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 this 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 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's protocol but it seems difficult to modify it in order 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 Shoup's protocol 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 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.
Key words: Threshold RSA key generation and signature

1 Introduction

The cryptosystem RSA [34] is widely used in today practical systems. For instance, a lot of PKI products are based on it. In such systems, the protection of the root key needs strong security requirements. Therefore, threshold protocols can be used to share the signature capabilities among a subset of people rather than to give the power of signing to only one person. Moreover, this kind of protocols can withstand stronger adversaries than "centralized" cryptosystems. Indeed, threshold cryptography can cope with *break-ins* adversaries that have the ability to corrupt people and read the memory of servers [6]. These adversaries are stronger than "normal" adversaries that can only read exchanged messages. In a "centralized" cryptosystem, if one *break-ins* adversary attacks the memory, he then knows all the key and the system is down. As this kind of attacks done by intruders (hackers, Trojan horses) or by corrupted insiders are very common and frequently easy to perform, systems must be protected against them. In threshold cryptography, the secret key is split into shares and each share is given to one of a group of servers. However, in order to be sure that at no moment the key is entirely in one machine, one can also distribute the key generation phase. Consequently, we say that a cryptosystem is *fully distributed* if it is distributed from the key generation to the signature or decryption phase.

In the case of discrete-log based cryptosystems, known solutions exist to distribute DSS, El Gamal, Cramer-Shoup [20, 38, 7]. Moreover, a protocol to distribute a discrete-log key has been first proposed by Pedersen in [27]. This protocol has been further revisited to solve a security flaw [21, 15]. Therefore, discrete-log cryptosystems are fully distributed. However, a *fully distributed* version of RSA is a more challenging and important task.

In this paper we propose new techniques to fully distribute RSA. 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 in [37] where the primes of the RSA modulus should be safe. On the other hand, Boneh and Franklin at Crypto '97 [4] describe a protocol to share the key generation of an RSA modulus. However, the generation of *safe* modulus seems to be hard with this protocol. The present work takes a different path by proposing a method to enhance the key generation with some additional properties and revisits Shoup's protocol to work with the resulting RSA moduli.

1.1 Why it is important to share Shoup's threshold RSA ?

Shoup threshold RSA signature scheme [37] 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 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 or $x = H(m)$ 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.

Efficient communication model against active adversary. The main characteristic of Shoup's protocol in relation to previous proposals [17, 16, 32] is the following. In the discrete-log case, it is easy to compute inverses mod q, if we note q the order of the group G generated by g, because q is public. With RSA, we cannot disclose inverses of a known value mod $\varphi(N)$ without revealing the factorization of N unless we use a special algebraic structure, called a module, as in [35, 19]. We can note that computations in such structure can be done efficiently if we consider [25]. If we do not want to use a module, we face the problem of computing inverses when we use polynomial sharing in order to compute the Lagrange coefficients. Consequently some authors in [17, 32] have proposed additive sharings to avoid this calculation. Therefore, as they need all shares to generate the signature, they devise strategies to cope with corrupted or crashed servers. Different strategies can be used to reconstruct the lost shares by non-corrupted servers : either using two different sharings (additive and polynomial) as in [32] or using probabilistic assignments as in [17]. With additive sharing, $d = \sum_{i=1}^{\ell} d_i \bmod \varphi(N)$, the combiner easily computes the signature s from the ℓ correct signature shares $s_i = x^{d_i} \bmod N$, where $x = H(m)$ is the message to be signed, by using the formula :

$$s = \prod_{i=1}^{\ell} s_i \left(= \prod_{i=1}^{\ell} x^{d_i} = x^{\sum_{i=1}^{\ell} d_i} = x^d \bmod N \right) \tag{1}$$

The main drawbacks of these techniques are the size of the key shares because of the need of different sharings and the use of protocols to reconstruct the bad signature shares in the presence of active (malicious) players.

