Eric Verheul
KeyControls and Radboud University Nijmegen
P.O. Box 9010,
NL-6500 GL Nijmegen
eric.verheul@keycontrols.nl

Bart Jacobs
Radboud University Nijmegen
P.O. Box 9010,
NL-6500 GL Nijmegen
bart@cs.ru.nl

Polymorphic Encryption and Pseudonymisation in Identity Management and Medical Research

To appear in: *Nieuw Archief voor Wiskunde*, 2017

This paper sketches how the classical ElGamal public key encryption system can be used in a novel way to provide new privacy protection mechanisms, notably in identity management and in medical research.

1. Introduction

In 2014 the first author identified a paradox in the foreseen Dutch eID scheme [3, 5]. This scheme allows citizens to authenticate to governmental organisations through private parties, like banks or telecom providers. For functional reasons, these parties would need to provide a national citizen identifier called 'BSN' to such governmental organisations. However, Dutch privacy regulation precludes private parties from processing the BSN. This led to the following question: is it possible to store the BSN in some encrypted form at an authentication provider such that it can be later transformed into a form decipherable by, and only by, the intended governmental organisation? During transformation the BSN should not temporarily emerge in the clear at the authentication provider, so that a common decrypt-encrypt transformation would not be suitable. Also, in the end, only the intended governmental organisation should be able to decrypt the BSN. A solution to this problem was needed. But the obvious approach to provide each governmental organisation with the same secret key for decryption would undermine the required level of security.

The second author encountered a similar challenge in healthcare and medical research. Here, different parties (doctors, researchers) wish to investigate patient data from various sources. To avoid cumbersome bilateral exchanges, a central repository is required. As in the eID case, privacy regulation mandates that these data be stored in encrypted form. Hence also in this case it would be desirable if the encrypted data (ciphertext) is transformable to a form that is locally decipherable for the different parties.

Both use cases gave rise to the development of Polymorphic Encryption and Pseudonymisation, abbreviated as PEP. With the similar techniques of polymorphic encryption and polymorphic pseudonymisation new security and privacy guarantees can be given which are essential in areas such as privacy-friendly identity management, (personalised) healthcare, medical data collection via self-measurement apps, and more generally in the internet of things and in data analytics.

The key ideas of polymorphic encryption are:

(1) Personal data can be encrypted in a 'polymorphic' manner and stored at a central party in such a way that the central storage facility cannot get access. Crucially, there is no need to fix *a priori* who can decrypt the data later, so that the data can immediately be protected at the source.

(2) Later on it can be decided who can decrypt the data, via some transformation of the encrypted data (ciphertext) which makes it locally decryptable via locally different (diversified) cryptographic keys. This decision will be made on the basis of a policy, in which the data subject should play a key role.

(3) This transformation of encrypted data can be performed by a trusted party in a blind manner, without seeing the content; the resulting transformed ciphertext is transformed into locally decryptable ciphertext, for a specific other party.

In an eID scheme the encrypted data can be a national citizen identifier (like BSN) stored at an authentication provider; it can be decided later which government organisation gets access to it, after a citizen's login request. In healthcare, a PEP-enabled measurement device — operated by a doctor or by a user himself — can immediately encrypt the data; the user can decide later that, for instance, doctors X, Y, Z may at some stage decrypt and use the data in their diagnosis, or medical research groups A, B, C may use it for their investigations, or third parties U, V, W may use it for additional services, *etc.*

This PEP technology can provide the necessary security and privacy infrastructure for big data analytics, where data comes from various sources, like in the internet of things. People can entrust their data in polymorphically encrypted form, and each time decide later to make (parts of) it available (decipherable) for specific parties, for specific analysis purposes. In this way users remain in control, and can monitor which parts of their data are used where, by whom, and for which purposes.

The polymorphic encryption infrastructure can be supplemented with a pseudonymisation infrastructure which is also polymorphic, and guarantees that each individual will automatically have different pseudonyms at different parties.

This paper provides an introduction to Polymorphic Encryption and Pseudonymisation (PEP), focusing on identity management and health care as two application areas.

