A Better Use of Smart Cards in PKIs
(Extended Abstract)*

[Published in *Gemplus Developer Conference*, Singapore,
November 12–14, 2002.]

Nathalie Feyt and Marc Joye

Gemplus Card International, Card Security Group
Parc d'Activités de Gémenos, B.P. 100, 13881 Gémenos Cedex, France
{nathalie.feyt, marc.joye}@gemplus.com
http://www.gemplus.com/smart/ – http://www.geocities.com/MarcJoye/

Keywords. Smart cards, public-key infrastructure (PKI), on-board key generation, RSA cryptosystem.

1 Public-Key Infrastructures

Public-key cryptography faces the problem of the authentication of the public keys: How do we can be sure that a pair of public key/user's identity are matching? A related problem is how to distribute public keys trustfully. These issues are proved to be the bottleneck of a wide deployment of public-key systems, such as the RSA cryptosystem. It is here the Public Key Infrastructures (PKIs) come into play [PKIX]. The idea behind PKI is fairly simple. It basically consists in producing an analogue of the phone directory. In the 'PKI directory', one should be able to find a user (or more generally an application) and the corresponding public key. Of course, this directory must in some sense be certified. To this purpose, in addition to the name and the public key, the directory also contains a certificate issued by a Certification Authority (CA). Furthermore, in order to make the system inter-operable, each user belongs to a domain and each domain has its own associated certification authority. Then, when the user (or the application) has to be identified and authenticated, he just produces the certificate issued by the CA of his domain. This certificate is a digital signature by the CA on at least the user's public key and his identity (along with some other credentials, if needed).

At present, it is common that, when smart cards hold public keys, for each embedded public key, a companion certificate is issued by a CA. A certificate has a cost, even if the corresponding public key is never used by the card holder. A cheaper solution may be to have an on-board key generation. So, sets of keys will be generated only if they will be used. A second advantage is that there is more EEPROM memory available. Furthermore, we should note that on-board

* The full paper will appear in [FJP].

key generation is more *secure* as the private keys are only known by the card holder, i.e., the end user. Although attractive, this second solution may be too slow for certain applications. This paper is aimed at presenting a mixed solution, resulting in a very fast *on-board* key generation.

2 The Case of RSA

RSA [RSA78], named after its inventors Rivest, Shamir and Adleman, is the most largely deployed public-key cryptosystem. It can be used for both public-key encryption [BR95] and digital signatures [BR96]. The security of RSA relies on the difficulty of factoring large integers, typically 1024- or 2048-bit moduli.

A set of RSA keys is a pair of matching private/public keys. Basically, a private RSA key consists of two secret large primes p and q satisfying the additional property that $\gcd(p-1,e) = \gcd(q-1,e) = 1$ where e represents the public exponent, and of the private exponent $d \equiv e^{-1} \pmod{\operatorname{lcm}(p-1,q-1)}$. The corresponding public RSA key is e and $N = pq$. We denote by ℓ the bit-length of N. Private exponent d is used for decrypting or signing messages and public exponent e is used for encrypting a message or for verifying a signature, respectively.

The RSA key generation consists, on input a pair (ℓ, e), in outputting a pair (N, d) such that $N = pq$ for two random primes p and q, $|N|_2 = \ell$, $\gcd(p-1,e) = \gcd(q-1,e) = 1$ and $d \equiv e^{-1} \pmod{\operatorname{lcm}(p-1,q-1)}$. Using the very efficient algorithm of [JP02] (see also [JPV00]), the generation a 1024-bit RSA modulus N along with d only requires a few seconds on recent smart cards. Although such timings are competitive, they may appear too slow in certain protocols, for example when the process is operated on-line.

