course

Polymorphic encryption

Polymorphic pseudonymisation

Formal description, mathematically

ElGamal crypto

Basic protocols

Where we are, so far

Introduction

A PEP crash course

Formal description, mathematically

Parkinson disease

Verily: under Alphabet, besides Google

Cooperation outline
Which medical data will be collected?

▶ Clinical data, via e-forms
▶ Biospecimens, via samples
   - analysed separately by RadboudUMC and by Verily
   - results will be shared via PEP
▶ MRI & ECG
   - images taken by Donders; large files
▶ Genetic data
   - also large
▶ Behavioural data, via wearables, and possibly apps

These sources will each use different pseudonyms of the same subject; data will be combined in the PEP database.

Holy grail of personalised medicine

▶ New development in healthcare: fine-grained personalised treatment based on statistical outcomes of large scale analysis of patient data
▶ In personalised healthcare one has to deal with:
   - identifiable medical data for the diagnosis and treatment of individual patients;
   - pseudonymised patient data for large scale medical research;
   - multiple sources of patient data, including in particular (wearable) self-measurement devices and apps.
   - the need to ensure confidentiality of patient data — and integrity, authenticity and availability too;
▶ The PEP framework is designed for this situation; it offers:
   - privacy protection by design via encryption and pseudonymisation
   - support for the basic data-access functionality for research, and potentially treatment too, in personalised healthcare.

Timeline

Oct’16  Project start
May’17  Beta version of PEP must be up-and-running
   - this is when enrolments of study participants start
   - clinical and biospecimen data has highest priority
   - wearable data must also be uploadable — via Verily
June’19  Enrolment of last of 650 patients
   - PEP database must be fully functioning, for both upload and download of all datagroups
   - possibly other (inter)national research groups have joined by then
Oct’21  Project end — but successive one-year extension are possible

Legal essentials

▶ RadboudUMC is data controller, Verily is processor
   - the contract is under N.L law
   - Google infrastructure may be used, in sub-processor role
▶ Data storage and exchange will be done only via PEP
   - pseudonymisation and encryption are intrinsic
   - De-pseudonymisation attempts are forbidden
▶ Study participation is based on explicit consent
▶ Raw & sanitised data are shared via PEP, but “inventions” are separate

External legal experts of Project Moore and Considerati have drafted the contract and helped with the negotiations.

New EU privacy regulation, and PEP

▶ Europe has recently (May 2016) adopted the GDPR
   - GDPR = General Data Protection Regulation
   - effective after a 2-year transition period
▶ It demands data protection by design and default
   - mandatory DPIA = data protection impact assessment
   - hefty fines for non-compliance
▶ The GDPR encourages innovation, as long as organisations implement appropriate safeguards
   - it allows for subsequent processing that is “compatible”

Don’t whine about the GDPR, but check what modern crypto can do for you!

This is where PEP comes in.
Traditional (public key) encryption, pictorially

- **Encryption of data**: putting it in a locked chest
- **Decryption of data**: unlocking the chest

**Terminology:**
- public key
- private key

Polymorphic locks

- Traditionally, only the owner of the private key can decrypt
- In polymorphic encryption we use malleable locks:
  - with multiple keys
  - By turning the wheel, the lock can be morphed to a specific key:

Polymorphic encryption scenario (no pseudonyms yet)

- Sensitive device data are stored under polymorphic encryption
- Later on, device user gives doctor X access to the data:

Basic idea in polymorphic pseudonymisation

- Each user/patient A has a unique identifier pid_A (= patient identifier)
  - e.g., social security number, like BSN in NL
  - This pid can be “morphed” into pseudonyms, different per data handler
  - We call the pseudonym for data handler X, generated from pid_A, the local pseudonym of pid_A at X
    - The central TransCrypto can create these local pseudonyms — again in a blind manner

Polymorphic pseudonyms, pictorially

- An encrypted pseudonym is a pid in a chest with an extra wheel:
  - This second wheel changes the content, in a blind manner
  - The TransCrypto can set both wheels coherently, so that participant X can decrypt and find the local pseudonym of pid at X
- There are now two chests:
  1. one data-chest, as for polymorphic encryption
  2. one pseudonym-chest, with an extra wheel
Storage scenario, with pseudonyms

- The user (device) puts medical data in the data-chest, and his/her pseudonym in it.
- The TransCryptor adjusts both wheels on the pseudonym box — but does nothing with the data box!
- The encrypted data are stored under the local pseudonym of the user, independent of any encryptions.