The PEP framework is currently being implemented in the Dutch eID scheme, see Section 5. The framework is also elaborated into an open design and open source (prototype) implementation at Radboud University in Nijmegen, The Netherlands. The technology will be used and tested in a real-life Parkinson research project at the Radboud University Medical Center, see Section 6.

2. ElGamal revisited

The expression 'ElGamal' is used for one of the first asymmetric, public key crypto algorithms, named after its inventor [1]. It can be used both for encryption and for digital signatures. Here we only use the encryption version. This section recalls the basic definitions and results, assuming familiarity only with elementary group theory. In particular, it describes three operations on ElGamal ciphertexts that form the basis for PEP. Indeed, the PEP functionality exploits the 'malleability' of ElGamal encryption, see Lemma 2.1 below.

ElGamal works in a cyclic group. In practice we shall use (prime order subgroups of) elliptic curves [4] as groups, involving addition of points on a curve, and so we prefer additive notation for a group $\mathbb{G} = (\mathbb{G}, +, 0)$. Let $\mathbb{G}$ be a group of prime order q and let $G \in \mathbb{G}$ be a fixed generator. This means that q is the least non-zero natural number with $q \cdot G = 0$ and that each element $H \in \mathbb{G}$ can be written as $H = k \cdot G$ for a unique $k \in \{0, 1, \dots, q-1\}$. The latter set is the carrier of the field $\mathbb{F}_q$ of size q, which is how we shall write it from now on. With $\mathbb{F}_q^*$ we denote the non-zero elements of the field, *i.e.* its multiplicative group. A randomly selected element in a set is denoted by $\in_R$.

The security of ElGamal encryption depends on the hardness of the discrete logarithm (DL) problem in the group. The DL problem says: given $n \cdot G \in \mathbb{G}$, for some number $n \in_R \mathbb{F}_q$, then it is computationally infeasible to find n in polynomial time in $\log_2(q)$, *i.e.* in the number of bits in the binary representation of q. A suitable instance of $\mathbb{G}$ is the (largest prime order subgroup of the) Montgomery Elliptic Curve Curve25519[1], offering 128 bits of security, or the Brainpool320r1 curve[2] offering 160 bits of security. The latter curve is currently also used in European electronic passports, including the Dutch ones.

We recall the basics of ElGamal encryption.

Private key The private key y of a user is a random element in $\mathbb{F}_q^*$, which is kept secret by the owner.

Public key The public key $Y \in \mathbb{G}$ is the group element $Y = y \cdot G \in \mathbb{G}$. Due to the DL problem, y cannot (feasibly) be obtained from Y and G. This value Y is assumed to be known to everyone.

Encryption Let $M \in \mathbb{G}$ be a message that we wish to encrypt, with public key Y. ElGamal encryption is 'randomised' or 'probabilistic': it uses randomness in each encryption so that encrypting the same message twice gives different ciphertexts, with high probability. We choose a non-zero $r \in_R \mathbb{F}_q$ and encrypt M as the pair of group elements:

$$\langle\, r \cdot G,\ M + r \cdot Y \,\rangle. \tag{1}$$

We recall that a fresh (new) random number r should be used for each encryption.

Decryption Let a ciphertext pair $\langle B, C\rangle \in \mathbb{G} \times \mathbb{G}$ be given corresponding to the public key $Y = y \cdot G$. The ElGamal decryption of $\langle B, C\rangle$ is the group element:

$$C - y \cdot B. \tag{2}$$

(We use the letters B for blinding and C for cipher.) One can easily verify correctness, *i.e.* that decryption returns the original message M. Security is based on the DL problem.

Notation We shall write $\mathcal{EG}$ for the ElGamal encryption function, but with a minor twist. We define:

$$\mathcal{EG}(r, M, Y) = \langle\, r \cdot G,\ M + r \cdot Y,\ Y \,\rangle. \tag{3}$$

As before r is the random number that needs to be different each time. Notice that the function $\mathcal{EG}$ produces a 3-tuple in (3), instead of a 2-tuple in (1): its type is $\mathcal{EG}\colon \mathbb{F}_q \times \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G} \times \mathbb{G} \times \mathbb{G}$. This is purely for administrative reasons: it makes it easier to formulate the results in Lemma 2.1 below. We do not use a special function or notation for ElGamal decryption.

