Privacy Requirements Implemented with a JavaCard

Anas Abou el Kalam
LIFO – CNRS / ENSIB, France
anas.abouelkalam@ensi-bourges.fr

Yves Deswarte
LAAS-CNRS, Toulouse, France
yves.deswarte@laas.fr

ABSTRACT

Privacy is extremely important in healthcare systems. Unfortunately, most of the solutions already deployed are developed empirically. After discussing some of such existing solutions, this paper describes an analytic and generic approach to protect personal data by anonymization. This approach is then applied to some representative scenarios. The architecture and its implementation with a Javacard are finally presented.

Our analysis, solution and implementation are generic enough to be adapted to various collaborative systems that process sensitive data such as e-commerce, e-government, social applications, etc.

Keywords: Privacy and security, collaborative environments, healthcare systems, Javacard.

1. Introduction

Many current information systems neglect security or privacy, or give more importance to security (in order to protect the system owner's assets) than to privacy (that would protect other people's personal data). In this paper, we suggest an analytic approach to privacy, and in particular for healthcare information systems. These systems cover most of the needs generally found in other applications: networking of organizations, sensitivity of information, and diversity of security requirements. Indeed, although healthcare networking facilitates data communication, it creates serious security risks. On the one hand, healthcare providers must reliably identify the patients and manage all the information that they need to provide care to patients. On the other hand, exchanging and sharing healthcare data between various actors endangers the patient's privacy, e.g., enabling attacks by inference on personal information.

First, we discuss examples of how the United States and the European countries attempt to implement their healthcare privacy legislation. Second, we present a systematic methodology that links privacy needs and adequate solutions. Then we propose a generic architecture that meets the privacy requirements. Finally, we give the details of our implementation and we list the benefits of our work.

2. Some existing solutions

2.1. Example of anonymization in the United States

In the United States, the Social Security Administration uses a "*Tricryption Engine*" to protect medical data against malicious (internal and external) attacks. The Tricryption Engine is a large encryption and automated key management system. It encrypts data with a per-call generated cryptographic key, encrypts the key and encrypts the link between the data and the key (Figure 1).

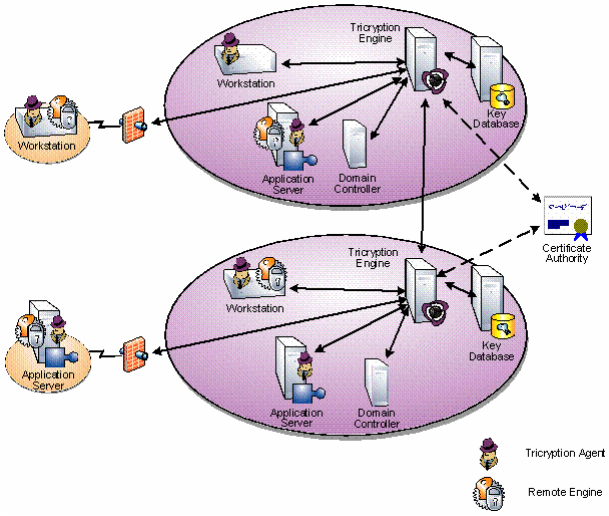

Figure 1: Description of the tricryption engine used in the USA.

The full process is detailed below:

- Sensitive data to be encrypted are selected by the user, and a request for encryption is sent.
- A randomly generated, symmetric session key is created; and a random Key ID is created.
- The key is encrypted.
- The encrypted key and its Key ID are stored in a Key Database.
- The Key ID is encrypted, producing a Hidden Link.
- The personal data are encrypted, using the session key.
- The encrypted data and the Hidden Link are returned to the user.
- The encrypted data and the session key used to encrypt them are completely separated, both physically and logically, and the link between them is hidden.

The Tricryption Engine includes a fully integrated administration module, which supports the administration of the Tricryption system, including the management of user authentication and authorization.

Note that the administration module as well as the Key Database are two critical entities that should be well protected, otherwise all the whole security of the system collapses.

2.2. Example of anonymization in Germany

The German National Cancer Registry gathers medical statistics related to German cancer cases. The procedure of the population-based cancer registration is realized in two steps by two institutions [1]. In the first stage, the Trusted Site accumulates the data recorded by doctors. The Trusted Site anonymizes these data by an asymmetric procedure, e.g., a hybrid IDEA-RSA encoding. The identifying data is encrypted with an IDEA session key, generated randomly. The IDEA key is encoded by a public RSA key. A control number (a pseudonym) is thus generated, using different attributes of the personal data that are sufficient to identify each patient uniquely. This control number is generated by using a one-way hash function and a symmetrical ciphering algorithm (IDEA). To allow data coming from the different federal Lander to be linked, the control number generation procedure and the key are unique ("Linkage Format"). The Trusted Site transfers both the encrypted patient-identifying data and the epidemiological plaintext data to the Registry Site.

