Identity Card Key Generation in the Malicious Card Issuer Model

Arnis Parsovs

May 27, 2014

Abstract

Generally accepted practice where the electronic identity card issuer generates private key on behalf of the cardholder allows for a malicious card issuer to compromise cardholder’s private key security. In this report we discuss several solutions for improving cardholder’s private key security against malicious card issuer.

1 Introduction

Governments in several countries around the world issue Electronic Identity Cards (eID) to their residents. These cards are smart cards which usually contain RSA key pairs that are bound to resident’s identity and allow the resident to authenticate in online services, sign electronic documents with a digital signature, and perform encrypted communication between residents.

The security of electronic identity depends on cardholder being the only one who can use the private key which has been bound to his identity. Therefore, the private key should be kept in a single copy inside a tamper resistant smart card. The private key operations should be performed only after the cardholder has authenticated himself to the smart card using symmetric PIN code shared only between the cardholder and smart card.

In practice the card issuer (usually the government or its delegated party) provides cardholders with fully personalized smart cards that contain RSA key pair, X.509 public key certificate and PIN codes generated for the cardholder.

By applying security threat modeling for smart cards [1] we see that since the card issuer in this case is not the data owner, the involvement of the issuer in providing security for cardholder’s private key opens up a possibility for malicious card issuer to launch attacks against cardholder.

The designers of eID presuppose that the card issuer holds the best interests of the cardholder, however, this might not necessarily be the case and as we know from security economics, where the party who is in a position to protect a system is not the party who would suffer the results of security failure, then problems may be expected [2].
One such problem was recently found in Taiwanese eID, where the hardware random-number generator built in the smart card was fatally flawed, allowing anyone to recover private keys stored on these cards [3].

In this report we will discuss possible solutions that could provide cardholder’s private key security even in case of careless or malicious card issuer. To simplify analysis we will assume that the card issuer, card manufacturer, card software manufacturer and CA certifying cardholder’s key is the same, possibly malicious party.

2 Attacks by the Issuer Against the Cardholder

The most straightforward attack against the cardholder is to compromise the secrecy of cardholder’s private key. Since the card issuer is the one who produces smart card and generates the keys for cardholder, compromising cardholder’s private key is trivial. In the simplest form of attack the issuer simply saves a copy of cardholder’s private key before loading it into the smart card.

To avoid such accusations, the card issuers usually claim that the key pair is generated on card using built-in secure hardware random number generator and that there is no way for private key to ever leave the smart card. These claims are usually attested by independent certification, such as Common Criteria or FIPS 140.

Unfortunately, there is no way for anyone else other than the card issuer to make sure that the chip and software that has passed the certification is the one that is used in the card. Furthermore, as we can learn from Taiwanese eID case, the certified card may allow insecure key generation with “FIPS mode” disabled [3] and only the issuer will know whether it is the case.

It is possible to argue that we have to trust the issuer anyway, since a malicious issuer can always impersonate the cardholder even without compromising his private key, by unlawfully issuing and using certificate in the name of the cardholder. However, the use of mis-issued certificate can be traced back to CA with cryptographic precision, while there is no way to trace back who holds a copy of the private key which has been abused. Under the European Directive 1999/93/EC [4, Article 6] the CA is liable for actions performed with mis-issued certificate, while the cardholder is liable for all actions performed with his private key. Therefore, while a malicious card issuer can indeed temporarily impersonate the cardholder, the eventual discovery of misuse is deterring the card issuer from mis-issuing certificates, while there is nothing that deters the issuer from abusing cardholder’s private keys.

