Fully Distributed Threshold RSA under Standard Assumptions

Pierre-Alain Fouque and Jacques Stern

École Normale Supérieure, Département d'Informatique
45, rue d'Ulm, F-75230 Paris Cedex 05, France
{Pierre-Alain.Fouque,Jacques.Stern}@ens.fr

Abstract. The aim of the present article is to propose a fully distributed environment for the RSA scheme. What we have in mind is highly sensitive applications and even if we are ready to pay a price in terms of efficiency, we do not want any compromise of the security assumptions that we make. Recently Shoup proposed a practical RSA threshold signature scheme in [17] that allows to share the ability to sign between a set of players. This scheme can be used for decryption as well. However, Shoup's protocol assumes a trusted dealer to generate and distribute the keys. This comes from the fact that the scheme needs a special assumption on the RSA modulus and this kind of RSA moduli cannot be easily generated in an efficient way with many players. Of course, it is still possible to call theoretical results on multiparty computation, but we cannot hope to design efficient protocols. The only practical result to generate RSA moduli in a distributive manner is Boneh and Franklin protocol [2] but this protocol cannot be easily modified to generate the kind of RSA moduli that Shoup's protocol requires.
The present work takes a different path by proposing a method to enhance the key generation with some additional properties and revisits the proof of Shoup to work with the resulting RSA moduli. Both of these enhancements decrease the performance of the basic protocols. However, we think that in the applications that we target, these enhancements provide practical solutions. Indeed, the key generation protocol is usually run only once and the number of players used to sign or decrypt is not very large. Moreover, these players have time to perform their task so that the communication or time complexity are not overly important.

1 Introduction

In this paper we propose new techniques to protect a shared RSA secret key from generation phase to applications. This solves an open problem where one needs to cope with requirements that do not match. On one hand, at Eurocrypt'00, Shoup describes a practical threshold signature scheme which requires an RSA modulus with specific properties : namely, the prime numbers of the modulus should be safe primes. On the other hand, Boneh and Franklin at Crypto '97 describe a protocol to share an RSA modulus. However, the latter cannot be used

to generate a safe modulus. The distributed generation of safe moduli seems to be hard and the method of Boneh and Franklin cannot be easily adapted to this context. The present work takes a different path by proposing a method to enhance the key generation with some additional properties and revisits the proof of Shoup to work with the resulting RSA moduli.

Shoup threshold RSA signature scheme [17] presents interesting features. First of all, it is secure and robust in the random oracle model assuming the RSA problem is hard. Next, the signature share generation and verification are completely non-interactive and finally, the size of an individual signature share is bounded by a constant times the size of the RSA modulus. However, this scheme requires a trusted dealer to generate the keys and distribute the shares of the secret key among the ℓ servers.

When a message m has to be signed by a quorum of at least $t+1$ servers, where $2t+1 \leq \ell$, a special server, called the *combiner*, forwards the message m to all servers. Then, each server computes its signature share along with a proof of correctness. Finally, the combiner selects a subgroup of $t+1$ servers by checking the proofs and combines the $t+1$ related signature shares to generate the signature s. A key point in this scheme is the *proof of correctness*, which guarantees the *robustness of the scheme*. Robustness means that corrupted servers should not be able to prevent uncorrupted servers from signing. This property is attractive for threshold protocols, but in this scheme, the proof of correctness requires an RSA modulus built with safe primes.

This raises the question of generating RSA moduli for use in Shoup's threshold scheme without a trusted dealer. There exist protocols that generate RSA keys in a distributive manner. Boneh and Franklin in [2] designed such protocol for the generation of an RSA modulus in the honest-but-curious model. Later, Frankel, MacKenzie and Yung in [7] made this algorithm robust against malicious servers. In [13], Poupard and Stern also provided a protocol to compute a shared modulus for two players only. Finally, Gilboa in [10] has extended Poupard and Stern method. As we can note, the Boneh and Franklin protocol is most efficient but no protocol is known to create shared safe RSA moduli.

It would be useful to fully distribute the RSA protocols from key generation to signature. To avoid the generation of shared safe moduli, which appears currently out of reach, this paper proposes a tradeoff between the requirements of the RSA modulus for the signature and decryption protocols and the requirements at key generation.

Independly of our work, Damgard and Koprowski [5] have considered the same problem in a recent paper. They revisited Shoup's paper and used a non-standard assumption to show that the proof of correctness works with an RSA modulus.

In our work, we consider environments where high security is required such as electronic voting schemes. Therefore, we prefer to use protocols based on standard assumptions and we are ready to pay the price. We believe that standard assumptions and security proofs are needed to build secure protocols.

1.1 Outline of the paper

In the first part, we present the problem and in section 3 the security model that we use. Next, in section 4, we describe how to enhance the Boneh-Franklin scheme to generate RSA moduli having special properties and in section 5 we present the new proof of correctness making Shoup scheme robust without using safe primes. Finally, in section 6 we present practical parameters for our scheme.

1.2 Notations and Definitions

Throughout this paper, we use the following notation: for any integer $N = pq$, where $n = \log(N)$ is a security parameter, as well as k, k_1 and k_2,

- we use Q_N to denote the group of square in ${\mathbb{Z}_N}^*$,
- we use $\varphi(N)$ to denote the Euler totient function, i.e. the cardinality of ${\mathbb{Z}_N}^*$,
- we use $\lambda(N)$ to denote Carmichael's lambda function defined as the largest order of the elements of ${\mathbb{Z}_N}^*$.

Let $p = 2p' + 1$ and $q = 2q' + 1$ where in general $p' = \prod_{p_i} {p_i}^{e_i}$ and $q' = \prod_{q_j} {q_j}^{e_j}$. Set $M = p'q'$.

Finally, a prime number p is a *safe prime* if p and p' are both prime.

2 Background

2.1 Where is the problem ?

Robustness guarantees that even if t malicious players send false signature shares, the scheme still correctly generates a signature s. This property is needed since otherwise combination faces the problem of selecting the correct shares.

One can note that this problem may seem more acute for the decryption process than the signing process, but a solution to this problem is required for both cases. For example, the combiner receives signature shares from the servers and has to generate the correct signature. One way for him is to pick at random $t+1$ signature shares, to generate the possible signatures s' and to test whether s' is a valid signature of m. If it is correct, the correct signature has been found, otherwise, the combiner has to test another group of $t+1$ signature shares. Since the combiner cannot guess where the bad shares are, it might face an exponential number of trials.

In the decryption process, the combiner cannot even test whether the decryption is correct or not. Therefore, it is necessary to devise an efficient test in order to check whether a player has correctly answered a request. Shoup has proposed an efficient proof to achieve such check and the same kind of proof appears in [15, 8] but still requires safe prime modulus.

At the same time, known methods to jointly generate an RSA modulus cannot be easily adapted to generate a safe prime modulus. Therefore our approach is to put additional criteria at key generation and modify the proof of correctness to work with the resulting RSA modulus.