The latter stores the record in the register database and brings together different records belonging to the same patient. After this matching of data, a random number is added to the control number and the result is symmetrically encrypted by IDEA ("Storage Format"). To match new records, the control numbers must be deciphered back from the "Storage Format" to the "Linkage Format".

2.3. Example of anonymization in Switzerland

The Swiss Federal Office for Statistics (SFSO) is responsible for collecting medical data (in Switzerland). To preserve the patients' privacy, the SFSO has contacted the Swiss Federal Section of Cryptography (SFSC) [2]. Their analysis concludes that it is not necessary to know to whom a given medical record belongs; however the SFSO needs to recognize if two different records belong to the same patient.

First, identifying data (date of birth, sex, last and first name) is replaced by a fingerprint, called anonymous linking code: *fingerprint* = H (*ID-Data*), with H being a secure hash function.

Before transmitting medical data to the SFSO, the hospital generates a session key *c*; this key is then used to encrypt the fingerprint during the transmission: $IDEA\{fingerprint\}_c$; a public key cryptosystem (RSA) is used to transmit the session key $RSA\{c\}_E$ using the SFSO public key *E*.

After reception, "*c*" is retrieved by using the SFSO private key *D*; the encrypted fingerprints are then decrypted, and uniformly re-encrypted by the symmetric key *K* of the SFSO: they become the anonymous linking codes used as personal codes. The key *K* is distributed among several trusted persons, using Shamir's secret sharing technique.

However, it is easy to notice that the intermediate steps of these transformations should never be visible to the SFSO operators. Indeed, how can we be sure that the secret key *c* and the fingerprints are never recorded in a storage medium? It is clear that these steps (calculation phases) should be done in a well-protected hardware module (a kind of secure "Black-box"). In addition, inviolable access control mechanisms (e.g., specific tamperproof hardware), could improve the protection. The aim is that only trustworthy persons, acting together, should carry out the composite operation.

2.4. Example of anonymization in France

French hospitals [3] transform the patient's identity by using a one-way hash function H (e.g., SHA). Actually, two keys have been added before applying H. The first pad, *k*1, is used by all senders of information: "$Code_1$=H (k_1|*Identity*)", and *k*2 used by the recipient: "H (k_2 | $Code_1$)". The aim of k1 (resp. k2) is to prevent dictionary attacks by a recipient (resp. a sender).

However, this protocol seems both complex and risky: the secret key should be the same for all information issuers and remains the same over time. Moreover, these keys must always remain secret: if a key is compromised, the security level is considerably reduced. It is very difficult to keep a key that is largely distributed secret for a long time. Hence, new keys must be generated and distributed periodically. The same applies when the algorithm (or the key length) is proven not sufficiently robust any more. But, how can we link all the information concerning the same patient before and after changing the algorithm/key? If this problem occurs, the only possible solution consists in applying another cryptographic transformation to the entire database, which may be very costly.

Table 1 summarizes the four solutions presented above.

Table1: Summary of the existing solutions.

Country	Purpose	Technique
USA	Social security data processing	Secret keys (Tricryption)
Germany Switzerland	Statistics	Hybrid encryption + hashing
France	Linking medical data for evaluation purposes	Symmetric keys + hashing

3. Analytic approach

3.1. Privacy needs

Most of the solutions presented above have been developed empirically and concern only one specific use. A first question that arises is "is it possible to develop a generic solution?" Second, we believe that before calling for technical or organizational solutions, it is necessary to develop a systematic methodology. In this sense, privacy analysis can be expressed with respect to two levels of abstraction:

- the *request* in the form of privacy needs to be satisfied.
- the *response* in the form of security functionalities and solutions to implement.

However, our methodology suggests some intermediary steps between the request and the response. In particular, it is necessary to clearly express the privacy *needs*, identify the privacy *objectives* (e.g., information to protect, threats to avoid) and specify the privacy *requirements* (formalization of needs, identification of functionalities).