3 Possible Solutions

3.1 Private key generated by the cardholder

The simplest solution would be to completely remove issuer’s role in cardholder’s private key generation and storage, making it a sole responsibility of cardholder
(as it is common practice when issuing server certificates). The cardholder could generate private key using his own media and include the corresponding public key in application form when applying for the certificate. This, however, would be too complicated for most of the cardholders and private keys would end up being stored in unprotected media which would be against our original intention to improve private key security. In fact, the European Directive 1999/93/EC requires signature-creation-data to be stored in secure signature-creation device, which must ensure that the signature-creation-data can practically occur only once and signature generation can be reliably protected by the legitimate signatory against the use of others [4, ANNEX III]. Therefore, storing private key somewhere else than in the tamper resistant device would be against the directive.

A basic improvement would seem to ship the smart card without private keys loaded, but the cardholder initiating the key generation on the first eID use. This would guarantee that the keys are generated on card. However, a malicious issuer can make the random number generator predictable (for example, by seeding PRNG with card’s serial number) or even generate RSA key pair such that the private key can be efficiently reconstructed knowing only the public key [5].

3.2 Use of two keys

Desirable security level against a malicious card issuer would be achieved if the cryptographic operations performed by the cardholder would require use of two keys – one generated and stored on smart card produced by the issuer, and another generated and stored on any medium by the cardholder. The digital signature would be considered valid only if data has been signed using both keys. Similarly, for hybrid encryption the symmetric transport key could be split into two parts using secret sharing and parts encrypted using the distinct public keys. This would prevent malicious issuer from abusing the private key that has been generated for the cardholder and would protect the cardholder if anyone (except the issuer) has compromised his carelessly stored private key.

While optimal, the solution is not practical, since it requires significant changes in user interface, standards, and protocols as they are used today.

3.3 Threshold RSA

The same security benefits as in two key use scenario can be achieved using a single key, but performing key generation and private key operations in a distributed manner. In this scenario, the RSA public key would be generated by issuer’s smart card and cardholder’s device, while no single device could perform private key operations without collaborating.

RSA can be generated on average in 15 minutes on Intel Core i5 dual core 2.3 GHz. Unfortunately, if one of the parties is a smart card with low computational resources, the estimated generation time would probably reach several days, which makes this approach undeployable in practice.

Even if it would be efficient, the solution still requires the private key share to be stored on cardholder’s computer which creates portability issues and risks eID to become unusable if the cardholder’s computer is reinstalled.

### 3.4 Abuse-free RSA key generation

In this solution the private key is fully stored on the smart card provided by the card issuer. However, the private key is generated in abuse-free way in cooperation with cardholder’s computer such that neither the issuer nor cardholder can learn the private key.

In this section we will discuss several protocols which might be used to achieve it, however, it is important to note that this approach compared to previous solutions does not give the same protection against malicious issuer.

A malicious issuer can deliberately weaken the smart card or implement backdoors that will leak the private key through various side channels. In the simplest form of an attack the smart card can simply return cardholder’s private key to the terminal if the terminal sends a special command to the card. Fortunately, these attacks require the card issuer to be in direct contact with the card after the key has been generated. In current eID use case the terminal is usually owned by the cardholder (terminal being cardholder’s card reader and computer) and the card is rarely inserted in a terminal that is under card issuer’s control.

A malicious smart card could also try to use a signature as a covert channel to leak the private key. However, RSA PKCS#1 v1.5 signing scheme uses deterministic padding thus the only place where to encode the private key would be the message itself, which would make the signature verification of the signed document to fail.

Therefore, while the abuse-free RSA key generation approach does not protect against malicious issuer who can obtain direct access to the card later in the smart card lifecycle, the use of this approach will eliminate risk of private key compromise that is caused by intentional or unintentional randomness flaw in the smart card. This will require a malicious issuer to carry out more sophisticated attack, increasing the conspiracy level required and thus the risk of being caught.

---

1 One such case is PIN escrow mechanism provided by issuer which allows the issuer to reset card’s PIN codes in case the cardholder has forgotten them. The procedure is performed after identity verification at issuer’s customer service points and the reset is performed in encrypted, mutually authenticated session between the issuer and the smart card.