Retrieval scenario, with pseudonyms

- Doctor X wants to get stored data for a patient:
  - she knows pseudonym and sends it in a pseudonym box
  - The Storage Facility finds his local pseudonym for the patient, and sends all associated (encrypted) data back.

Where we are, so far

- Introduction
- A PEP crash course

Formal description, mathematically

ElGamal crypto
Basic protocols

ElGamal basics

Let \( G \) be an additive (elliptic curve) group with generator \( g \) of prime order \( p \) (so \( p \cdot g = 1 \)).

- Keys: \( x \in \mathbb{Z}_p \) private key, \( y = x \cdot g \) is associated public key
- recall discrete log problem, for hiding \( x \)
- Encryption of \( M \in G \)
  \( \langle r \cdot g, r \cdot y + M \rangle \) where \( r \in \mathbb{Z}_p \) is random
  - This \( r \) randomises the ciphertext.
- Decryption of \( (b, C) \) using private key \( x \)
  \( C = x \cdot b \)
- Correctness: decryption after encryption is identity:
  \( (r \cdot y + M) - (x \cdot (r \cdot g)) = (r \cdot (x \cdot g) + M) - (r \cdot (x \cdot g)) = M \).

ElGamal manipulations

We introduce explicit notation, retaining the public key \( y \)

\[
E_G(r, M, y) = (r \cdot g, r \cdot y + M, y)
\]

We describe three operations on ElGamal ciphertexts:

1. **re-randomise**: to change the appearance, but not the content
2. **re-key**: to change the target, who can read the ciphertext (1)
3. **reshuffle**: to raise the plaintext to a certain power (1)

These operations will be defined as three functions \( RR, RK, RS \) each of type, independent of any encryptions:

\[
G^3 \times \mathbb{F}_p \longrightarrow G^3.
\]

(1) Re-randomisation

Definition of \( RR : G^3 \times \mathbb{F}_p \rightarrow G^3 \)
Define re-randomisation with \( s \in \mathbb{F}_p \) as:

\[
RR((b, C, y), s) \overset{\text{def}}{=} (s \cdot g + b, s \cdot y + C, y)
\]

Lemma

The re-randomising is an encryption of \( M \) with random \( s + r \), that is:

\[
RR(E_G(r, M, y), s) = G((s + r \cdot g, s \cdot y + r \cdot y + M, y))
\]

Proof:

\[
RR(E_G(r, M, y), s) = G((s + r \cdot g, s \cdot y + r \cdot y + M, y)) = G((s \cdot g + s + r \cdot g, s \cdot y + s \cdot y + r \cdot y + M, y) = E_G(s + r, M, y).
\]
(2) Re-keying (wheel on lock \( \llbracket 2 \rrbracket \))

Definition (of \( RK : G^1 \times \mathbb{F}_p \rightarrow G^1 \))
Define re-keying with \( k \in \mathbb{F}_p \) as:
\[
    RK((b, C, y), k) \overset{\text{def}}{=} \left( \frac{1}{b}, C, k \cdot y \right)
\]
where \( b \in \mathbb{F}_p \) is the inverse of \( b \).

Lemma
This re-keying is an encryption of \( M \) with public key \( k \cdot y \), that is:
\[
    RK((g, r, y + M, y), k) = \mathcal{E}(k, M, k \cdot y)
\]
It can be decrypted with adapted private key \( k \cdot x \).