The *privacy needs* represent the user's expectations; they depend on the system, the environment, etc. generally, their form is neither very explicit nor very simple to formalize. For instance, in healthcare systems, some privacy needs could be: both directly and indirectly nominative data should be anonymized; a patient appears in a database (e.g., for a medico-commercial study) only if he gives his consent; etc.
The next two sections present what we mean by privacy objectives and requirements.

3.2. Privacy objectives

We define the privacy objective according to one of the three following properties, applied to the anonymization function:

- *Reversibility*: hiding data by encryption. In this case, from encrypted data, it is always possible to retrieve the corresponding original nominative data.
- *Irreversibility*: the property of anonymization. The typical example is a one-way hash function. Once replaced by anonymous codes, the original nominative data are no longer recoverable.
- *Inversibility*: this is the case where it is, in practice, impossible to re-identify the person, except by applying an exceptional procedure restricted to duly authorized users. This exceptional procedure must be done under surveillance of a high trustworthy authority like the medical examiner, the inspector-doctor or a trustworthy advisory committee. This authority can be seen as the privacy guarantor. Actually, it is a matter of a pseudonymisation according to the common criteria terminology [5].

3.3. Privacy requirements

At this stage, the analysis is carried on by taking into account the possible attacks, the environment, etc. For instance, knowing that a malicious user can deduce confidential information by using illegitimate types of reasoning, we can deduce that the anonymization function should resist to attacks by inferences.

In general, two kinds of privacy requirements must be studied for any anonymization system: the "*linking*" requirements and the "*robustness*" requirements.
Linking allows associating (in time and in space) one or several pseudonyms to the same person. Linking can be temporal (e.g., always, sometimes, never) or geographic (e.g., international, national, local).

The robustness requirements concern illicit disanonymization. We distinguish robustness to reversion and to inference. The *reversion robustness*

concerns the possibility to invert the anonymization function, for example if the used cryptographic techniques are not strong enough. The *inference robustness* concerns data disanonymization by means of unauthorized computation, e.g., by inference.

3.4. Solution characterization

Once we have specified the privacy requirements, it is time to choose and characterize the most suitable solutions. In particular, we have to specify:

- the *type* of solution to develop: is it an organizational procedure, a cryptographic algorithm, a one-way function, or a combination of subsets of these solutions?
- the *plurality* of the solution to implement: do we need simple, double or multi-anonymization?
- the *interoperability* of the solutions that must be combined: transcoding (manually) translating (mathematically) or transposing (automatically) several anonymization systems into another one.

4. Representative scenarios

4.1. Medical data transmission

The sensitivity of the information exchanged between healthcare providers (e.g., between biology laboratories and physicians) emphasizes the *needs* for confidentiality and integrity on transmitted data. Moreover, it is necessary that only the legitimate addressee should be allowed to receive and read the transmitted data. The use of an asymmetric (or hybrid) cryptographic system seems suitable. The technique used should be reversible (*objective*) and robust to illegitimate reversion (*requirement*).

4.2. Evaluation of professional activities

In France, for evaluation purpose, the doctors have to send data related to their activity, to an authority called the "professional unions" (*the need*). The first privacy objective is to hide the patient's and the doctor's identities. However, when the purpose is to evaluate the physician's behavior (to assess care quality), it should be possible to re-identify the concerned physician. Our study of the French law allowed us to identify the following anonymization objectives:

- Inversible anonymization (pseudonymisation, in the sense of the common criteria [5]) of the physician's identities: only an user duly authorized to evaluate the physician's behavior can re-establish the identities.
- Inversible anonymization of the patient's identifiers: only consulting doctors can reverse this anonymity.

4.3. Evaluating hospital information systems

In order to evaluate hospital information and allocate resources systems while reducing budgetary inequality, some authorities (e.g., the PMSI in France) analyses the healthcare establishments activities. Given that the purpose is medico-economic (and not epidemiologic), it is not necessary to know to whom a given medical information belongs (*anonymization*). On the other hand it is important to recognize that different data are related to the same, anonymous person even if they come from different sources at different times (*linkability* [7]). Having said that, every patient must (always) have the same irreversible, anonymous identifier for the PMSI.

4.4. Statutory notification of disease data

Some diseases have to be monitored, through statutory notification, to evaluate the public healthcare policy (e.g., AIDS) or to trigger an urgent local action (e.g., cholera, rabies). Various needs can be identified: prevention, care providing, epidemiological analysis, etc. The main objectives are *anonymization* and *linkability*. Furthermore, universal linking, robustness to inversion, and robustness to inference are the main requirements.