2.2 Our results

The proof of "correctness" described in Shoup's protocol uses two important properties of the subgroup Q_N of squares of ${\mathbb{Z}_N}^*$ when N is a safe modulus. On one hand, this subgroup is cyclic and on the other hand, its order M does not have small prime factors. The cyclic group is used to show the **existence of the discrete log** in the proof of correctness. The use of safe primes allows to guarantee that, with overwhelming probability, a random element in Q_N is a **generator**.

Our first observation relates the structure of Q_N with $\gcd(p-1, q-1)$ and the search for generators in this group to the prime factor decomposition of $\frac{p-1}{2}$ and $\frac{q-1}{2}$. In particular, if $\frac{p-1}{2}$ and $\frac{q-1}{2}$ have no small prime factors, then with high probability few randomly chosen elements generate the entire group Q_N. Moreover, using a nice trick of Gennaro *et al.* which first appeared in [9] and the protocol recently proposed by Catalano *et al.* in [4], the calculation of $\gcd(p-1, q-1)$ can be performed in a distributed way. This method allows to keep key generation efficient.

In this paper, we show how to jointly construct RSA moduli such that the subgroup Q_N is cyclic, which guarantees the existence of discrete logs and of generators of Q_N. Moreover, the order M of this group does not have small prime factors less than some sieving bound B. Checking such primes does not exceedingly increase the running time of the key generation algorithm.

There is still a difficulty : contrary to the case of safe prime modulus, picking at random an element in Q_N does not necessarily provide a generator of Q_N with high probability. To overcome the difficulty, we choose enough random elements $(g_1, \ldots, g_k)$ to guarantee that the group generated by $\langle g_1, \ldots, g_k \rangle$ is all of Q_N with high probability. Such techniques have already been used by Frankel *et al.* in [7] and a precise treatment has been given by Poupard and Stern in [14]. What the paper shows is that the overhead can remain at a practical level as well.

2.3 Previous works

As already pointed out, another solution to the same problem has been proposed by Damgard and Koprowski. Besides being based on standard assumptions, our method is different in that :

1. we make the group of squares cyclic,
2. we build p and q such that $\frac{p-1}{2}$ and $\frac{q-1}{2}$ do not contain small prime factors,
3. we use a set of generators of Q_N found by picking at random few elements in Q_N.

3 Security Model

3.1 The Network

We assume a group of ℓ servers connected to a broadcast medium, and that messages sent on the communication channel instantly reach every party connected to it.

3.2 Formal definition

A RSA threshold signature scheme consists of the four following components :

- A *key generation algorithm* takes as input security parameters n, k, the number ℓ of signing servers, the threshold parameter t and a random string ω; it outputs a public key (N, e) where n is the size in bits of N, the private keys $d_1, \ldots d_\ell$ only known by the correct server and for each $u \in [1, k]$ a list $v_u, v_{u,1} = {v_u}^{d_1}, \ldots v_{u,\ell} = {v_u}^{d_\ell} \bmod N$ of verification keys.
- A *share signature algorithm* takes as input the public key (N, e), an indices $1 \leq i \leq \ell$, the private key d_i and a message m; it outputs a signature share $s_i = H(m)^{d_i} \bmod N$, where $H(.)$ is a hash-and-pad function, and a proof of its validity $proof_i$.
- A *combining algorithm* takes as input the public key (N, e), a message m, a list $s_1, \ldots s_\ell$ of signature shares, for each $u \in [1, k]$ the list $v_u, v_{u,1}, \ldots v_{u,\ell}$ of verification keys and a list $proof_1, \ldots proof_\ell$ of validity proofs; it outputs a signature s or fails.
- A *verification algorithm* takes as input the public key (N, e), a message m, a signature s; it outputs a bit b indicating whether the signature is correct or not.

3.3 The players and the scenario

Our game includes the following players : a dealer, a combiner, a set of ℓ servers P_i, an adversary and users. All are considered as probabilistic polynomial time Turing machines. We consider the following scenario :

- At the initialization phase, the servers use the distributed key generation algorithm to create the public, private and verification keys. The public key (N, e) and all the verification keys v, v_i are published and each server obtains its share d_i of the secret key d.
- To sign a message m, the combiner first forwards m to the servers. Using its secret key d_i and its verification keys $v, v_{u,i}$ for $u \in [1, k]$, each server runs the share signature algorithm and outputs a signature share s_i together with a proof of validity of the share signature $proof_i$. Finally, the combiner uses the combining algorithm to generate the signature, provided enough signature shares are available and valid.

3.4 The adversaries

We consider an adversary able to corrupt up to t out of the ℓ servers. Such a corruption can be passive, *i.e.* the attacker only eavesdrops the servers. It can also consist in making the servers fail and stop. Finally, it can be active; in this case, the adversary completely controls the behavior of the corrupted servers. In the following, we only consider non-adaptive adversaries who choose the servers they want to corrupt before key generation.

3.5 Properties of threshold signature schemes

The two properties of a t out of ℓ threshold signature scheme of interest to us are robustness and unforgeability. As we already mentioned, robustness guarantees that even if up to t malicious players send false signature shares, the scheme still returns a correct signature.

Unforgeability guarantees that any subset of $t+1$ players can generate a signature s, but disallows the generation by fewer than t players. This unforgeability property should hold even if some subset of less than t players are corrupted and collude.

3.6 The Games

In this section, we describe the security of the distributive version of the key generation protocol of our scheme. We do not describe the game for the unforgeability of the signature scheme as the proof remains identical with what appears in [17]. We have to show that the public information revealed during execution does not release any information to the adversary.

Game for the distributive version of key generation.
The correctness of the key generation requires that the probability of the secret keys d, p, q, and the public key (N, e) seem be uniformly distributed to the adversary.
The secrecy of the key generation means that if there exists an adversary $\mathcal{A}$ which corrupt at most t servers at the beginning of the game, then he cannot obtain more information on the secret key held by uncorrupted players.

Remarks We can note that we do not show that the information learned during the key generation protocol does not help the adversary to decrypt or to sign the message. We assume here a stronger adversary who can factor the modulus N as Boneh and Frankel do.

4 Enhancing the Boneh-Franklin scheme

The aim of this section is to generate RSA moduli such that the group of squares is a cyclic group whose order has no small prime factors. In the following section, we will prove that this group can be generated with few random elements. In this section, we use a sieving method to simultaneously improve the key generation protocol and the probability of finding a set of generators of Q_N. Next, we present a protocol to compute the GCD of a known value and a shared value. We also prove the robustness and the secrecy of this new distributed key generation protocol.

4.1 Distributed RSA key Generation

In [2] Boneh and Franklin describe a protocol for generating a shared RSA modulus. We describe this protocol and show how to generate the verification keys that we need as well.