We now describe the three homomorphic properties of ElGamal that form the basis of PEP. They are used in the

[1]See `https://cr.yp.to/ecdh.html` for more information.
[2]See `http://www.ecc-brainpool.org`.

operations of *re-randomising*, *re-keying*, and *re-shuffling* that act on ciphertexts.

Lemma 2.1. *In the notation introduced above we define three functions $\mathcal{RR}, \mathcal{RK}, \mathcal{RS}$ each with type:*

$$\mathbb{G}^3 \times \mathbb{F}_q^* \longrightarrow \mathbb{G}^3$$

and describe their properties.

(a) *The* ***re-randomisation*** *of a triple $\langle B, C, Y\rangle \in \mathbb{G}^3$ with $s \in \mathbb{F}_q^*$ is defined via the function:*

$$\mathcal{RR}(\langle B, C, Y\rangle, s) \overset{def}{=} \langle\, s\cdot G + B,\ s\cdot Y + C,\ Y\,\rangle. \qquad (4)$$

If the input $\langle B, C, Y\rangle$ is an ElGamal ciphertext, then so is the output:

$$\mathcal{RR}\big(\mathcal{EG}(R, M, Y), s\big) = \mathcal{EG}(s + r, M, Y). \qquad (5)$$

This ciphertext decrypts to the original message M via the original private key y.

(b) *The* ***re-keying*** *with $k \in \mathbb{F}_q^*$ is defined via the function:*

$$\mathcal{RK}(\langle B, C, Y\rangle, k) \overset{def}{=} \langle\, \tfrac{1}{k}\cdot B,\ C,\ k\cdot Y\,\rangle, \qquad (6)$$

where $\frac{1}{k}$ is the multiplicative inverse of k in the field $\mathbb{F}_q$. We then have:

$$\mathcal{RK}\big(\mathcal{EG}(r, M, Y), k\big) = \mathcal{EG}(\tfrac{r}{k}, M, k\cdot Y). \qquad (7)$$

This ciphertext decrypts to the orignal message M via a different private key $k\cdot y$.

(c) *The* ***re-shuffling*** *with $n \in \mathbb{F}_q^*$ is defined as a function:*

$$\mathcal{RS}(\langle B, C, Y\rangle, n) \overset{def}{=} \langle\, n\cdot B,\ n\cdot C,\ Y\,\rangle. \qquad (8)$$

Then:

$$\mathcal{RS}\big(\mathcal{EG}(r, M, Y), n\big) = \mathcal{EG}(n\cdot r, n\cdot M, Y). \qquad (9)$$

Hence in this case we can decrypt with the original private key to a re-shuffled message $n\cdot M$.

Proof All results are obtained by easy calculations. As an illustration we prove that equation (5) holds: re-randomisation (4) on an ElGamal encryption yields a new ElGamal encryption of the same message with the same public key, but with random number $s + r$, since:

$$\begin{aligned}\mathcal{RR}\big(\mathcal{EG}(r, M, Y), s\big) &\overset{(1)}{=} \mathcal{RR}(\langle r\cdot G, r\cdot Y + M, Y\rangle, s)\\ &\overset{(4)}{=} \langle s\cdot G + r\cdot G, s\cdot Y + r\cdot Y + M, Y\rangle\\ &= \langle (s + r)\cdot G, (s + r)\cdot Y + M, Y\rangle\\ &= \mathcal{EG}(s + r, M, Y). \qquad \square\end{aligned}$$

The purpose of re-randomisation in the first part of Lemma 2.1 is to create a copy of an ElGamal encryption that is unlinkable to the original. The obtained unlinkablity is equivalent to a mathematical problem called the Decision Diffie-Hellman problem in $\mathbb{G}$ which is believed to be hard in the elliptic curve groups mentioned earlier. This problem can be formulated as: given $H \in_R \mathbb{G}$ and the quadruple $(G, H, a\cdot G, b\cdot H)$ for $a, b \in_R \mathbb{F}_q^*$ decide if $a = b$. Compare [2, Theorem 10.20]. Sometimes we shall combine the re-keying and re-shuffling operations. The next result tells that the order of such combinations does not matter.