In [16], the authors proposed the first proven scheme based on polynomial sharing, which is based on Desmedt and Frankel's scheme [13]. However in the case of active adversaries, which are allowed to send bad shares, the protocol has to be rewind at most t times, to remove the bad servers as the signature shares depend on the subgroup of $t+1$ servers enabling the reconstruction of the signature. Let $\Delta = \ell!$. The shares of d are such that $\Delta | d_i$ and $d_i = f(i)$ where f is a polynomial of degree t and of constant term equals to d. If we denote by S the subgroup of $t+1$ servers, let Lagrange coefficients to be ${\lambda'}_{i,j}^{S} = \prod_{j' \in S \setminus j} \frac{i-j'}{j-j'}$. Therefore, $d = \sum_{i \in S} {\lambda'}_{0,i}^{S} d_i \bmod \varphi(N)$ from the Lagrange formula. There are two problems. First of all, ${\lambda'}_{i,j}$ cannot be computed in $\mathbb{Z}_{\varphi(N)}$ since $(j - j')$ could be even and not invertible $\bmod \varphi(N)$. Next, the combiner has to compute

$$s = \prod_{i \in S} s_i^{{\lambda'}_{0,i}^{S}} \left(= \prod_{i \in S} s^{{\lambda'}_{0,i}^{S} d_i} = s^{\sum_{i \in S} {\lambda'}_{i,0}^{S} d_i} = s^d \bmod N \right) \tag{2}$$

But, as ${\lambda'}_{i,j}^{S}$'s are not integers and the combiner cannot compute roots modulo a composite number, otherwise it can solve the RSA Problem, he cannot compute equation (2). The key idea is to note that $\Delta \times {\lambda'_{i,j}}^{S}$ are integers. Therefore, if we write $d_i = \Delta \times d'_i$, then, for a group S of $t+1$ servers, each one can compute $l_{0,i}^{S} = \Delta \times {\lambda'}_{0,i}^{S} \in \mathbb{Z}$ and ${s'}_i = x^{{\lambda'}_{i,0}^{S} d_i} = x^{l_{0,i}^{S} \times {d'}_i} \bmod N$. Finally, the combiner computes

$$s = \prod_{i \in S} s'_i \left(= \prod_{i \in S} x^{l^S_{0,j} \times d'_i} = \prod_{i \in S} x^{\Delta \times {\lambda'}^S_{0,i} \times d'_i} = \prod_{i \in S} x^{{\lambda'}^S_{0,i} d_i} = x^d \bmod N \right) \quad (3)$$

If the signature is not valid, the combiner removes the bad servers thanks to a proof of robustness, defines another group S and rewind the protocol. It is then obvious that after t trials, all bad servers are removed from the set S and the signature will be correct. However, this redefinition of the subgroup does not seem very nice and Shoup and others have proposed a new trick to avoid this problem.

Shoup in [37] and Miyazaki, Sakurai and Yung in [26] solve this problem by using a well-known lemma to extract an e-root of w modulo a composite number from a e-root of a known power of w [24] without any secret. The solution is to multiply Lagrange coefficients by Δ such that they are integers : $\lambda^S_{i,j} = \Delta \times {\lambda'}^S_{i,j} \in \mathbb{Z}$ and $\Delta d = \sum_{i \in S} \lambda^S_{0,i} d_i$, if we denote by S a subset of $t+1$ elements. Therefore, if we let $s_i = x^{d_i}$, the combiner can compute signatures and change of group (compute new Lagrange coefficients according to the group of $t+1$ servers) without asking new signature shares to the servers. He computes equation (2) using $\lambda^S_{i,j}$ and gets $s^\Delta = \prod_{i \in S} s_i^{\lambda^S_{0,i}} (= x^{\Delta d}) \bmod N$. Finally, the combiner can compute a e-root of x^Δ with the previous formula and can recover a e-root of x using the well-known lemma. Consequently, if we use Shoup's scheme, there is no need to generate d_i such that $\Delta | d_i$ as it is done in [18, 12].

Even if the protocol of Frankel *et al.* in [16] proposed a fully distributed version of RSA, it is less elegant than Shoup's one and it will be nice to share this protocol. Moreover, this scheme proposes others improvements that are valuable such as the proof of robustness. This proof uses safe primes, like Gennaro *et al.*'s one [20] and avoids the drawbacks of a special relation between prover and verifier. Furthermore, in [16], the authors describe an interactive protocol which leads to a less efficient protocol than Shoup's one which uses non-interactive zero-knowledge proofs. Therefore, we face the problem of distributing this non-interactive proof.