These objectives could be refined by taking into account the purpose of use. In fact, would we like to globally evaluate the impact of prevention actions? Would we like to institute a fine epidemiological surveillance of the HIV evolution, and to finely evaluate the impact of therapeutic actions, as well as a follow-up of certain significant cases?

Indeed, the purpose of use has important consequences on the nature of data to be collected and so on the privacy requirements. Currently, we identify the following findings related to data impoverishment, to reduce inference risks:

- Instead of collecting the zip code, it is more judicious to collect a region code.
- Instead of collecting the profession, we think that a simple mention of the socio-professional category is sufficient.
- Instead of mentioning the country of origin it is sufficient to know if the HIV positive person has originated from a country where the heterosexual transmission is predominant.

4.5. Processing of medical statistical data

For statistical processing and scientific publications, not only should data be anonymized, but it also should be impossible to re-identify the concerned person. Indeed, even after anonymization, identities could be deduced by a malicious statistician if he could combine several well-selected queries, and possibly, by complementing the reasoning by external information. The problem of statistical database inference has been largely explored in previous works [6, 7]. In some cases, the solution could be to exchange the attributes values (in databases) so that the global precision of the statistics is preserved, while the precise results are distorted. However, the inherent difficulty in this solution is the choice of values to be permuted. Another solution could modify the results (of statistical requests) by adding random noise. The aim is to make request cross-checking more difficult.

Therefore, the privacy requirements for statistical processing could be the anonymization inversibility, non linkability and robustness to inference are essential.

4.6. Focused epidemiological studies

In order to improve care quality, it is sometimes desirable to re-identify patients, especially in some focused studies such as cancer research protocols and genetic disease follow-up. For instance, in the case of cancer research protocols, the process starts by identifying the disease stage, then the protocol corresponding to the patient is identified, and finally, according to this protocol, the patient is registered in a regional, national or international registry. The epidemiological or statistical studies of these registries could bring out new results (concerning patients following a certain protocol). In order to refine these studies and improve the scientific research, it is sometimes useful to re-identify the patients, to link some data already collected separately, and finally complement the results. We can thus conclude that this is a matter of inversible anonymization. Only authorized persons should be able to reverse the anonymity (e.g., consulting physician), and only when it is necessary.

5. A systematic approach

5.1. General scheme

In the previous sections, we explained our analysis method and we applied it to some representative scenarios. Now, we give shape to our analysis by developing a generic solution that satisfies the raised requirements and summarizes the six possible use cases (Figure 2).

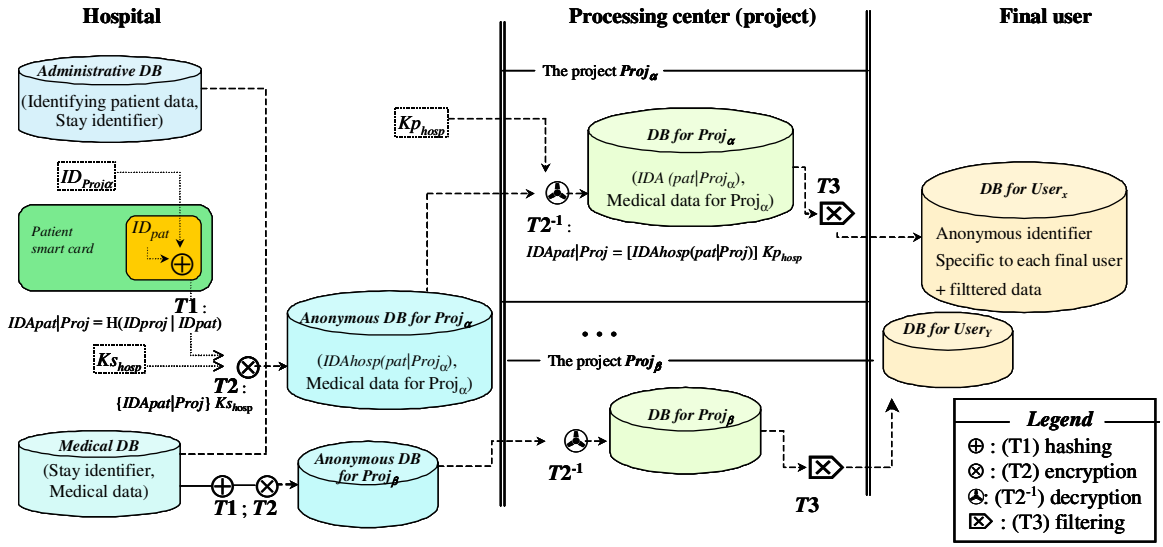

Figure 2: The suggested anonymization procedure.