Lemma 2.2. *The re-keying and re-shuffling operations $\mathcal{RK}$ and $\mathcal{RS}$ from Lemma 2.1 commute. Explicitly:*

$$\mathcal{RS}\big(\mathcal{RK}(\langle B, C, Y\rangle, k), n\big) = \mathcal{RK}\big(\mathcal{RS}(\langle B, C, Y\rangle, n), k\big).$$

Proof This follows from an easy calculation:

$$\begin{aligned}\mathcal{RS}\big(\mathcal{RK}(\langle B, C, Y\rangle, k), n\big) &= \mathcal{RS}\big(\langle \tfrac{1}{k}\cdot B, C, k\cdot Y\rangle, n\big)\\ &= \langle n\cdot(\tfrac{1}{k}\cdot B), n\cdot C, k\cdot Y\rangle\\ &= \langle \tfrac{1}{k}\cdot(n\cdot B), n\cdot C, k\cdot Y\rangle\\ &= \mathcal{RK}\big(\langle n\cdot B, n\cdot C, Y\rangle, k\big)\\ &= \mathcal{RK}\big(\mathcal{RS}(\langle B, C, Y\rangle, n), k\big). \qquad \square\end{aligned}$$

Based on the above lemma we can combine re-keying and re-shuffling into a single function $\mathcal{RKS}\colon \mathbb{G}^3 \times (\mathbb{F}_q^*)^2 \to \mathbb{G}^3$ by:

$$\mathcal{RKS}(\langle B, C, Y\rangle, k, n) = \langle \tfrac{n}{k}\cdot B, n\cdot C, k\cdot Y\rangle. \qquad (10)$$

3. Polymorphic encryption

From now on we assume that there is a system-wide fixed group $\mathbb{G}$ with generator $G \in \mathbb{G}$ of prime order q. Also, some trusted party has generated a master private key $y \in_R \mathbb{F}_q^*$, with corresponding public key $Y = y\cdot G$. The private y is securely stored, for instance in a hardware security module (HSM). It will not be used for decryption, but only for generating other, derived private keys.

Given certain (personal) data D, anyone can form what we call the *polymorphic encryption* of D, of the form:

$$\mathcal{EG}(r, D, Y) \qquad \text{where } r \in_R \mathbb{F}_q^*. \qquad (11)$$

This means that any *data source* can encrypt data, using the master public key Y, and a self-chosen random number r. Our aim is to transform this ciphertext in such a way that dedicated parties can decrypt it.

We consider a collection of *service providers* S_j, for some finite index sets of j's. In order to perform their services, they need to get access to (parts of) the polymorphically encrypted data. In an identity management context this could be a governmental organisation and in an healthcare context this could be a doctor or a medical researcher. Note that in the first context a data source can polymorphically encrypt any data and not only the BSN.

In our setup, there is for each service provider S_j a secret number $s_j \in_R \mathbb{F}_q^*$ that is only known to a trusted party called the *transformer*. The service provider obtains a private key $y_j \in \mathbb{F}_q^*$ which has the form $y_j = s_j\cdot y$, where y is the master private key, mentioned earlier. The corresponding public key Y_j of S_j is then equal to $s_j\cdot Y$, where Y is the master public key. Indeed:

$$y_j\cdot G = (s_j\cdot y)\cdot G = s_j\cdot(y\cdot G) = s_j\cdot Y = Y_j.$$

Given some ciphertext $\langle B, C, Y\rangle$ arising as in (11), the transformer can turn it into a ciphertext that can be decrypted by a given service provider S_j. This is done via re-keying with the secret factor s_j, as in:

$$\mathcal{RK}(\langle B, C, Y\rangle, s_j) = \langle \tfrac{B}{s_j}, C, s_j \cdot Y\rangle = \langle \tfrac{B}{s_j}, c, Y_j\rangle.$$

As we have seen in Equation (7), via such re-keying, any data D that is polymorphically encrypted with the master public key Y, becomes encrypted with the public key Y_j of service provider S_j. Hence this service provider can, after this intervention of the transformer, decrypt the data.

We remark that the transformer should also apply re-randomisation on either the input or output of the transformation to avoid linkability issues. Notice that the security of the system rests on having two separate trusted parties, one holding the master private key y, and one 'transformer' holding the key factors s_j for each service provider S_j. If these two trusted parties collude, the system breaks down. Notice that the transformer manipulates ciphertexts, but cannot see the content. This is a very powerful and useful feature that we will further discuss in Sections 4 and 6.