1. In the first step, each server picks at random two values p_i and q_i in the interval $\{\sqrt{2}.2^{n/2-1}, \ldots, \lfloor\frac{2^{n/2}-1}{\ell}\rfloor\}$ according to [18], where n is the size in bits of the modulus N. Then, we use a sieving algorithm in order to discard $p_1 + \ldots + p_\ell$ and $q_1 + \ldots + q_\ell$ that have small prime factors and to also discard $p_1 + \ldots + p_\ell$ and $q_1 + \ldots + q_\ell$ if $p_1 + \ldots + p_\ell - 1$ or $q_1 + \ldots + q_\ell - 1$ have small prime factors.
2. Then the BGW protocol [1, 2] is run to compute the product N of $p_1 + \ldots + p_\ell$ and $q_1 + \ldots + q_\ell$.
3. Next, the parties perform a primality test similar to the Fermat test modulo N. The practicality of this test is based on the empirical results of [16] where Rivest showed that if sieving is performed, the Miller-Rabin primality test is not needed as pseudoprimes are rare according to Pomerance's conjectures [11,12]. We set $p = p_1 + \ldots + p_\ell$ and $q = q_1 + \ldots + q_\ell$.
4. Finally, we use another protocol described below in section 4.2 to compute the inverse of the public key modulo $\varphi(N)$. Each server knows its share d_i of the secret key d, then he computes and publishes $v_{u,i} = {v_u}^{d_i} \bmod N$ for all verification keys v_u. All other values are erased.

4.2 Computing the gcd of a public value and a shared secret value

We briefly recall the protocol presented by Catalano *et al.* [4] for inverting a public value e modulo a shared value φ. The basic trick stems from the observation that $\gcd(e, \varphi) = \gcd(e + R\varphi, \varphi)$ where R is a large integer used to mask the shared and secret value $\varphi = \varphi_1 + \ldots + \varphi_\ell$. Server i chooses a random integer $r_i \in_R [0..2^{n+k'}]$, where k' is a security parameter, computes $c_i = \varphi_i + er_i$ and forwards c_i to all other servers. Each server can compute $c = \sum_i c_i = \varphi + eR$ if we set $R = \sum_i r_i$. This value can be publicly known and then, all servers can compute $\gcd(e, c)$ which is equal to $\gcd(e, \varphi)$ and u and v such that $eu + cv = 1$ when $\gcd(e, \varphi) = 1$. Then, it is easy to show that if we replace c by $\varphi + eR$, we obtain $e(u + Rv) + \varphi v = 1$. Hence, $u + Rv$ is the inverse of e modulo φ. In this case, if we note d the inverse of e mod φ, each server assigns its share of the inverse d to $d_i = vr_i$ and the first server to $d_1 = u + vr_1$.

We have presented here the protocol in the honest-but-curious model. But this protocol can be made robust following [4]. We can also note that this algorithm allows to compute the gcd.

4.3 Efficient sieving algorithm improving the generation of random number without small factors

Here we want to show how to generate N such that $p' = \frac{p-1}{2}$ and $q' = \frac{q-1}{2}$ have no small prime factors. Our method uses a new distributed sieving protocol designed by Boneh, Malkin and Wu in [3] that we patch in order to create p such as neither p, nor p' has a small prime factor less than B. Moreover, we show how to withstand malicious adversaries. We denote by P the product of all odd small primes up to B.

1. Each server picks a random integer a_i in the range $[1, \ldots, P]$ such that a_i is relatively prime to P. Then, since each a_i is a random integer relatively prime to P, their product $a = a_1 \times \ldots \times a_\ell \bmod P$ is also relatively prime to P.
2. The servers perform a protocol to convert the multiplicative sharing of a to an additive sharing of $a = b_1 + \ldots + b_\ell$ using the BGW protocol.
3. Each server picks a random $r_i \in [0, \frac{2^n}{P}]$ and sets $p_i = r_i P + b_i$.

Clearly, $p = \sum p_i \equiv a \bmod P$ and hence p is not divisible by any prime smaller than B. We can note that $p = RP + a$ where $R = \sum_i r_i$.

In order to also prove that $p' = \frac{p-1}{2}$ has no prime factors less than B, one has to check whether $\gcd(p-1, P) = \gcd(2p', P) = 1$ and $\gcd(p-1, 4P) = 2$ to test the power of 2. If we denote P' by $4P$, we can perform a single test $\gcd(p-1, P') = 2$.

To distribute this test in the honest-but-curious model, the first server sets its parts p_1 to $p_1 - 1$ and we use the distributed GCD protocol described in section 4.2.

It is also possible to make this test robust in presence of malicious players. To resist against such players, we first run a "sum-to-poly" algorithm as described in [6, 7]. When the polynomial sharing of p is obtained, one can note that $\sum_{j \in S \setminus \{0\}} \lambda_{0,j}^S = 1$, where $\lambda_{0,j}^S$ denotes the Lagrange coefficient of the jth server. This follows from the fact that if one shares with Shamir's Sharing Secret Scheme, the constant polynomial equal to 1, the value of this polynomial in 0 is also 1. Therefore, server i can set its new polynomial share to $f(i) - 1$ if $f(i)$ denotes its polynomial share. Indeed, if $p = f(0) = \sum_{j \in S \setminus \{0\}} \lambda_{0,j}^S f(j)$, then

$$p - 1 = f(0) - 1 = \sum_{j \in S \setminus \{0\}} \lambda_{0,j}^S f(j) - \sum_{j \in S \setminus \{0\}} \lambda_{0,j}^S = \sum_{j \in S \setminus \{0\}} \lambda_{0,j}^S (f(j) - 1)$$

Next, the GCD protocol can also be applied with a polynomial sharing of the secret value $\varphi = p - 1$.

Finally, the protocol that transforms the multiplicative sharing of a into an additive sharing can also be made robust as it uses the BGW protocol. This transformation calls ℓ times the BGW protocol. At the beginning, $b_{i,0} = 0$ for all $i \in \{0, \ldots, \ell\}$. Then, for $i = 1$ to ℓ, $u_i = a_i$ and $u_j = 0$ for $\forall j \neq i$, and the BGW protocol performs $(b_{1,i-1} + \ldots + b_{\ell,i-1}) \times a_i = (b_{1,i-1} + \ldots + b_{\ell,i-1})(u_1 + \ldots + u_\ell) = b_{1,i} + \ldots + b_{\ell,i}$.

Theorem 1. *The key generation protocol of Boneh-Franklin and the sieving protocol allow to generate RSA moduli such that the order M of the group Q_N does not contain small prime factors less than B.*

It is obvious to see that the use of the sieving method to guess p_i's and q_i's allows to replace the first step of the Boneh-Franklin protocol and speed up the running time of this algorithm since this avoids many rewindings in the phase 3 of Boneh-Franklin. Moreover, the sieving protocol can be adapted to take into account the small factors in the factorization of p' and q'.