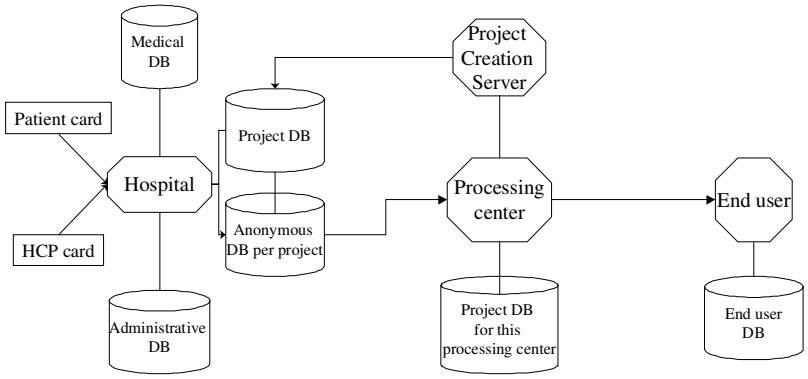

Figure 3: Architecture scheme.

5.2. Transformations processed in hospitals

In hospitals, three kinds of databases can be distinguished: administrative, medical and several anonymized databases. Each anonymized database contains the information for a particular project. A project is a program or a study intended for statistical, epidemiological or medico-economical data processing.

The transition from a medical database to an anonymized one requires the application of two transformations ($T1$, $T2$).

***T*1**: consists in calculating "$IDA_{pat|Proj}$", an anonymous identifier per person and per project. "*IDproj*" is the project identifier; while "*IDpat*" is the patient anonymous identifier (a random number). We suggest that *IDpat* is held under the patient's control, e.g., on his personal medical data smart card.

In the hospital, when entering data into anonymous databases, the user sends ID_{proj} to the card. The card already contains ID_{pat}. By supplying his card, the patient gives his consent to exploit his data as part of this project. The *T1* procedure, run by the smart card, consists in applying a one-way hash function (i.e., SHA-2) to the concatenated set ($ID_{proj} \mid ID_{pat}$):

$$\textbf{(T1)} \quad IDApat|Proj = \mathrm{H}\,(IDproj \mid IDpat)$$

Nevertheless, the transformation $T1$ does not protect against attacks where attackers try to link data held by two different hospitals. To make this clearer, let us take an example where Paul has been treated in the hospitals $Hosp_A$ and $Hosp_B$. In each of these two hospitals, Paul has consented to give his data to the project $Proj_\alpha$. Let us assume that Bob, an $Hosp_B$ employee, knows that the fingerprint X ($=IDA_{Paul|Proj\alpha}$) corresponds to Paul, and that Bob obtains (illicitly) access to the anonymous database held by $Hosp_A$ and concerning $Proj_\alpha$. In this case, the malicious user Bob can easily establish the link between Paul and his medical data (concerning $Proj_\alpha$) held by $Hosp_A$ and $Hosp_B$.

In order to face this type of attacks, a cryptographic asymmetric transformation (**T2**) is added. Thus, before setting up the anonymous databases (specific to each project), the hospital encrypts (using an asymmetric cipher) the fingerprint $IDA_{pat|Proj}$ with the key Ks_{hosp} specific to the hospital; (the notation "{M}K" indicates that M is encrypted with key K):

$$\textbf{(T2)} \quad IDAhosp(pat|Proj) = \{IDApat|Proj\}Ks_{hosp}$$

If we take again the previous scenario, the malicious user Bob cannot re-identify the patients because he does not know the decryption key Kp_A. Ks_{hosp} and Kp_{hosp} are a key pair of a public key cryptosystem, but that does not mean that Kp_{hosp} is really public : it is known only by the project processing centers and by the hospital's security officer (who knows also Ks_{hosp}, of course).

Basically, the anonymous databases intended to one or several projects are permanently sent by hospitals to processing centers. A processing center could be an association, an office for medical statistics, or a research center.