Proof of robustness and use of safe primes. As we said before, a second key point in Shoup's signature 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 in presence of active adversaries that can modify the behavior of servers. In Shoup's scheme, the proof of correctness requires an RSA modulus built with safe primes. In the proof of correctness, servers must prove that they raise x to the correct power, namely d_i, their share of the secret. To this end, each server i has a verification key $v_i = v^{d_i} \bmod N$ and makes a proof that $\log_v v_i = \log_x s_i (= d_i \bmod \varphi(N))$. The problems are : $\mathbb{Z}_N{}^*$ is not a cyclic group, its order is unknown, such generator v do not exist and elements of maximal order cannot be easily found. However, Shoup noted that if we use

RSA moduli with safe primes, then the group of squares in $\mathbb{Z}_N{}^*$ is cyclic and it is easy to find generators. Consequently, the proof of correctness can be made non-interactively and correctly proved without further assumptions. Finally, safe prime moduli are also used in the key generation protocol in order to guarantee the secrecy of Shamir's secret sharing.

Shared Generation of RSA Keys. This raises the question of generating RSA moduli for Shoup's threshold scheme without a trusted dealer. There exist protocols that generate RSA keys in a distributive manner [4, 18, 9, 10, 3, 30, 23]. Boneh and Franklin in [4] designed such protocol for the generation of an RSA modulus in the honest-but-curious model. Later, Frankel, MacKenzie and Yung in [18] made this algorithm robust against malicious servers. In [30], Poupard and Stern also provided a protocol to compute a shared modulus for two players only. Finally, Gilboa in [23] has extended Poupard and Stern method. As we can note, the Boneh and Franklin protocol is most efficient but no protocol is known to efficiently create shared safe RSA moduli.

1.2 Outline of the paper

We begin by presenting the problem in section 2, *i.e.* where the properties of safe primes are used in Shoup's protocol, and in section 3 the security model. Next, in section 4, we describe how to enhance the Boneh-Franklin scheme to generate RSA moduli having special requirements. In section 5 we show that Shoup's protocol is still secure against passive adversary and in section 6, we show a new proof of correctness making Shoup scheme robust against active adversary. Finally, in section 7 we present practical parameters for our scheme.

1.3 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, ℓ, t, k', k_1 and k_2,

- we use Q_N to denote the group of squares 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. A RSA modulus $N = pq$ is called a *safe prime modulus* if p and q are both safe primes.

2 The problem

As we will see in the following, safe primes are used in the key generation in order to prove that Shamir secret sharing scheme [36] is secure in the ring $\mathbb{Z}_M$, and not in a finite field, and in the proof of correctness. Let us explain the second problem as it is less obvious.

2.1 What is the problem ?

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

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, generate the possible signatures s' and test whether s' is a valid signature of m. If s' passes the verification protocol, 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. 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 non-interactively and the same kind of proof appears in [32, 19] but still requires safe prime modulus.

In order 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 have recently considered the same problem in [12]. They revisited Shoup's paper and used non-standard assumptions to show that the proof of correctness works without other requirements on the RSA modulus. They use [18] to generate standard RSA moduli.

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. We believe that standard assumptions and security proofs are needed to build secure protocols. Several electronic schemes [14, 11, 1] have been based on Paillier cryptosystem which is related to RSA. The techniques developed in the paper can be used to *fully share* this cryptosystem.

2.2 Our results

We prove that Shoup's protocol can be modified to work with RSA moduli having special properties under standard assumptions and that these moduli can be jointly generated.

Safe prime moduli are needed in the proof of robustness and in the key generation. Moreover, different characteristics of these numbers are used in the proof of robustness. Indeed, 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. If we choose enough random elements $(g_1, \ldots, g_k)$, we can 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 [18] and a precise treatment has been given by Poupard and Stern in [31]. Moreover, using a nice trick of Gennaro *et al.* which first appeared in [22] and the protocol recently proposed by Catalano *et al.* in [8], the calculation of $\gcd(p-1, q-1)$ can be performed in a distributed way. These methods allow to keep key generation and signature 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.