4.4 The key generation of N such that Q_N is cyclic

Here, we show how to generate N such that the group Q_N is cyclic. To guarantee that Q_N is cyclic, we use the fact that the product of two cyclic groups which orders are coprime is a cyclic group. The following lemma and the GCD protocol enables to check that p' and q' are coprime in a distributed way. First we prove a lemma which has been used in another form in [9].

Lemma 1. *Let $N = pq$ an RSA modulus,* $\gcd(p-1, q-1) | \gcd(N-1, \varphi(N))$ *and the square free part of* $\gcd(N-1, \varphi(N))$ *divides* $\gcd(p-1, q-1)$.

Proof. We can note that $\varphi(N) = N - p - q + 1 = (N-1) - (p-1) - (q-1)$. So,

$$(N-1) - \varphi(N) = (p-1) + (q-1)$$

Consequently, $\gcd(N-1, \varphi(N)) = \gcd((N-1) - \varphi(N), \varphi(N)) = \gcd((p-1)+(q-1), \varphi(N))$. If we note $a = p-1$ and $b = q-1$, we have to compare $\gcd(a+b, ab)$ and $\gcd(a, b)$. It is easy to see that $\gcd(a,b) | \gcd(a+b, ab)$, because if $f | \gcd(a,b)$, $f|a$ and $f|b$, so $f|a+b$ and $f|ab$.
But let $f | \gcd(a+b, ab)$. As,

$$\gcd(a+b, ab) = \gcd(a+b, ab - a(a+b)) = \gcd(a+b, -a^2) = \gcd(a+b, a^2)$$

We can assume that $f|(a+b)$ and $f|a^2$. If f is a prime number, $f|a$ and as $f|(a+b)$, $f | \gcd(a,b)$. If f is not a prime number but a power of some prime number, say $f = f'^{\alpha}$, we have $f'^{\alpha}|a^2$ and $\alpha = 2\beta$. Hence, $f'^{\beta}|a+b$ and $f'^{\beta}|a$, so $f'^{\beta} | \gcd(a,b)$. □

Corollary 1. *If* $\gcd(N-1, \varphi(N)) = 2$, *then* $\gcd(p-1, q-1) = 2$.

Proof. If p' and q' have not 2 as a prime factor, therefore, we have to check whether $\gcd(N-1, \varphi(N)) = 2$ in order to see that $\gcd(p-1, q-1) = 2$. These last verification can be made using the GCD protocol described in section 4.2 □

Theorem 2. *The key generation protocol of Boneh-Franklin and the GCD protocol allow to generate RSA moduli such that the group Q_N is cyclic of order $M = p'q'$, where $N = pq$, $p = 2p' + 1$, $q = 2q' + 1$ and neither p' nor q' have prime factors smaller than B. The iteration number of this protocol with respect to the Boneh-Franklin protocol is on average $4 \times e^{\gamma} \ln(B)$.*

Proof. Following section 4.3, we can assume that we get an RSA modulus such that $p-1$ and $q-1$ have all theirs prime divisors greater than B, they do not have common divisors, *i.e.*, $\gcd(p-1, q-1) = 2$ and $\frac{p-1}{2}$ and $\frac{q-1}{2}$ do not have small prime factors. As the product of cyclic groups whose order are coprime is a cyclic group, the groups of squares in $\mathbb{Z}_p{}^*$ and in $\mathbb{Z}_q{}^*$ are cyclic, and so the group Q_N is also cyclic. This allows to guarantee that there exists a cyclic subgroup in $\mathbb{Z}_N{}^*$ of order $M = p'q'$.

We can estimate the iteration number of this algorithm with respect to the Boneh-Franklin protocol. First, it is a well-known fact that $\Pr_{p',q'}[\gcd(p', q') =$

1] $= \frac{6}{\pi^2} > 1/2$. Moreover, the only slowing factor at the key generation is the check that $\gcd(P', p-1) = 2$, where $P' = 4P$. We can note that $\Pr_{p'}[\gcd(2p', P') = 2] = \Pr_{p'}[2 \nmid p' \wedge 3 \nmid p' \wedge \ldots \wedge B \nmid p'] = (1-\frac{1}{2})(1-\frac{1}{3})\ldots(1-\frac{1}{B}) = \prod_{p_i \leq B}(1-\frac{1}{p_i}) \approx \frac{1}{e^\gamma \ln(B)}$ according to the second theorem of Mertens, where γ is the Euler constant. Therefore, we have to run this algorithm $2 \times (e^\gamma \ln(B) + e^\gamma \ln(B))$ on average in order to get such RSA moduli. □

4.5 Proofs of security and robustness

Theorem 3. *The distributed key generation is secure and robust against static and malicious adversaries controlling up to t servers.*

Proof. We begin to show that the distributive version of the key generation is secure against a t-static adversary and then, we will show that this protocol is robust against a t-malicious adversary.

Indistinguishability of data received by the adversary.
In the proof of secrecy, we have to show that if there exists an adversary $\mathcal{A}$ which corrupts at most t servers at the beginning of the game, then we can use it to construct an attacker against the centralized version of the key generation protocol in order to factor the modulus N. This attacker is a simulator, whose role is to simulate fake information to the adversary in order to provide him the same information as it will receive in its normal game so that this fake information cannot be distinguished from real ones by the adversary. Therefore, the attacker will be sometimes called the simulator $\mathcal{S}im$.

Let us consider the following game A :

A1 The attacker chooses to corrupt t servers. He learns all their secret information and actively controls their behavior.
A2 The key generation algorithm is run; the public and secret keys and the verification keys are computed.
A3 The adversary tries to factor the RSA modulus based on the information he learned at the key generation or to learn information on the shares of the private key hold by the non corrupted servers.

While the Boneh and Franklin scheme requires to simulate only the *public modulus* N at key generation, our scheme additionally needs *verification keys*. We begin to prove the **security of the RSA modulus**.

In lemma 2.1 of [2], Boneh and Franklin reduce the distributed protocol to the centralized protocol and show in particular that if there is an attacker $\mathcal{A}$ that achieves breaking the secrecy of the distributed key generation protocol (*i.e.*, with t parts of the secret key), there is a simulator $\mathcal{S}im$ that factors a non-negligible amount of RSA modulus N of size n. The aim of their reduction is to prove the secrecy of the RSA modulus assuming the hardness of factoring a non-negligible fraction of the RSA moduli in $\mathbb{Z}_n^{(2)}$, where $\mathbb{Z}_n^{(2)}$ is the set of RSA moduli such that $N = pq$ that can be output by their protocol when t parties

are involved and each party picks two shares p_i and q_i of $n/2$ bits. They prove the following theorem.