Proof:
\[
    RK((g, r, y + M, y), k) = \mathcal{E}(k, M, k \cdot y) = \mathcal{E}(k, M, k \cdot x) = y \cdot K \rightarrow \mathcal{E}(k, M, k \cdot y)
\]

Some algebraic properties
(1) Re-keying and re-shuffling commute:
\[
    RK(RS((b, C, y), k), n) = RS(RK((b, C, y), k), n)
\]
(2) Re-randomisation is a group action, of \( \mathbb{F}_p \) on \( G^3 \)
\[
    RR(RK((b, C, y), s), s') = RK((b, C, y), s' + s)
\]
\[
    RR((b, C, y), 0) = (b, C, y)
\]

(3) Re-shuffling (wheel on chat \( \llbracket 3 \rrbracket \))

Definition (of \( RS : G^1 \times \mathbb{F}_p \rightarrow G^3 \))
Define re-shuffling with \( n \in \mathbb{F}_p \) as:
\[
    RS((b, C, y), n) \overset{\text{def}}{=} (b, n \cdot C, y)
\]

Lemma
This re-shuffling with \( n \) is an encryption of \( n \cdot M \) with random \( n \cdot r \):
\[
    RS((g, r, y + M, y), n) = (n \cdot r, g, n \cdot (r \cdot y + M), y)
\]

Proof:
\[
    RS((g, r, y + M, y), n) = \mathcal{E}(n \cdot r, n \cdot M, y)
\]

Polymorphic encryption via re-shuffling
- There is a master private key \( x \in \mathbb{F}_p \), with public key \( y = x \cdot g \in G \).
- Each participant \( A \) has a diversified private key \( x_A = K_a \cdot x \).
- Only the TransCrypto knows the table of pairs \( (A, x_A) \), in a HSM.
- \( A \)’s public key is: \( y_A = y \cdot x_A = K_a \cdot x \cdot g = K_a \cdot y \).
- Polymorphic encryption of \( D \) is \( \mathcal{E}(r, D, y) \), with master public key \( y \).
- Anyone can encrypt her data \( D \) in this way, and put it into storage.
- If needed, the TransCrypto can re-key this ciphertext to participant \( A \).
- Via: \( RK((g, r, y + M, y), K_a) = \mathcal{E}(x_A, D, K_a \cdot y) = \mathcal{E}(y_A, D, y_A) \).
- Then \( A \) can decrypt this, since \( y_A = K_a \cdot y \) is her public key.
- This only describes the bare essentials:
- Proper authentication, authorisation and logging must be added.

Polymorphic pseudonymisation via re-shuffling
- Each patient \( B \) has personal identifier \( \pi_B \in G \).
- \( B \)'s local pseudonym at \( A \) is \( \pi_B \cdot A = S_A \cdot \pi_B \).
- Only the TransCrypto knows these pairs \( (A, S_A) \).
- \( B \)'s polymorphic pseudonym is \( \mathcal{E}(r, \pi_B \cdot y, K_A) \).
- All \( B \)'s data (for storage) is sent to the TransCrypto with the PP.
- The TransCrypto re-shuffles and re-keys PP to the local pseudonym \( \pi_B \cdot BSF = S_{BSF} \cdot \pi_B \) of the Storage Facility.
- Via: \( RK((RS((g, r, \pi_B \cdot y), S_{BSF}), K_A \cdot y) = \mathcal{E}(S_{BSF} \cdot r \cdot \pi_B \cdot y, S_{BSF}) \).
- SF decrypts and uses this local pseudonym \( \pi_B \cdot BSF \) as database key to store the (polymorphically encrypted) data of \( B \).
- If doctor \( A \) wants to retrieve \( B \)'s data:
  - \( A \) sends \( PP \cdot \mathcal{E}(r, \pi_B \cdot y) \) to the TransCrypto, who re-keys and re-shuffles it to \( SF \), who obtains his local pseudonym of \( B \), and looks up and returns the requested data, which gets re-keyed to \( A \).

Conclusion
- Privacy and security are a license to operate in medical (big data) research.
- PEP will be a strategic high-profile open source project, potentially also with high impact, via a broad range of users.
- It provides essential infrastructure for (academic) medical research.
- It will be tested first in a large Parkinson study with Radboud UMC and Verily.
- PEP will be integrated with DRE (Digital Research Environment).
- Applications in other areas are exist, but are postponed.
- See https://pep.cs.ru.nl for more info and documentation.

- For more privacy-friendly technology:
  - https://privacybydesign.foundation