4. Polymorphic pseudonymisation

In some cases service providers do not only want access to personal data but also want to have a persistent identifier related to the person to which the data pertains. That is, for the same person this identifier is the same over all sources providing data. In an identity management context this could be a webshop that is not allowed to process the BSN but needs a persistent identifier to give clients access to their own accounts. Different webshops should get different identifiers for the same client, so that they cannot combine their records — simply based on the identifier. Researchers in a healthcare context typically are not allowed to process the BSN either, but need to be able to link the data from various sources to the same individual.

To facilitate these requirements PEP supports service provider specific pseudonyms that can accompany the data. To this end, we assume that the data sources also have access to a 'global' personal identification number Id of the person to which it relates. In a Dutch setting one can think of (some hash of) the earlier mentioned BSN.

In the previous section we assumed a master public key Y and a transformer which holds for each service provider S_j a secret key factor s_j. We now assume that there exists another a master public key $Z = z \cdot G$ and that the transformer has for each S_j secret key factors t_j similar to s_j and additional 'pseudonym' factors u_j. Thus, these t_j and z play the same role as s_j and y. All these factors s_j, u_j, t_j are random but fixed.

For the actual usage of these pseudonyms, the transformer plays an important role again. Suppose service provider S_j also wants access to a pseudonym related to the person with identity Id. Then the data source first embeds Id into the group $\mathbb{G}$ through an (one-way) embedding $\mathcal{I}(.)$. Then the data source polymorphically encrypts $\mathcal{I}(\mathit{Id})$ using public key Z. This results in the *polymorphic pseudonym* $\mathcal{EG}(r, \mathcal{I}(\mathit{Id}), Z)$, and sends this together with the index j to the transformer. The transformer looks up the key factor t_j and the pseudonym factor u_j for service provider S_j, and performs both re-keying (with t_j) and re-shuffling with u_j, written as $\mathcal{RKS}$ in (10). This gives:

$$\begin{aligned}\mathcal{RKS}\big(\mathcal{EG}(r, \mathcal{I}(\mathit{Id}), Z),\, t_j,\, u_j\big) &= \mathcal{EG}(\tfrac{r}{t_j},\, u_j \cdot \mathcal{I}(\mathit{Id}),\, t_j \cdot Z)\\ &= \mathcal{EG}(\tfrac{r}{t_j},\, u_j \cdot \mathcal{I}(\mathit{Id}),\, Z_j).\end{aligned}$$

The result is the encrypted local pseudonym of the form $u_j \cdot \mathcal{I}(\mathit{Id})$ for S_j, which can be decrypted by S_j. Notice that the transformer learns nothing, except that someone is accessing service provider S_j. Each time this process is run, it produces the same local pseudonym $u_j \cdot \mathcal{I}(\mathit{Id})$ at service provider S_j, and a different local pseudonym $u_k \cdot \mathcal{I}(\mathit{Id})$ at a different service provider S_k. Pseudonym unlinkability is guaranteed through the hardness of the Decision Diffie-Hellman problem. As remarked earlier, the transformer should apply re-randomisation on either the input or output of the transformation to avoid linkability issues.

5. Polymorphic pseudonymisation in the Dutch eID scheme

In the projected Dutch eID scheme a central government organisation called *BSN Linking* (BSN-l) plays the role of data source discussed in Sections 3 and 4. The transforming role is played by (private) parties performing authentication for the government. As part of user (citizen) registration, authentication providers provide BSN-l with information uniquely identifying the citizen, *e.g.* first and last name, date of birth *etc.* BSN-l then looks up the citizen and its BSN. The BSN-l forms both a Polymorphic Idenitity (PI) and a polymorphic Pseudonym (PP). The PI is simply a polymorphic encryption $\mathcal{EG}(r, \mathrm{BSN}, Y)$ of the BSN. The Polymorphic Pseudonym is a polymorphic encryption of the form $\mathcal{EG}(s, \mathcal{I}(\mathrm{BSN}), Y)$. The embedded BSN, *i.e.* $\mathcal{I}(\mathrm{BSN})$, of Section 4 is based on a keyed hash function (HMAC). Although the embedded value should never be accessible outside BSN-l, the keyed hash ensures that the BSN cannot be derived from it. Both PI and PP are then sent to the requesting authentication provider and stored in a client database. A citizen can register at multiple authentication providers, for instance in order to have a back-up authentication mechanism.