5.3. Transformations carried out upstream from processing centers

Data contained in the anonymous databases undergoes transformations that depend on $IDA_{proj|pat}$ and on Ks_{hosp}. Every center decrypts received data by using Kp_{hosp}:

$[IDAhosp(pat|Proj)]Kp_{hosp} =$
$[\{IDA_{pat|Proj}\}Ks_{hosp}]Kp_{hosp} =$
$IDA_{pat|Proj}$

Note that since the resulting data is associated to $IDApat|Proj$, each project can link data corresponding to the same patient (even if they come from different hospitals).

5.4. Transformations carried out before the distribution to the final users

Before their distribution to the final users (web publishing, press, etc.) the anonymized data can undergo a targeted filtering (a data aggregation, data impoverishment, etc.). If, in addition, the security objective is to forbid file linkage, it is advisable to apply another anonymization (e.g., by MD5) with a secret key $Kutil|proj$ generated randomly.

(T3) $IDApat|util = H(IDApat|Proj \mid Kutil|proj)$

In accordance with the needs, this last case corresponds to two different processes:

- if the aim is to allow full time linking, $Kutil|proj$ has to be stored by the processing center;
- inversely, if the center wishes to forbid linking data, the key is randomly generated just before each distribution.

6. Implementation

6.1. Architecture and components

Figure 3 describes a very-simplified architecture of our implementation. Our prototype has been tested on several platforms: Unix (CPU: HP 9000/L2000; OS: HPUX 11.00) Linux (CPU: Athlon 1.4 Ghz; OS: Mandrake Linux 10; kernel v2.6.3.4) Macintosh (CPU: Power G3 600MHz; OS: MacOS X 10.x) and Windows. The prototype requires about 50 Mbytes free disk space, MySQL or Oracle 10g, Java JRE 1.4.2 or later and JavaCard 2.1. SUN recommend a configuration with 1 Ko RAM, 16 Ko EEPROM, and 24 Ko ROM [8].

For the implementation, it is advisable to use cards that support cryptographic procedures (smartcards). The kit should provide a complete environment of development and should contain an interface that makes the communication easier with the smart card (e.g., the "JCardManager" interface provided in the Gemexpresso RAD III kit). Programming is done in the standardized language "*Javacard*". Javacard is a reduced-API Java. Even if this API provides most characteristics of Java (e.g., exceptions, constructors, inheritance, unidimensional arrays), it has some limits, insofar as the primitive types are limited to byte, shorts, and Boolean; it does not support cloning, threads, garbage collector, etc.

In order to use the application, you should have some software components such as: the patient cards, the Health Care Professional certificates (embedded in the "Health Care Professional" cards), the hospital keys (public and private keys) and the certification authority public key.

6.2. Keys generation

We used RSA for asymmetric keys (e.g., Kphosp and Kshosp). The implementation was as follows:

```
/* generateRSAKey */
public static KeyPair generateRSAKey() throws
RSAException {
      KeyPairGenerator kpg ;
      KeyPair kp ;
      try {
/* Define the encryption mechanism */
  kpg=KeyPairGenerator.getInstance("RSA") ;
    }
/* Managing the exceptions */
  catch (NoSuchAlgorithmException nsae) {
  throw new RSAException("Algo does not
exist") ;
      }
/* Specify the keys length: e.g., 2048 bits */
   kpg.initialize(2048) ;
/* Keys generation */
   kp = kpg.generateKeyPair() ;
   if (kp==null) throw new
RSAException("Error") ;
return  kp ; }
```

To generate symmetric keys (e.g., Kutil|proj) we could use the AES algorithm as follows:

```
public static Key generateAESKey() throws
Exception {
/* Specify the algorithm provider */
  Security.addProvider(new
org.bouncycastle.jce.provider.BouncyCastleProvi
der());
/* Secret Key generation */
KeyGenerator
kg=KeyGenerator.getInstance("AES");
Key k ;
/* Specify the keys length: e.g., 128 bits */
kg.init(128) ;
k = kg.generateKey() ;
if (k==null) System.out.println("Error") ;
return k; }
```

6.3. Communicating with the smartcard

The class managing the communication with the card is called APDU and is defined in the package "javacard.framework". The client's application sends the request to the card reader. The latter transmits the information to the Java Card Runtime Environment (JCRE). Data are then transmitted to the applet, which will have to answer.

Actually, the communication with the card is carried out in a buffer "APDU buffer" and must be processed in a "public void process (APDU apdu)" method. The applet is running in this method.

It is possible to get the content of the APDU buffer by using the following method: byte[] apdu_buffer = apdu.getBuffer().The data (contained in the buffer) can be read when the applet uses the following method: public "short setIncomingAndReceive() throws APDUException.