2 Less popular padding scheme RSA-PSS uses random padding, which could be used by a smart card to leak the private key.
3.4.1 Threshold RSA

We can obtain abuse-free key generation right away by using protocols described in threshold RSA solution, since by definition the threshold RSA generates private key in abuse-free manner. After the key is generated using threshold RSA protocol, the cardholder’s computer simply sends his private exponent share to the smart card.

In the following subsections we will discuss abuse-free key generation approaches that do not require distributed private key operations and therefore can be more efficient than two party threshold RSA protocols.

3.4.2 Verifiable randomness

In these protocols the smart card would prove to the cardholder that the randomness received from the cardholder was mixed with the smart card’s randomness to generate unpredictable RSA factors. The first generic protocol was suggested by Desmedt in [10]. There, the cardholder’s computer would send randomness to the smart card, and smart card using commitments and zero-knowledge proofs would prove that cardholder’s computer’s randomness was used to generate RSA modulus in abuse-free way.

Specific protocol was proposed by Juels-Guajardo in [11]. The work [12] estimates that Juels-Guajardo protocol would likely take over 40 minutes to execute on the home router, which likely will take several days on low performance smart card, making the protocol impractical for our purpose.

3.4.3 Multi-prime RSA

In this protocol proposed by the author the cardholder’s computer and smart card will generate 4096-bit 4-prime balanced RSA modulus, which will be constructed from 4 1024-bit prime factors ($p_1, q_1, p_2, q_2$). The card issuer will ship the smart card containing $p_1, q_1$ and $e = 65537^3$. On the first eID use, the following protocol will be executed between the cardholder’s computer and smart card.

1. The smart card will send $p_1 * q_1$ (or it’s commitment) to cardholder’s computer.

2. The cardholder’s computer will generate $p_2, q_2$ (such that $gcd(p_2 - 1, e) = 1$ and $gcd(q_2 - 1, e) = 1$) and send it to the smart card.

3. The card will calculate $n, d$ and will return $n$ to the computer (and reveal the committed value from step 1).

4. The cardholder’s computer will verify whether the $p_2 * q_2$ divides $n$ without remainder (to make sure that the modulus contains cardholder’s factors)

---

3 Public exponent $e$ is fixed to prevent it being used as a covert channel.
4 Note, that this does not require any interaction from the cardholder.
5. The cardholder’s computer will verify whether the \( \frac{n}{p_2 \cdot q_2} \) is equal to \( p_1 \cdot q_1 \) received in the first step (to make sure that smart card’s factors \( p_1, q_1 \) do not depend on cardholder’s \( p_2, q_2 \)).

6. The cardholder’s computer will send \( n \) to the card issuer.

7. The issuer will verify whether the \( p_1 \cdot q_1 \) (stored in issuer’s database) divides \( n \) without remainder (to make sure that the modulus contains issuer’s factors – this prevents cardholder from obtaining certificate for private key that does not reside on smart card).

8. The issuer returns X.509 certificate to the cardholder’s computer.

9. The cardholder’s computer loads X.509 certificate into the smart card.

As can be seen from the protocol, the card issuer is unable to learn factorization of \( p_2 \cdot q_2 \), and in his turn the cardholder is not able to learn factorization of \( p_1 \cdot q_1 \). Therefore, for malicious issuer, cardholder or anyone who has compromised cardholder’s computer (except the issuer) the effective security of private key is equivalent to 2048-bit RSA, while for all other attackers the private key has 4096-bit RSA security [13, Section 2].

The protocol involves only few simple calculations, however, the drawback being 4096-bit key size. Compared to 2048-bit RSA, the 4096-bit RSA public key operations are approximately 4 times slower, while private key operations are only 2 times slower if the Chinese remainder theorem is used.

4 Conclusion

In this report we have discussed several solutions that could improve cardholder’s private key security against malicious card issuer. While practically deployable solutions do not provide full security against malicious issuer, they eliminate private key compromise caused by unintentional or intentional random number generator flaws and thus require more sophisticated attack by the card issuer.

References