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 The Adversary

The adversary is computationally bounded and it can corrupt servers at any moment by viewing the memories of corrupted servers (passive adversary), and/or modifying their behavior (active adversary). The adversary decides on whom to corrupt at the start of the protocol (static adversary). We also assume that the adversary corrupts no more than t out of ℓ servers throughout the protocol, where $\ell \geq 2t + 1$.

3.3 Formal definition

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

- A *key generation algorithm* takes as input security parameters n,the number k of elements to generate Q_N, 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 index $1 \leq i \leq \ell$, the private key d_i and a message m; it outputs a signature share $s_i = x^{d_i} \bmod N$, where $x = H(m)$ and $H(.)$ is a hash-and-pad function, and a proof of its validity $proof_i$ (for all $u \in [1, k]$, $\log_{v_u} v_{u,i} = \log_x s_i$).

- 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.

3.4 The players and the scenario

Our game includes the following players : a combiner, a set of ℓ servers P_i, an adversary and users asking signature. 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_u's and $v_{u,i}$'s 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 $v_u, v_{u,i}$ 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.

3.5 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. This property is useful only in the presence of active adversaries.

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. This property expresses the security of the signature scheme and is useful in the presence of passive or active adversary.

3.6 The Games

In this section, we describe the security notions for threshold key generation and threshold signing protocols. We have to show that the information revealed during the key generation and the signing protocols does not release secret information to the adversary.

Game for the threshold 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 corrupts at most t servers at the beginning of the game, then he cannot obtain more information on the secret key held by uncorrupted players.

Game for the threshold signing protocol. The secrecy of the signing protocol means that if there exists an adversary $\mathcal{A}$ which corrupt at most t servers at the beginning of the game and even if he can obtain signatures on messages adaptively chosen, then he cannot forge a signature on a new message.

4 Enhancing the Boneh-Franklin scheme to generate RSA moduli with special requirements

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, namely $N = pq$, $p = 2p' + 1$ and $q = 2q' + 1$, $\gcd(p', q') = 1$ and no primes $p < B$ divises neither p' nor q'. In section 6, we will prove that this group can be generated with few random elements. Moreover, we use here a sieving method to simultaneously improve the key generation protocol, the probability of finding a set of generators of Q_N, and to make secure Shamir Secret Sharing Scheme. We also present a protocol to compute the GCD of a known value and a shared value and prove the robustness and the secrecy of this new distributed key generation protocol.

4.1 A New Distributed RSA modulus Generation

In [4] Boneh and Franklin present a protocol for generating a shared RSA modulus. We describe this protocol here with our adaptation.

1. In the first step, each server picks at random two values p_i and q_i in the interval $[\lfloor 2^{(n-1)/2} \rfloor, \lfloor \frac{2^{n/2}-1}{\ell} \rfloor[$ according to [39], 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 if $p_1 + \ldots + p_\ell - 1$ or $q_1 + \ldots + q_\ell - 1$ have small prime factors. The servers check whether $\gcd(p-1, 4P) \stackrel{?}{=} 2$ and whether $\gcd(4P, q-1) \stackrel{?}{=} 2$, where $P = \prod_{2 < p_i < B} p_i$ and B is the sieving bound using the GCD algorithm that we describe below.
2. Then the BGW protocol [2, 4] is run to compute the product N of $p_1 + \ldots + p_\ell$ and $q_1 + \ldots + q_\ell$. They also compute the product $\varphi(N) = (p-1)(q-1)$ and check whether $\gcd(\varphi(N), N-1) \stackrel{?}{=} 1$ using the GCD algorithm.
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 [33] where Rivest showed that if a sieving algorithm is first performed, the Miller-Rabin primality test is not needed as pseudoprimes are rare according to

Pomerance's conjectures [28, 29]. Moreover, carmichael numbers are avoided due to a trick similar to Soloway-Strassen primality test. We set $p = p_1 + \ldots + p_\ell$ and $q = q_1 + \ldots + q_\ell$.

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

We briefly recall the protocol presented by Catalano *et al.* [8] for inverting a public value e modulo a shared value φ. The basic trick stems from the observation that $\gcd(e, \varphi) = \gcd(e, \varphi + Re)$ 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 $e \bmod \varphi$, 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 [8]. We can also note that this algorithm allows to compute the gcd of a known value and a shared one. We call this protocol the GCD algorithm.