The mechanism could be summarized as follows:

```
public void process(APDU apdu) {
byte[] apdu_buffer = apdu.getBuffer() ;
        short total_bytes = (short)
(apdu_buffer[ISO7816.OFFSET_LC] & 0xFF);
// Read data in the "APDU" buffer
short read_count =
apdu.setIncomingAndReceive() ;
short bytes_left = (short) (total_bytes –
read_count) ;
while (true) {
 if(bytes_left <= 0) break ;
   read_count= apdu.receiveBytes((short)0) ;
   bytes_left -= read_count;
   } ……..}
```

6.4. Authentication

In our implementation we generated a certificate for patients as well as for Health Care Professionals (e.g., X.509-v3).

Besides, according to a challenge-response protocol, the authentication starts by checking the entered code. The "verify" method is invoked with a parameter containing the response of the card. This parameter corresponds to the object called APDU (Application Protocol Data Unit).

```
/* verify: check the certificate
// Sign with  keyPAC, and verify the equality
public boolean verify(PublicKey keyPAC) throws
RSAException
{
/* Concatenating data and hashing */
byte[] temp =
concat(concat(concat(concat(concat(concat(conc
at(version.getBytes(), autority.getBytes()),
num.getBytes()), lastName.getBytes()),
firstName.getBytes()), role.getBytes()),
expiration.getBytes()), keyPAC.getEncoded()) ;
return
RSACrypto.verify(temp,signature,keyPAC) ; }
```

6.5. Hashing, encryption and decryption

Hashing is applied in the T1 and T2 transformations, To calculate IDApat|proj and to sign (with the method Signature.getInstance("MD5withRSA")).

Encryption requires several steps:

- specify the provider "p=Security.getProvider("BC)";
- set the encryption system "cp = Cipher.getInstance("RSA","BC)";
- specialize the encryption mode "cp.init(Cipher.ENCRYPT_MODE,pk)",
- specify the data size "int blockSize = cp.getBlockSize()";
- and finally encrypt by using

```
for (int i = blockSize; i < message.length; i+=
blockSize) {
/* Intermediary encryption */
cp.update(message,debut,blockSize) ;
/* Encryption */
tmp1 = cp.doFinal() ;
tmp2 = concat(tmp2,tmp1) ;
debut += blockSize ;
}
return
concat(tmp2,(cp.doFinal(message,debut,messag
e.length - debut))) ;
}
```

7. Discussion

The solution that we propose guarantees several benefits. First, it is fine-grained, generic enough and could be easily adapted to different sector needs (e.g., social domain, demographic projects, etc.).

Second, the use of smartcards (that are sufficiently tamper-resistant) helps to keep the sensitive data (e.g., the patient identifier IDpat) secret and to protect the critical processes. Moreover, the secret as well as the anonymization is held under the patient's control. The patient's consent is required for each generation of an anonymized form of his personal data. Indeed, the medical data can appear in a certain database only if, by supplying his card, the patient allows the use of his

medical data as part of a certain project. The patient's consent is also necessary when reversing the anonymity. Let us take the example when the end user (e.g., researcher in rare diseases) discovers important information that requires re-identifying the patients. At first, it sends back results to the hospitals participating in the project (e.g., a given orphan disease study). When the patient produces his medical data card (which implies that he gives explicitly his consent), it is possible to calculate $IDA_{pat|Proj} = H(ID_{proj} \mid ID_{pat})$ and $IDAhosp(_{pat|Proj}) = \{IDA_{pat|Proj}\}Ks_{hosp}$, and so, to establish the link between the patient, his anonymous identifiers, and his medical data. A simple (and automatic) comparison between the anonymous identifier and the inversion list would trigger an alarm. This alarm asks the patient if he wants to consult the results.

Third, the solution resists dictionary attacks that could be run in various organizations: hospitals, processing centers or end users.

Fourth, contrary to existing solutions (e.g., the current French anonymization procedure), in our solution, the identifiers (ID_{proj}, ID_{pat}, $IDA_{pat|Proj}$ and $IDA_{pat|util}$) used in the various transformations are located in different places. Similarly, the keys (Ks_{hosp}, Kp_{hosp}) are held by different persons. Indeed, ID_{proj} concerns a unique project; the pair (Ks_{hosp}, Kp_{hosp}) is specific to one hospital; $IDA_{pat|util}$ is dedicated to a single end user. Moreover, ID_{pat} is specific to one patient, and only held on his card. Thus, even if a certain ID_{pat} (corresponding to Paul, for example) is disclosed (which is not easy!), only Paul's privacy could be endangered (but not all the patients' privacy, as it is the case in the current French procedure) and only for certain projects.

