Polymorphic Encryption and Pseudonymisation (PEP)
For privacy-friendly Personalised Medicine
Bart Jacobs and the PEP team
bart@cs.ru.nl
Sept. 16, 2016

Outline

Introduction
Informal description, with pictures
Polymorphic encryption
Polymorphic pseudonymisation

Formal description, mathematically
ElGamal crypto
Basic protocols

Conclusions

Where we are, so far

Introduction
Informal description, with pictures
Formal description, mathematically
Conclusions

PEP overview

We present a new, innovative set-up for encryption and pseudonymisation of sensitive personal data.
It’s called polymorphic: the encryption and pseudonymisation can be “transcribed” to different recipients.
It can be applied in privacy-friendly cloud storage and identity management.
Here it will be described in a medical context.

There are many other application areas, e.g., in the Internet of Things — dealing with sensor / behavioural / surveillance data.

Here it will be described in a medical context.
Both encryption and pseudonymisation are highly relevant.
Data sources are e.g., medical equipment, doctor’s input, or self-measurement devices.
Well within the constraints of European regulation.

Cryptographic basis

Malleability of ElGamal public key encryption
- ideas originally developed by colleague Eric Verheul
- they form the basis for complex protocol (team effort)

The presentation contains two parts:
- an informal one, to convey the main ideas
- a formal one, for the underlying math and protocol hints

Personalised medicine & PEP

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:
- unprecedented privacy-protection via encryption and pseudonymisation.
- support for the basic data-access functionality for research, and potentially treatment too, in personalised healthcare.
New EU privacy regulation, and PEP

- Europe has recently (April 2016) adapted 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 set-up proper protection!

This is where PEP comes in.

Why “polymorphic”?

- In traditional encryption, one encrypts for some chosen recipient
  - this recipient holds the cryptographic key for decryption
- In polymorphic encryption, one encrypts in a generic manner
  - later on, this encryption can be transcribed/tuned so that any participant

Illustration in a medical setting:
1. any patient/user with a self-measurement device encrypts the collected (medical) data polymorphically
2. later, access to the data can be transcribed specifically:
   - e.g., for selected doctors, or researchers, or service companies

PEP overview picture: the “PEPcloud”

Where we are, so far

Introduction
Informal description, with pictures
Polymorphic encryption
Polymorphic pseudonymisation

Formal description, mathematically

Conclusions

Traditional (public key) encryption, pictorially

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

Terminology: \( \text{public key} \quad \downarrow \quad \text{private key} \)

Polymorphic locks

- Traditionally, only the owner of the private key \[ \text{can decrypt} \]
- 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:

  - The TransCrypto learns nothing about the data!

**Conclusions about polymorphic encryption**

- Sensitive data can be encrypted, without a priori fixing who can decrypt:
  - the data is inaccessible by the cloud storage provider
- At any later stage, the data can be made decryptable, for any participant:
  - this works by blind transcription
  - it can be repeated at will

**Basic idea in polymorphic pseudonymisation**

- Each user/patient \( A \) has a unique identifier \( \text{pid}_A \) (= patient identifier)
  - e.g. social security number, like BSN in NL.
- This \( \text{pid} \) can be “morphed” into pseudonyms, different per data handler:
  - data handler means: doctor, researcher, assessor, ... 
  - morphing into pseudonyms is done in a uniform manner
  - represented again as a wheel, that can be turned blindly
- We call the pseudonym for data handler \( X \), generated from \( \text{pid}_A \), the local pseudonym of \( \text{pid}_A \) at \( X \) written later as \( \text{pid}_A^@X \).
- The central TransCrypto can create these local pseudonyms — again in a blind manner.

**Polymorphic pseudonyms, pictorially**

- An encrypted pseudonym is a \( \text{pid} \) in a chest with an extra wheel:

**Storage scenario, with pseudonyms**

- The user (device) puts medical data in the data-chest, and his/her \( \text{pid} \) in the pseudonym chest, and sends both to the TransCrypto:

  - The TransCrypto adjusts both wheels on the pseudonym-box — but does nothing with the data box!

  - The encrypted data are stored under the local pseudonym of \( \text{pid} \) for the Storage Facility
  - the same happens with data from other sources

**Retrieval scenario, with pseudonyms**

- Doctor \( X \) wants to get stored data for a patient
  - she knows \( \text{pid} \) and sends it in a pseudonym box

  - The Storage Facility finds his local pseudonym for \( \text{pid} \) in the chest, and sends all associated (encrypted) data back:
Where we are, so far

Introduction

Informal description, with pictures

Formal description, mathematically

ElGamal crypto

Basic protocols

Conclusions

ElGamal basics

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
3. Re-shuffle: to raise the plaintext to a certain power

These operations will be defined as three functions $\text{RR}, \text{RK}, \text{RS}$ each of type, independent of any encryptions

$G^3 \times \mathbb{F}_p \rightarrow G^3$

(1) Re-randomisation

Definition of $\text{RR}: G^3 \times \mathbb{F}_p \rightarrow G^3$

Define re-randomisation with $s \in \mathbb{F}_p$ as:

$\text{RR}((b, C, y), s) \equiv (s \cdot g + b, s \cdot y + C, y)$

Lemma

This re-randomisation is an encryption of $M$ with random $s + r$, that is:

$\text{RR}(E_G(r, M, y), s) = E_G(s + r, M, y)$

Proof:

$\text{RR}(E_G(r, M, y), s) = E_G((s + r) \cdot g, (s + r) \cdot y + M, y) = E_G(s + r, M, y)$

(2) Re-keying (wheel on lock)

Definition of $\text{RK}: G^3 \times \mathbb{F}_p \rightarrow G^3$

Define re-keying with $k \in \mathbb{F}_p$ as:

$\text{RK}((b, C, y), k) \equiv (\frac{1}{k} \cdot b, C, k \cdot y)$

where $\frac{1}{k} \in \mathbb{F}_p$ is the inverse of $k$.