Theorem 4. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ that given : (1) a random $N \in \mathbb{Z}_n^{(2)}$ chosen from the distribution on $\mathbb{Z}_n^{(2)}$ induced by the protocol, and (2) the shares $\langle p_i, q_i \rangle$ of t parties, factors N with probability at least $1/n^c$. Then there exists an expected polynomial time algorithm $\mathcal{B}$ that factors $1/4(t+1)^3 n^c$ of the integers in $\mathbb{Z}_n^{(2)}$.*

We argue that the modification that we have made on the key generation protocol do not change Boneh and Franklin result on N. Indeed, we slightly restrict the choice of p and q by allowing only values such that $\gcd(p-1, q-1) = 2$ and $\frac{p-1}{2}$ and $\frac{q-1}{2}$ do not have small prime factors. We have seen in theorem 2 that we decrease the probability by a factor $1/4(e^\gamma \log(B))$. Hence, the result is still valid by replacing $1/4(t+1)^3 n^c$ by $1/16(e^\gamma \log(B))(t+1)^3 n^c$.

Now, we have to prove that the information revealed by the **verification keys** does not help the adversary.

From the information known by the adversary, namely the t shares of the private key obtained from the corrupted servers and the verification keys $v_u \in Q_N$ for $u \in [1, k]$, we want to prove that the verification key of each server can be computed by the adversary. Hence, this revealed information cannot give him information on the shares of the private key held by non corrupted servers.

Without loss of generality, we can assume that $\mathcal{A}$ corrupt the t first servers. So, the attacker chooses the secret keys $d_1, \ldots, d_t$ of the corrupted players in the phase A1 of game A; d_i should be in the interval $\{0, \ldots, M\}$, but since M is unknown, $\mathcal{S}im$ picks d_i in $\{0, \ldots, \lfloor N/4 \rfloor\}$. Anyway, the statistical distance between the uniform distribution on $\{0, \ldots \lfloor N/4 \rfloor - 1\}$ and the uniform distribution on $\{0, \ldots M-1\}$ is $O(N^{-1/2})$ so the adversary $\mathcal{A}$ cannot distinguish real and simulated corrupted secret keys.

In the simulation, $\mathcal{S}im$ chooses the v_u's by picking k random elements w_u's in ${\mathbb{Z}_N}^*$ such that $v_u = {w_u}^{2e} \bmod N$. Therefore, we know that ${v_u}^d = {w_u}^2 \bmod N$ and the v_u's are in Q_N. The verification keys of corrupted servers are computed using the known secret keys d_i and the missing $v_{u,t+1}, \ldots, v_{u,\ell}$ are obtained with the Lagrange interpolation formula. Of course, we are not able to find the missing secret keys but in fact we do not need them.

Indeed, we denote by S, 0 and the index of corrupted players. For each verification keys v_u, the shares of the corrupted server i are $v_{u,i} = {v_u}^{\Delta d_i} \bmod N$ where we denote $\ell!$ by Δ. $\mathcal{A}$ can compute these values as he knows d_i for the set of corrupted servers. For the other servers, the simulator uses the polynomial interpolation.

$$v_{u,i} = {w_u}^{2\lambda_{i,0}^S} \times \prod_{j \in S \setminus \{0\}} {v_u}^{d_j \lambda_{i,j}^S} = {v_u}^{d\lambda_{i,0}^S + \sum_{j \in S \setminus \{0\}} d_j \lambda_{i,j}^S} \bmod N$$

where $S = \{0, 1, \dots t\}$,

$$\lambda_{i,j}{}^{S} = \Delta \frac{\prod_{j' \in S \setminus \{j\}} (i - j')}{\prod_{j' \in S \setminus \{j\}} (j - j')} \in \mathbb{Z} \quad \text{and} \quad \Delta f(i) = \sum_{j \in S} \lambda_{i,j}^{S} f(j) \bmod M$$

Robustness of the key generation protocol.
To show the robustness of our scheme we use two robustness results. The first one comes from the Frankel *et al.* proof of [7] where they make the Boneh and Franklin protocol robust against malicious t-adversary. The other result comes from the Catalano *et al.* proof of [4] where they make the inverse computation modulo a shared value robust against t-adversary. Moreover, the sieving protocol, developed by Boneh *et al.* can be made robust as well. So, we have to show that this last protocol can be achieved in a robust way. As it uses the BGW protocol, if we add some commitment stages, it is possible to make this last protocol robust. □

5 Enhancing the Shoup scheme

The aim of this section is to revisit the proof of correctness originally designed by Shoup to cover the case of RSA moduli generated as in the previous section. This uses a method by which we generate the entire group of squares with few random elements with high probability.

5.1 Proof of correctness

Let N be a modulus such that $N = pq$ and $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' have no small prime factors and $\gcd(p - 1, q - 1) = 2$. Accordingly Q_N is cyclic and there exists a generator g in Q_N. Thus, the discrete log of any element ${c_i}^2$ in basis g exists, where $c_i = c^{2\Delta d_i}$ and $\Delta = \ell!$. As we will see in section 5.3, we can denote by $v_1, \dots, v_k$ a k-tuple of random elements in $(Q_N)^k$ such that with high probability, this tuple generates the whole group Q_N of order $M = p'q'$, *i.e.* for each $x \in Q_N$, there exists $(a_1, \dots, a_k) \in [0, M[^k$ such that $x = \prod_{i=1}^{k} {v_i}^{a_i} \bmod N$.

Each server i has a k-tuple of verification keys $v_{1,i} = {v_1}^{d_i} \bmod N, \dots, v_{k,i} = {v_k}^{d_i} \bmod N$. He computes a signature share, $c_i = c^{2d_i\Delta} \bmod N$, where d_i is the ith signature share of d and proves that

$$\log_{v_1}(v_{1,i}) = \dots = \log_{v_k}(v_{k,i}) = \log_{c^4\Delta}({c_i}^2)$$

The value ${c_i}^2$ is a square and is an element of Q_N.

Now, we describe the proof of "correctness" and still let $d_i \in [0, M[$ be the secret share of a server, and A and B' two integers such that $\log(A) \geq \log(B'Mh) + k_2$ where B' and k_2 are security parameters and h is the number of rounds. Finally, k_1 is a parameter such that the cheating probability $1/{B'}^h$ is

$< 1/2^{k_1}$. Whereas security parameter k_1 controls the completeness and statistical zero-knowledge results, security parameter k_2 controls the soundness result. We present the proof for $h = 1$.

The prover chooses a random r in $[0, A[$. Then, he computes $t = (v'_1, \ldots, v'_k, c') = (v_1^r, \ldots, v_k^r, c^{4\Delta r})$. Let e be the first $b' = \log(B') - 1$ bits of the hash value

$$e = [H(v_1, \ldots, v_k, c^{4\Delta}, v_{1,i}, \ldots, v_{k,i}, {c_i}^2, v'_1, \ldots, v'_k, c')]_{b'}$$

if we denote by $[x]_{b'}$ the first b' bits of x. Next, the prover calculates z where $z = r + ed_i$. The proof is the pair $(e, z) \in [0, B'[\times[0, A[$. To check it, the verifier has to compute whether

$$e = [H(v_1, \ldots, v_k, c^{4\Delta}, v_{1,i}, \ldots, v_{k,i}, {c_i}^2, {v_1}^z {v_{1,i}}^{-e}, \ldots, {v_k}^z {v_{k,i}}^{-e}, c^{4\Delta z} {c_i}^{-2e})]_{b'}$$

and verify whether $0 \leq z < A$.