Finally, it is possible to merge data belonging to several establishments without compromising security nor flexibility. Indeed, if two hospitals ($Hosp_A$ and $Hosp_B$) decide to merge someday, it would be easy to link data concerning every patient that has been treated in these hospitals. In fact, each hospital would have to decrypt its data with its key Kp_{hosp}, and then encrypts the result by $Ks_{hosp_{AB}}$ the new hospital private key.

Furthermore, according to the security needs of the studied cases, we suggest to complement our solution by other technical and organizational procedures; In particular, the access to data has to be strictly controlled; a well-defined security policy must be implemented by appropriate security mechanisms (hardware and/or software). Reference [9] suggests a security policy and an access control model *(Or-BAC: Organization-Based Access Control*) that are suitable to collaborative systems. Indeed, Or-BAC offers the possibility to handle several security policies associated with different organizations.

It is also recommended to control the purpose of use by implementing intrusion detection mechanisms; in particular, these mechanisms should easily detect sequences of malicious requests (illicit inferences, abuse of power, etc.).

8. Conclusions

Privacy is a critical issue in many emerging applications and networked systems: e-commerce, e-government, e-health, etc. International legislations [10, 11], authorities as well as computer science communities are concerned by this problem. This paper presents an analytic approach and a generic solution to protect personal and sensitive data by anonymization. It also gives details of our architecture and our implementation. We have described, in particular, how authentication, hashing, encryption, decryption and communication with the smartcard have been programmed in the Javacard language.

A demonstration of our software will be available for the conference. In the near future, we would like to study the genericity of our solution and how it can be optimized for specific applications.

Acknowledgment

This study has been partly supported by the project MP6 (Modèles et Politiques de Sécurité pour les Systèmes d'Information et de Communication en Santé et Social) of the French research program RNRT and by the project PRIME (PRivacy and Identity Management for Europe) IST-507591 of the 6th European Research Framework. In particular, the experimentation described in this paper has been realized as part of the initial prototype of PRIME.

References

[1] Blobel B, "Clinical Record Systems in Oncology", in *Personal Medical Information – Security, Engineering and Ethics*, Anderson R.J. (Editor), Springer-Verlag, pp. 39–56, 1997.

[2] Jeanneret J.P., Jacquet-Chiffelle D.O., "How to Protect Patient's Rigottes to Medical Secret in Official Statistics", *Information Security Solutions Europe Conference* (*ISSE*), London, UK, September 2001, pp. 26-28.

[3] Quantin C., Bouzelat H., Allaert F.A., Benhamiche A.M., Faivre J. and Dusserre L., "How to ensure data security of an epidemiological follow-up", *International Journal of Medical Informatics*, vol. 49, No 1, 1998, pp 117-122.

[4] *Oggy Vasic, Using Tricryption to Protect Privacy*, Tech. Report ERUCES Data Security, 11 Feb. 2005, available at <http://www.eruces.com/images/stories/pdf/ERUCES_EFE_Protecting_Privacy_%20Final.pdf>.

[5] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, ISO/IEC 15408-1 (1999).

[6] Denning D. and Denning P., "Data Security". *ACM Computer Survey*, vol. 11, n° 3, September 1979, ACM Press, ISBN : 0360-0300, pp. 227-249.

[7] S. Castano, M. G. Fugini, G. Martella, P. Samarati, "*Database Security*", 1995, ACM press, ISBN: 0201593750, 456 pp.

[8] Z. Chen "*Java Card Technology for Smart Cards: Architecture and Programmer's Guide"* Addison-Wesley, 2000 ISBN: 0-201-70329-7, 400 pp.

[9] Abou El Kalam A., Balbiani P., Benferhat S., Cuppens F., Deswarte Y., El-Baida R., Miège A., Saurel C., Trouessin G., "Organization-Based Access Control", *4th International Workshop on Policies for Distributed Systems and Networks (Policy'03)*, Como, Italy, 4-6 June 2003, IEEE Computer Society Press, pp. 120-131.

[10] Resolution A/RES/45/95, General assembly of United Nations: *"Guidelines for the Regulation of Computerized Data Files"*, 14 December 1990.

[11] Directive 2002/58/EC of the European Parliament on: *"the processing of personal data"*, July 12, 2002.