If registration was successful, the authentication provider supplies the citizen with a (strong) means of authentication, linked to the PI/PP pair. This, for instance, could be a smart card, a challenge/response token or an authentication APP on a mobile device. If a citizen wants to login to a web service he is re-directed to an authentication provider of his choosing — where he has been registered already. By use of the authentication means, the citizen can be linked to its PI/PP in the client database of the authentication provider. If the web service is allowed to use the BSN, the authentication provider then blindly turns the PI to an Encrypted Identity holding the BSN using Equation (7). The Encrypted Identity is then sent to the organisation who can decipher the BSN from it. If the organization is not allowed to use the BSN, the authentication provider selects the PP and blindly turns this to Encrypted Pseudonym via (10). This is then sent to the organisation who can decipher a local pseudonym from it.

Dutch governmental organisations are allowed by law to use the BSN, but only if strictly necessary. If a pseudonym suffices, then that should be used. This is also known as the *data minimisation principle* stipulated in European privacy regulations. Hence in the case of governmental organisations there is a choice, namely to use the Polymorphic Identity (PI) — for BSN — or the Polymorphic Pseudonym (PP) — for a pseudonym — at the authentication provider. The status controller introduced below is a first example of a governmental service based on pseudonyms instead of BSNs.

To facilitate these transformations, the authentication providers are given secret factors s_j, t_j, u_j by the government which need to be stored in a hardware security module. Notice that in the polymorphic setup, authentication providers do not get access to citizen BSNs solving the paradox from the introduction. Actually, the polymorphic setup can also solve another paradox. In the setup indicated above, authentication providers know both the identities of citizens and the service providers that they want to login to. There are many cases where just registering that a user accessed a specific service can constitute a breach of privacy. Such cases include a user retrieving his results of a medical test or a user having an online psychiatric consultation. This issue becomes even more manifest if one is using private organisations that also provide other services. As an illustration, suppose one is regularly logging into an online consultation for alcoholics through a bank acting as authentication provider. How comfortable would one then be to apply for a mortgage or a car insurance application at that bank?

We note that privacy regulations mandate that authentication providers need user consent to send information to the governmental organisation. So not supplying the authentication provider with the identity of the governmental organisation is not suitable. In the projected Dutch eID scheme [3] such 'privacy hotspot' issues are procedurally mitigated: authentication providers are required to separate their registrations holding identifying user data, *e.g.* name, address *etc.*, from registrations holding usage data, *i.e.* authentication transactions. Note that the polymorphic setup conveniently caters for this as authentication providers can store transactions under a local pseudonym.

This lead to the question if this separation can be technically enforced: is it possible that an authentication provider authenticates a user for an organisation without knowing the identity of the user? This is paradoxical as the authentication provider is required to identify the user and to personally provide him with means of authentication. This paradox can also be solved through the polymorphic setup via a personal PEP-enabled smart card. This is actually being developed for the public authentication provider (DigiD) in the Dutch eID scheme. For this version of the eID system the PI and PP are not (only) stored by the authentication provider, but also on a contactless Dutch identity card, or driver's license card (hereafter simply called eID card). During authentication DigiD reads the PI and PP from the eID card whereby the card re-randomises these first. DigiD is then able to do the transformation for a governmental organisation but cannot determine the identity of the citizen. Actually, if the same citizen would authenticate one second later, DigiD would not even be able to determine this. This is due to the hardness of the Diffie-Hellman Decision problem mentioned earlier. To determine if the card has not been revoked, *e.g.* after loss or theft, DigiD also uses the polymorphic setup. DigiD forms an encrypted pseudonym for a so-called status controller and requests the status of the card by sending this to the controller. The status of the card is maintained by the issuer of the card which is also provided an encrypted pseudonym during production of the card. See also [5].