5.2 Security analysis of the proof of correctness

Proof of Completeness.

Theorem 5. *The execution of the protocol between a prover who knows the secret d_i and a verifier is successful with overwhelming probability if $B'Mh/A$ is negligible where h is the number of rounds.*

Proof. If the prover knows a secret $d_i \in [0, M[$ and follows the protocol, he fails only if some $z \geq A$. For any value $x \in [0, M[$ the probability of failure of such event taken over all possible choices of r is smaller than $B'M/A$. Consequently the execution of the proof is successful with probability $\geq (1 - \frac{B'M}{A})^h \geq 1 - \frac{B'Mh}{A}$. □

Proof of Soundness. Let us focus on soundness.

Lemma 2. *If the verifier accepts the proof, with probability $\geq 1/B' + \epsilon$ where ϵ is a non-negligible quantity, then using the prover as a "black-box" it is possible to compute σ and τ such that $|\sigma| < A$ and $|\tau| < B'$ such that ${v_1}^\sigma = {v_{1,i}}^\tau, \ldots, {v_k}^\sigma = {v_{k,i}}^\tau, c^{4\Delta\sigma} = {c_i}^{2\tau}$.*

Proof. If we rewind the adversary and get two valid proofs for the same commitment t, (e, z) and (e', z'), we have for $u = 1, \ldots, k$ *i.e.* for all verification keys, ${v_u}^r = {v_u}^z v_{u,i}^{-e} = {v_u}^{z'} v_{u,i}^{-e'}$. So, we obtain ${v_u}^\sigma = {v_{u,i}}^\tau \bmod N$ if we set $\sigma = z' - z$ and $\tau = e - e'$. Therefore we can write ${v_1}^\sigma = {v_{1,i}}^\tau, \ldots, {v_k}^\sigma = {v_{k,i}}^\tau, c^{4\Delta\sigma} = {c_i}^{2\tau}$. □

Theorem 6. ***(Soundness)*** *Assume that some probabilistic polynomial Turing machine $\tilde{P}$ is accepted with non-negligible probability. If $B' < B$, $h \times \log(B') = \theta(k_1)$, $k = \theta(k_1 / \log(B))$ and $\log(A)$ is a polynomial in k_1 and $\log(N)$, we can prove that $c^{4\Delta d_i} = {c_i}^2$ and so c_i is a correct signature share.*

Proof. By the previous lemma we can assume that we have τ and σ such that ${v_u}^{\sigma} = {v_u}^{d_i\tau}$ for $u = 1, \ldots, k$ and $c^{4\Delta\sigma} = {c_i}^{2\tau}$.

Then, we can write $c^{4\Delta}$ with the set of generators of Q_N since it is a square : $c^{4\Delta} = {v_1}^{\beta_1} \times \ldots \times {v_k}^{\beta_k}$.

Consequently if we raise this equation to the power σ, we obtain $c^{4\Delta\sigma} = {v_1}^{\sigma\beta_1} \times \ldots \times {v_k}^{\sigma\beta_k}$. But, $c^{4\Delta\sigma}$ is equal to ${c_i}^{2\tau}$ and ${v_1}^{\sigma\beta_1} \times \ldots \times {v_k}^{\sigma\beta_k}$ is equal to $({v_1}^{\beta_1} \times \ldots \times {v_k}^{\beta_k})^{\tau d_i}$ as ${v_u}^{\sigma} = {v_u}^{d_i\tau}$ for $u = 1, \ldots, k$.

Therefore, ${c_i}^{2\tau} = (c^{4\Delta})^{\tau d_i}$ with $|\tau| < B'$. We can simplify this equation by τ if τ is coprime with $p'q'$. So we obtain $c^{4\Delta d_i} = {c_i}^2$ if $B' < B$.

Let $\tilde{\pi}(k_1)$ the probability of success of $\tilde{P}$. If $\tilde{\pi}(k_1)$ is non-negligible, there exists an integer c such that $\tilde{\pi}(k_1) \geq 1/{k_1}^c$ for infinitely many values k_1. The probability for $\tilde{P}$ to generate a correct signature share while the v_is generate the group Q_N is larger than $\tilde{\pi}(k_1) - 2 \times \frac{2}{k-1} \times \frac{1}{B^{k-1}}$ according to the result of the section 5.3. So, if $k = \theta(k_1/\log(B))$, for infinitely many values k_1, $2 \times \frac{2}{k-1} \times \frac{1}{B^{k-1}} \leq 1/3{k_1}^c$.

Furthermore, for k_1 large enough, $1/{B'}^h < 1/3{k_1}^c$ if $h \times \log(B') = \theta(k_1)$. So by taking $\epsilon = \tilde{\pi}(k_1)/3$ in lemma 2 we conclude that it is possible to obtain (σ, τ) in polynomial time $O(1/\epsilon) = O({k_1}^c)$. □

Proof of Statistical Zero-Knowledge.

Proof. Furthermore, we can prove that if A is much larger than $B' \times N$, the protocol statistically gives no information about the secret. In the random oracle model where the attacker has a full control of the values returned by the hash function H, we define the first b' bits of the value of H at

$$(v_1, \ldots, v_k, c^{4\Delta}, v_{1,i}, \ldots, v_{k,i}, {c_i}^2, {v_1}^z {v_{1,i}}^{-e}, \ldots, {v_k}^z {v_{k,i}}^{-e}, c^{4\Delta z} {c_i}^{-2e})$$

to be e. With overwhelming probability, the attacker has not yet defined the random oracle at this point so the adversary $\mathcal{A}$ cannot detect the fraud. □

5.3 Choice of parameters

In this section we prove that with high probability we generate the entire square group Q_N with only few random elements.