2.1 First solution

An obvious way for addressing the key generation is to store several sets of RSA keys when the card is issued. However, such an approach is not satisfactory: If, in order to be functional, an application requires a pair (ℓ, e) different from those pre-stored in the card, the application will be inaccessible to the user. Moreover, the keys require memory to be stored. Finally, and more importantly, the card issuer may have the knowledge of the user's private keys. *The card holder has no guarantee whatsoever that the entity producing the keys does not keep a copy of his private keys at the initialization stage.*

2.2 Second solution

A second solution consists in letting the cryptographic device compute on-board a valid pair (N, d) from the input (ℓ, e) required by the application. As already mentioned, the drawback in this second approach is the running time which may appear to be excessive in certain scenarios (e.g., mobile applications). However, now the card issuer does not know the private keys since they are randomly computed by the card.

3 Our Proposal

The on-board key generation is preferable as the keys are not 'imposed' by the card issuer. Building on [JP01], we devised a new method for generating RSA keys. Contrary to what is actually done, we propose to divide the process in two phases. The first phase is performed off-line, before the values of (ℓ, e) are even known. The second phase is performed on-line by the cryptographic device once (ℓ, e) are known, and is supposed to be *very fast*.

We only sketch our method and refer the reader to [FJP] for more detail. The core of our solution relies on the following algorithm which computes an RSA ℓ_0-bit prime q:

Input: parameters ℓ_0, e, and a (of large order) in $\mathbb{Z}_\Pi{}^*$
Output: a prime $q \in \left[\lceil 2^{\ell_0 - 1/2} \rceil, 2^{\ell_0} - 1\right]$

---

1. Compute $v = \left\lceil \frac{\lfloor 2^{\ell_0 - 1/2} \rfloor}{\Pi} \right\rceil$ and $w = \left\lfloor \frac{2^{\ell_0}}{\Pi} \right\rfloor$
2. Randomly choose $j \in_R \{v, \ldots, w-1\}$ and set $l \leftarrow j\Pi$
3. Randomly choose $k \in_R \mathbb{Z}_\Pi{}^*$
4. Set $q \leftarrow k + l$
5. If (q is not prime) or ($\gcd(e, q-1) \neq 1$) then
   (a) Set $k \leftarrow ak \pmod{\Pi}$
   (b) Go to Step 4
6. Output q

Fig. 1. RSA Prime Generation Algorithm.

We see that a prime produced by our algorithm has the form $q = a^{f-1}k_{(0)} \bmod \Pi + j\Pi$ where $k_{(0)}$ denotes the initial value of k and f is the number of failures of the test in Step 5. To drastically reduce the memory requirements, each ℓ_0-bit prime q may be stored as a pair (i, f) where i is a unique indexed identifier (i is used as the input of a pseudo-random generator for constructing $v, w, j, k_{(0)}$). This phase is performed off-line. Then the on-line stage consists in reconstructing primes from pairs (i, f).

Remarkably, in addition to be fast, our method allows the on-line generation of RSA moduli $N = pq$ of *arbitrary* length from a very small set of values computed at the off-line phase. Furthermore, parameters can be chosen so that the keys are guaranteed to work for the usual public exponents $e \in \{3, 17, 2^{16}+1\}$ (or in any other set of exponents chosen at the off-line stage), covering most of the applications available on the market. We note, however, that the choice of the public exponent is usually let at the discretion of the user.

References

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In A. De Santis, editor, *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1995.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - How to sign with RSA and Rabin. In U. Maurer, editor, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer-Verlag, 1996.

[FJP] Nathalie Feyt, Marc Joye, and Pascal Paillier. Off-line/On-line on-board generation of RSA keys. To appear in *2nd International Workshop for Asian Public Key Infrastructures*, Taipei, Taiwan, October 30–November 1, 2002.

[JP01] Marc Joye and Pascal Paillier. Constructive methods for the generation of prime numbers. In S. Murphy, editor, *2nd Open NESSIE Workshop*, Egham, UK, September 12–13, 2001.

[JP02] Marc Joye and Pascal Paillier. Efficient generation of RSA keys in portable devices. April 2002. Preprint.

[JPV00] Marc Joye, Pascal Paillier, and Serge Vaudenay. Efficient generation of prime numbers. In Ç.K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 340–354. Springer-Verlag, 2000.

[PKIX] Public Key Infrastructure (X.509) series, http://www.ietf.org/html.charters/pkix-charter.html.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.