A bank card file

Gilles Grimaud

January, 2021

You may have noticed that nowadays there is two ways for a website to ask for a payment. The first solution is to ask you a card number plus the CSV that is in the back of the card. The second one uses 3D Secure. 3D Secure should become prevalent because it allows to setup a more secure payment, where the seller website ask to the bank to authenticate the holder of the bank card. However this payment solution implies:

1. depending on the bank, each user has a different authentication interface ;
2. the authentication interface is asked for every payment.

Thus, numerous economic actors for the web stay away from this solution. Indeed, because of (1) the lost client ask help from the seller website about things (the authentication) which doesn't belongs to the bank. And because of (2) the "one-click" payment is not possible.

Thus, numerous historical seller website keep using the bank card + CSV system.

This solution, less protected, encourage to steal seller website files that contains bank card data. It can be an IT steal as well as a physical one, with potentially some system administrator involvement.

Also, the bank card GIEs (*Groupement d'intérêt économique*) have defined norms and audits precising the condition required for an online server to legitimately have bank card identifiers. We don't detailed the 1200 pages of the documentation that defines these norms, but we keep for the exercise the following points:

1. The server that stores and manages name/card number pair should be identified, it provides the following services:
   - i. Putting into service,
   - ii. Add a pair,
   - iii. Delete a pair,
   - iv. Find card numbers associated to a name ;
2. The security file must not rely on:
   - i. the security of the disk containing the server (this disk can be IT or physically stolen),

   ii. the security of the programs giving these services (they are on the disk),
   iii. the willingness of the server administrator (pressure can be put on him) ;

3. Also, name/card number pairs are stored in a ciphered file ;

4. Ciphering keys must never be (in clear text) on the disk or on the stored file, but they can be stored, during the putting into service of the server, on a RAM disk for example ;

5. The putting into service of the server (and the medium used to decipher the file) must be possible only under the authority of two persons in charge (one in charge for the technical aspect, one for the juridic aspect), that must be physically present and authenticated by two factors during the launching.

To understand these points, it is necessary to understand that the attacker model anticipates that the attacker can have a physical or logic access to the disk and the server program. Depending on the attacker model, an attack has failed, if after this attack nobody is able to use the file. That means denial of service is not feared.

We ask you to realise the proof of concept of this server. This PoC will demonstrate the proposed cryptography usage, and how it achieves to guarantee the required security. The PoC may be composed of bash or python script as well as services.

Part 1: shared responsibility

The security constraints stipulate that it is possible to start the service only under the authority of 2 persons in charge. The ciphered file should remains indecipherable to anyone (including the software creator) when the 2 factors authentication of these 2 persons in charge is absent. Since we suppose that the attacker have the program, obtaining the deciphering key should be possible only if the 2 persons in charge are present, by two factor.

This is called *security by design.*

Question 1:

Propose a solution for the deciphering such that it can be possible only by the two factor authentications of the two persons in charge. For practical reason, the two factors will be:

1. what I know (a password) and ;

2. what I have (a usb key).

For practical reason, the PoC won't rely on biometry.

Question 2:

Propose an implementation of the services 1.i., 1.ii, 1.iii and 1.iv in *bash*, *zsh* or in *python*. Moreover, since the putting into service supposes that the usb keys of the persons in charge and potential the disk have been initialized, propose an initialisation service 1.v. that initialises the two usb keys and the disk.

Part 2: Right delegation

Since persons in charge are not always available and the proper functioning of the server determines the society turnover. The norms allows to define a "legal representative" for each person in charge (including one representative for the technical aspect and one representative for the juridic aspect).

Question 3:

Propose an evolution of your solution to allow the representative to substitute, if needed, the persons in charge when (re)booting the service server. The system can't rely on the confusion between a person in charge and its representative. At the contrary, it have to be able to discriminate between them without changing the services functioning.

Question 4:

Modify your services if needed.

Part 3: Right revocation

Because it is possible to dismiss a person in charge, the authentication mechanism should be able to handle it.

Question 5:

Propose a new repudiation service 1.vi. which allows the server to remove the responsibility of someone in charge or its representative. This repudiation can be only if the persons in charge (except the removed one) are authenticated, and must be impossible *by design* without them.

Question 6:

Implement a repudiation service 1.vi. according to the proposition made in the question 5.