Theorem 7. *With probability greater than* $1 - 2 \times \frac{2}{k-1} \times \frac{1}{B^{k-1}}$*, a random* k*-tuple* $(v_1, \ldots, v_k)$ *generates* Q_N*.*

Let us first define additional notations. If $(v_1, \ldots, v_k)$ is a k-tuple of $(Q_N)^k$, we use $\langle v_1, \ldots, v_k \rangle$ to denote the subgroup of Q_N that is generated by the v_i's, *i.e.*,

$$\langle v_1, \ldots, v_k \rangle = \{x \in Q_N | \exists (\lambda_1, \ldots, \lambda_k)\ x = \prod_{i=1}^{k} v_i^{\lambda_i} \bmod N\}$$

We also denote by $\zeta(k)$ the Riemann Zeta function defined by $\zeta(k) = \sum_{d=1}^{+\infty} \frac{1}{d^k}$ for any integer $k \geq 2$. If $n = {q_1}^{e_1} \times {q_2}^{e_2} \times \ldots \times {q_j}^{e_j}$, we denote by $\varphi_k(n)$

$$n^k \times (1 - \frac{1}{{q_1}^k})(1 - \frac{1}{{q_2}^k}) \ldots (1 - \frac{1}{{q_j}^k})$$

the generalization of the Euler function in the case of k generators. Finally, if n has no prime factors less than B, we define $\zeta_B(k)$ has $\sum_{d=B}^{+\infty} \frac{1}{d^k}$.

We can note that $\frac{p-1}{2}$ and $\frac{q-1}{2}$ are mutually prime. To find a generator v of Q_N, we have to find a v such that $v \bmod p$ generates Q_p and $v \bmod q$ generates Q_q.

We estimate the probability that $x \in Q_N$ is a generator of Q_N. The probability to catch such number depends on the factorization of the order p' of Q_p and q'. Yet, even if $M = p'q'$ has no small factors, the probability is to obtain such generator is not overwhelming. Indeed, if we pick a random element v in Q_p, the probability that v is a generator of Q_p is

$$Pr = \Pr_{v \in Q_p}(\langle v \rangle = Q_p) = \frac{\varphi(p')}{p'} = \prod_{p_i \geq B, p_i | p-1} (1 - \frac{1}{p_i}) \leq 1 - \frac{1}{p_1}$$

and if $p_1 \leq 2B$, we can bound the probability by $\leq 1 - \frac{1}{2B}$. The probability that $B \leq p_1 \leq 2B$ is equal to the probability that p' is divisible by at least one prime that belongs in $[B, 2B]$. So, $\Pr_{p_1}[B \leq p_1 \leq 2B] = \sum_{B \leq q_i \leq 2B, q_i prime} \frac{1}{q_i} \geq \frac{1}{2B} \times (\pi(2B) - \pi(B))$ if we denote by $\pi(x)$ the number of primes between 2 and x. If $B = 2^{16}$, with probability $\geq 1/26$, $Pr \leq \frac{1}{2^{17}}$. Consequently, we cannot say that this probability is overwhelming.

However, if we allow to choose several random elements in Q_N, then the subgroup $\langle v_1, \ldots, v_k \rangle$ is a equal to Q_N with high probability. A k-tuple $(v_1, \ldots, v_k)$ is a set of generators of Q_N if $(v_1 \bmod p, \ldots, v_k \bmod p)$ is a set of generators of Q_p and if $(v_1 \bmod q, \ldots, v_k \bmod q)$ is a set of generators of Q_q. Hence, the number of k-tuples of $(Q_N)^k$ that generate Q_N is the number of these k-tuples viewed as elements of $(Q_p)^k$ that generate Q_p and viewed as elements of $(Q_q)^k$ that generate Q_q.

There are $p' = \frac{p-1}{2}$ elements in Q_p. To generate this cyclic subgroup of ${\mathbb{Z}_p}^*$ (since it is a subgroup of a cyclic group), there are $\varphi(p')$ such generators.

The analysis made by Poupard and Stern in [14] can be extended in our context as it is true in general cyclic groups and not only in ${\mathbb{Z}_{p^e}}^*$. Let us now present a preliminary lemma.

Lemma 3. *The number of k-tuples of $(Q_p)^k$ that generate Q_p is $\varphi_k(p')$.*

The proof of this lemma is given in appendix.

Now, we return back to the proof of the theorem 7. Let us first introduce a notation : for any integer x, let S_x be the set of the indices i such that p_i is a factor of x. From the previous lemma, we know that the probability for a

k-tuple of $(Q_p)^k$ to generate Q_p is $\frac{\varphi_k(p')}{{p'}^k}$. Lemma 3 shows that Pr is equal to the product $\prod_{i\in S_{p'}} 1-\frac{1}{{p_i}^k}$. The inverse of each term $1-\frac{1}{{p_i}^k}$ can be expanded in power series : $(1-\frac{1}{{p_i}^k})^{-1} = \sum_{j=0}^{\infty}(1/{p_i}^k)^j$. The probability Pr is a product of series with positive terms, Pr= $(\prod_{i\in S_{p'}} \sum_{\alpha_i=0}^{\infty} \frac{1}{{p_i}^{\alpha_i k}})^{-1}$ so we can distribute terms and obtain that $\Pr^{-1}$ is the sum of $1/d^k$ where d ranges over integers whose prime factors are among p_is, $i \in S_{p'}$. This sum is smaller than the unrestricted sum $\sum_{d=1}^{\infty} 1/d^k = \zeta(k)$. Finally, we obtain $\Pr > 1/\zeta(k)$.

In our case, neither p' nor q' have prime factors less than B, therefore, the Riemann Zeta function is bounded by the following integral : $\zeta_B(k) = \sum_{d=B}^{\infty} 1/d^k < 1 + 1/B^k + \int_B^{\infty} dx/x^k = 1 + \frac{k-1+B}{k-1} \times \frac{1}{B^k}$. Since for all $x > -1$, $1/(1+x) \geq 1-x$, $1/\zeta(k) > 1 - \frac{k-1+B}{k-1} \times \frac{1}{B^k}$.

Therefore, the number of k-tuples of $(Q_p)^k$ that generate Q_p is $\varphi_k(p')$ and

$$\Pr_{(v_1,\ldots,v_k)\in(Q_p)^k}\{\langle v_1,\ldots,v_k\rangle = Q_p\} = \frac{\varphi_k(p')}{{p'}^k} > \frac{1}{\zeta(k)} > 1 - \frac{k+B-1}{k-1} \times \frac{1}{B^k}$$

Consequently, with probability greater than $1 - 2 \times \frac{2}{k-1} \times \frac{1}{B^{k-1}}$, the k-tuple $(v_1,\ldots,v_k)$ generates Q_p and Q_q and therefore Q_N. For example, with $k = 6$ and $B = 2^{16}$, this probability is larger than $1 - 1/2^{80}$.

6 Practical parameters for the scheme

In the key generation we can test whether p, q, p' and q' are divisible by small primes $\leq B$ and $\gcd(p', q') = 1$. We can assume that B is the first prime greater than 2^{16}. The loss in the key generation phase is a factor 80 on average. Indeed, we have seen in the proof of theorem 2 section 4.4 that $\Pr_{p'}[p' \text{ has no small prime factors} \leq B] \approx \frac{1}{e^{\gamma}\ln(B)}$. If we fix B to 2^{16}, we can assume that $\Pr_{p'}[p' \text{ has no small prime factors} \leq B] > \frac{1}{20}$. Therefore, to generate p and q such that neither p' nor q' have small prime factors and such that $\gcd(p-1, q-1) = 2$, we have to run on average $2 \times (20+20) = 80$ times this protocol. This factor is not critical as this algorithm is run only once.