Lemma

This re-keying is an encryption of $M$ with public key $k \cdot y$, that is:

$\text{RK}(E_G(r, M, y), k) = E_G(\frac{r}{k} \cdot g, r \cdot y + M, k \cdot y)$

It can be decrypted with adapted private key $k \cdot x$.

Proof:

$\text{RK}(E_G(r, M, y), k) = \text{RK}((r \cdot g, r \cdot y + M, y), k) = E_G(\frac{r}{k} \cdot g, r \cdot y + M, k \cdot y)$

(3) Re-shuffling (wheel on chest)

Definition of $\text{RS}: G^3 \times \mathbb{F}_p \rightarrow G^3$

Define re-shuffling with $n \in \mathbb{F}_p$ as:

$\text{RS}((b, C, y), n) \equiv (n \cdot b, n \cdot C, y)$

Lemma

This re-shuffling with $n$ is an encryption of $n \cdot M$ with random $n \cdot r$:

$\text{RS}(E_G(r, M, y), n) = E_G(n \cdot r, n \cdot M, y)$

Proof:

$\text{RS}(E_G(r, M, y), n) = \text{RS}((r \cdot g, r \cdot y + M, y), n) = (n \cdot r \cdot g, n \cdot (r \cdot y + M), y) = E_G(n \cdot r, n \cdot M, y)$
Some algebraic properties

1. Re-keying and re-shuffling commute:
   \[ RK(RS((b, c, y), n), k) = RS(RK((b, c, y), k), n) \]

2. Re-randomisation is a group action, of \( F \) on \( G^3 \)
   \[ RR(RR((b, c, y), s), s') = RR((b, c, y), s + s') \]
   \[ RR((b, c, y), 0) = (b, c, y) \]

Polymorphic pseudonymisation via re-shuffling

- Each patient \( B \) has personal identity \( pid_B \in G \)
- \( B \)'s local pseudonym at \( A \) is \( pid_B \cdot A = S_A \cdot pid_B \)
- only the TransCrypto knows those pairs \( (A, S_A) \)
- \( B \)'s polymorphic pseudonym is \( \Sigma_B(r, \cdot pid_B, y) \)
- All \( B \)'s data (for storage) is sent to the TransCrypto with this PP
  - the TransCrypto re-shuffles and re-keys PP to the local pseudonym \( pid_B \cdot B = S_B \cdot pid_B \) of the Storage Facility
  - Via: \( RK(RS \cdot \Sigma_B, S_A \cdot pid_B, K_S) \) \[ = \Sigma_B(S_A \cdot pid_B, K_S, y) = \Sigma_B(S_A \cdot r, \cdot pid, y, y) \]
  - SF decrypts and uses this local pseudonym \( pid_B \cdot B \) as database key to store the (polymorphically encrypted) data of \( B \)
- If doctor \( A \) wants to retrieve \( B \)'s data:
  - \( A \) sends PP \( \Sigma_B(r, \cdot pid_B, y) \) to the TransCrypto, who re-keys and re-shuffles it to \( S_B \), who obtains its local pseudonym of \( B \), and looks up and returns the requested data, which gets re-keyed to \( A \)

Actual protocols are a bit more complicated

Polymorphic encryption via re-keying

- There is a master private key \( x \in \mathbb{F}_p \), with public key \( y = x \cdot g \in G \).
  - only the trusted key authority has \( x \), stored in a HSM
- Each participant \( A \) has a diversified private key \( x_A = K_A \cdot x \).
  - only the TransCrypto knows the table of pairs \( (A, K_A) \), in a HSM
  - \( A \)'s public key is: \( y_A = x_A \cdot g = K_A \cdot y \).
- Polymorphic encryption of \( D \) is \( \Sigma_A(r, D, y) \), with master public key \( y \)
  - anyone can encrypt her data \( D \) in this way, and put it in storage
  - if needed, the TransCrypto can re-key this ciphertext to participant \( A \)
    - via: \( RK(\Sigma_A(r, D, y), K_A) = \Sigma_A(r, D, K_A \cdot y) = \Sigma_A(r, D, y_A) \)
    - 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

Simplified example: key distribution protocol

Neither Key Server nor TransCrypto (Tweaker) learns \( A \)'s private key \( x_A \).

Where we are, so far

Introduction
Informal description, with pictures
Formal description, mathematically
Conclusions
Main points (PEP is PET)

- Polymorphic encryption & pseudonymisation give unprecedented privacy protection
  - while retaining basic functionality for personalised healthcare
  - in line with GDPR's data protection by design
- The PEP-technology can become a new NL/EU/...standard
- Province of Gelderland has given 750K€ support for development
- Nijmegen's Digital Security group is:
  - elaborating and documenting the design
  - developing prototype implementations
  - making the open source software freely available
  - working with interested parties to get this up and running
- PEP forms the backbone of new Parkinson studies, using data from multiple sources, including wearables
- See PEP whitepaper at http://eprint.iacr.org/2016/411