To allow his eID card to be read by DigiD, the user needs to connect a contactless card reader to his computer or to use a mobile device (smartphone) supporting Near Field Communication (NFC). The eID card is heavily based on electronic passport technology; in effect the PI/PP on the card are protected as fingerprints on passports. Through this technology it is also arranged that the user is technically in control of whether his card is providing both a PI and a PP to DigiD or only a PP. This allows for applications such as referenda where the pseudonym enforces that a citizen can cast his ballot only once but where BSN usage would violate ballot secrecy.

6. Polymorphic encryption and pseudonymisation for medical research

The second application area for PEP that we briefly elaborate on is medical research. In addition to traditional 'one-time' medical data sources, like an ECG or MRI scan, researchers nowadays like to have continuous, real-time access to patient data, for instance via various wearable monitors and activity trackers. This presents challenges for protected data management.

Via polymorphic encryption each data item D can be stored securely at some storage facility as $\mathcal{EG}(r, D, Y)$. Each device can itself compute such encryptions for the data that it generates, since all that is needed is the public key Y. As before, a 'transformer' can keep key factors for each participating researcher or doctor, and re-key the data so that it can be decrypted by that particular party.

In addition, via polymorphic pseudonyms it can be achieved that different researchers receive different pseudonyms for the same patient. This makes it hard to combine data, for instance after data loss or theft. Again, the pseudonymisation factors need to be associated with each participating party, known by the transformer, who can then re-shuffle and re-key in order to form local, decryptable pseudonyms.

By also providing local database pseudonyms to the researchers in encrypted form, researchers can put their findings back into the database, so that it can become visible for other participating researchers. Use of re-randomisation of encrypted pseudonyms avoids linkability issues. Even though these researchers all have different polymorphic pseudonyms, the enriched data that they put back will end up with the right individuals.

The PEP technology will be used for the first time in 2017 on a larger scale in a medical research project on Parkinson's disease[3], set up jointly by the Radboud University Medical Center and by Verily, the life science branch of the Google group, now called Alphabet. This study will involve 650 patients, who will be monitored for three years. Verily will provide wearable devices for this purpose. Radboud's computer security group, to which the authors belong, contributes with an implementation of the PEP technology[4].

Apart from the basic functionality sketched above, the PEP implementation provides authentication and authorisation for the various participants. The study is open to other (international) research groups. They will first have to submit a research plan to a supervisory body, and, after approval, will get the appropriate (derived) cryptographic keys, in order to access parts of the collected data that are relevant for their research. In this set-up they will also get their own (polymorphic) pseudonyms for the patients involved.

Thus, the PEP infrastructure functions as a secure database with encryption and pseudonymisation. This sounds ideal, but there are also some restrictions. We mention the two most prominent ones.

(1) It always remains possible to de-pseudonymise patients via the contents of the data, esp. with rare symptoms, or by combining the data with other sources. PEP will not protect against this. In the Parkinson study such de-pseudonymisation is simply forbidden by contract.

(2) When data is stored in encrypted form, searching in the stored data, in order to select specific parts, is not possible[5]. In principle, a researcher will have to download all the data, decrypt locally, and then search and select. This problem is alleviated by storing the encrypted data together with unencrypted meta-data. These meta-data can be used for selection, for instance based on content or dates.

Once the PEP software is sufficiently tested and stable, it will be made available as open source.

References

[1] T. ElGamal, A Public Key Cryptosystem and a Signature scheme Based on Discrete Logarithms, IEEE Transactions on Information Theory 31(4), 1985, pp. 469-472.

[2] J. Katz, Y. Lindell, Introduction to Modern Cryptography, CRC PRESS, 2008.

[3] Ministry of the Interior and Kingdom Relations, Uniforme Set van Eisen, version 1.0, 15-12-2016.

[4] D. Hankerson, A. Menezes and S. Vanstone, Guide to Elliptic Curve Cryptography, Springer, 2004.

[5] E.R. Verheul, The polymorphic eIDAS token, Keesing Journal of Documents & Identity, February 2017.

[3]See `http://www.parkinsonopmaat.nl/`

[4]See `pep.cs.ru.nl` for more information; the PEP development is funded by the province of Gelderland.

[5]There are advanced cryptographic techniques for searching in encrypted data, but they have not been integrated (yet) with PEP.