In the proof of correctness, if we want to have a security parameter of 2^{80}, we choose $B' = 2^{16} < B$. Hence, we have to choose $h = 5$ rounds. To generate the group of squares with probability greater than $1 - 2^{80}$, we need $u = 6$ verification keys. Therefore, we need 30 proofs of correctness but is is acceptable in the applications that we have in mind.

7 Conclusion

In this paper, we have showed how to avoid safe prime RSA modulus in Shoup's proof of robustness such that the proof remains correct. We consider environments where high security is required such as electronic voting schemes, and therefore, we need protocols using standard assumptions and we are ready to pay the price for it.

Basically, we use three different techniques allowing to prove that :

- the group of square is cyclic,
- we generate p and q such that p' and q' do not contain small prime factors, which allows us to generate the group Q_N
- we generate a set of generators of Q_N by picking at random different generators in Q_N.

Finally, we show how to adapt Shoup proof in order to work with different elements that generate Q_N instead of a single one.

References

1. M. Ben-Or, S. Goldwasser, and A. Widgerson. Completeness theorems for non-cryptographic fault-tolerant distributed computing. In *Proceedings of the 20th STOC*, ACM, pages 1–10, 1988.
2. D. Boneh and M. Franklin. Efficient Generation of Shared RSA keys. In *Crypto '97*, LNCS 1233, pages 425–439. Springer-Verlag, 1997.
3. D. Boneh, M. Malkin, and T. Wu. Experimenting with Shared Generation of RSA keys. In *Internet Society's 1999 Symposium on Network and Distributed System Security (SNDSS)*, pages 43–56, 1999.
4. D. Catalano, R. Gennaro, and S. Halevi. Computing Inverses over a Shared Secret Modulus. In *Eurocrypt '00*, LNCS 1807, pages 190–207. Springer-Verlag, 2000.
5. I. Damgård and M. Koprowski. Practical Threshold RSA Signatures Without a Trusted Dealer. Technical report, Aarhus University, BRICS, November 2000.
6. Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Optimal Resilience Proactive Public-Key Cryptosystems. In *FOCS '97*, pages 384–393, 1997.
7. Y. Frankel, P. MacKenzie, and M. Yung. Robust Efficient Distributed RSA Key Generation. In *STOC '98*, pages 663–672, 1995.
8. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust Threshold DSS Signatures. In *Eurocrypt '96*, LNCS 1070, pages 425–438. Springer-Verlag, 1996.
9. R. Gennaro, D. Micciancio, and T. Rabin. An Efficient Non-Interactive Statistical Zero-Knowledge Proof System for Quasi-Safe Prime Products. In *Proc. of the Fifth ACM Conference on Computer and Communications Security '98*. ACM, 1998.
10. N. Gilboa. Two Party RSA Key Generation. In *Crypto '99*, LNCS 1666. Springer-Verlag, 1999.
11. C. Pomerance. On the distribution of pseudoprimes. In *Mathematics of Computation*, 37(156), pages 587–593, 1981.
12. C. Pomerance. Two methods in elementary analytic number theory. pages 135–161. Kluwer Academic Publishers, 1989.
13. G. Poupard and J. Stern. Generation of Shared RSA Keys by Two Parties. In *Asiacrypt '98*, LNCS 1514, pages 11–24. Springer-Verlag, 1998.
14. G. Poupard and J. Stern. Short Proofs of Knowledge for Factoring. In *PKC '00*, LNCS 1751, pages 147–166. Springer-Verlag, 2000.
15. T. Rabin. A Simplified Approach to Threshold and Proactive RSA. In *Crypto '98*, LNCS 1462, pages 89–104. Springer-Verlag, 1998.
16. R. Rivest. Finding Four Million Large Random Primes. In *Crypto '90*, LNCS 537, pages 625–626. Springer-Verlag, 1991.
17. V. Shoup. Practical Threshold Signatures. In *Eurocrypt '00*, LNCS 1807, pages 207–220. Springer-Verlag, 2000.
18. R.D. Silverman. Fast Generation of Random, Strong RSA Primes. RSA Laboratories, May 1997.

8 Appendix

In this appendix, we prove the lemma 3.

Proof. Let $(v_1, \ldots, v_k)$ be k-tuple of $(Q_p)^k$ and v be a generator of Q_p; for $i = 1, \ldots, k$, we define $\alpha_i \in \mathbb{Z}_{p'}$ by the relation $v^{\alpha_i} = v_i \bmod p$.

We first notice that $(v_1, \ldots, v_k)$ generates Q_p if and only if the ideal generated by $\alpha_1, \ldots, \alpha_k$ in the ring $\mathbb{Z}_{p'}$ is the entire ring. Bezout equality shows that this event occurs iff $\gcd(\alpha_1, \ldots, \alpha_k, p') = 1$.

Now, we count the number of k-tuples $(\alpha_1, \ldots, \alpha_k) \in (Q_p)^k$ such that $\gcd(\alpha_1, \ldots, \alpha_k, p') = 1$.

Let $\prod_{i=1}^{t'} {q_i}^{f_i}$ the prime factorization of p'. We know that

$$\gcd(x, \prod_{i=1}^{t'} {q_i}^{f_i}) = 1 \iff \forall i \leq t',\ \gcd(x, {q_i}^{f_i}) = 1 \iff \forall i \leq t',\ \gcd(x \bmod {q_i}^{f_i}, {q_i}^{f_i}) = 1$$

Using the Chinese remainder theorem, the problem reduces to counting the number of k-tuples $(\beta_1, \ldots, \beta_k)$ of $(\mathbb{Z}_{{q_i}^{f_i}})^k$ such that $\gcd(\beta_1 \bmod {q_i}^{f_i}, \ldots, \beta_k \bmod {q_i}^{f_i}, {q_i}^{f_i}) = 1$ for $i = 1, \ldots, t'$. The k-tuples that do not verify this relation for a fixed index i are of the form $(q_i\gamma_1, \ldots, q_i\gamma_k)$ where $(\gamma_1, \ldots, \gamma_k) \in {\mathbb{Z}_{{q_i}^{f_i-1}}}^k$ and there are exactly ${q_i}^{k(f_i-1)}$ such k-tuples.
Finally, there are $\prod_{i=1}^{t'}({q_i}^{kf_i} - {q_i}^{k(f_i-1)})$ k-tuples of $(\mathbb{Z}_{p'})^k$ such that $\gcd(\alpha_1, \ldots, \alpha_k, p') = 1$ and this is equal to $\prod_{i=1}^{t'} \varphi_k({q_i}^{f_i}) = \varphi_k(p')$ since φ_k is multiplicative. □