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

In this subsection, we present the sieving algorithm used in phase 1 of protocol 4.1 and we show how to generate N such that neither $p' = \frac{p-1}{2}$ nor $q' = \frac{q-1}{2}$ have no small prime factors. Our method uses a new distributed sieving protocol designed by Boneh, Malkin and Wu in [5] 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 $[1, P]$ 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 $r_i \in [\lfloor \frac{\sqrt{2}.2^{n/2-1}}{P} \rfloor, \lfloor \frac{2^{n/2}-1}{P\ell} \rfloor[$ 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 $R = \sum_i r_i$. This sieve works only for B s.t. $P < p$. We can increase the bound B to B_1 by checking whether $\gcd(P', p) = 1$ where $P' = \prod_{B \leq p \leq B_1} p$ thanks to the GCD algorithm.

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 as it is explained in appendix 9.1.

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 improve the first step of Boneh-Franklin's protocol and speeds up the running time of this algorithm since this avoids many rewindings in phase 3 of Boneh-Franklin. Moreover, this sieving protocol is 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 this property, 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 [22].

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* $\gcd(p-1, q-1)$.

See appendix section 9.2 for a proof of this lemma.

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

Proof. If $\gcd(N-1, \varphi(N)) = 2$, then as $\gcd(p-1, q-1) | \gcd(N-1, \varphi(N))$, $\gcd(p-1, q-1) = 2$ since $\gcd(p-1, q-1)$ cannot be equal to 1. These last verification can be jointly done using the GCD algorithm described in section 4.2 $\square$

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 and corollary1, 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 phase 1 of the Boneh-Franklin protocol. First, it is a well-known fact that $\lim_{n\to\infty} \Pr_{(p',q')\in[1,n[^2}[\gcd(p',q') = 1] = \frac{6}{\pi^2} > 1/2$ assuming prime numbers in $[2^{\frac{n-1}{2}}, 2^{\frac{n}{2}}[$ are uniformly distributed. 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 $\Pr_{p'}[\gcd(2p', P') = 2] = \Pr_{p'}[2 \nmid p' \wedge 3 \nmid p' \wedge \ldots \wedge B \nmid p'] = (1-\frac{1}{2})(1-\frac{1}{3})\ldots(1-\frac{1}{B}) = \prod_{p_i \le B}(1-\frac{1}{p_i}) \approx \frac{1}{e^\gamma \ln(B)}$ 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

For security and robustness proofs see [4, 18, 8] or the appendix 9.3.

4.6 Distributed generation of the keys in Shoup's protocol

Once N is generated, let prime e be the first prime greater than $4\Delta^2$ so that each server can compute it. Then, Catalano *et al.* protocol's [8] is run to generate a shared secret key d in a distributed manner. At the end of the protocol, each server can compute its verification keys as $v_{u,i} = v_u^{\Delta d_i}$ for random v_u computed as $y_u^2 \bmod N$ where y_u is the concatenation of $H(N\|i)$ for sufficiently many i's to get the correct security parameter in the random oracle model.

5 Security of Shoup protocol against passive adversary

In this part we show that Shoup's protocol is secure with the RSA moduli generated in previous section against static and passive adversary.

5.1 Key Generation

At the end of the key generation protocol, we want to know if the information an adversary can collect, helps him to get useful information on the secret key d. Let $d_{i_1}, \ldots, d_{i_t}$, t shares of the secret key obtained by the adversary from the t correupted servers.

Information revealed by the shares d_{i_j}. For each d_{i_j}, we can note that

$$d_{i_j} = f(i_j) = a_0 + a_1 i_j + \ldots + a_t i_j^t \bmod M$$

If we add a $(t+1)^{\text{th}}$ equation, $a_0 = d$, we obtain the following linear system :

$$\begin{cases} a_0 + a_1 i_1 + \ldots + a_t {i_1}^t = d_{i_1} \bmod M \\ a_0 + a_1 i_2 + \ldots + a_t {i_2}^t = d_{i_2} \bmod M \\ \ldots \\ a_0 + a_1 i_t + \ldots + a_t {i_t}^t = d_{i_t} \bmod M \\ a_0 + 0 + \ldots + 0 = d \end{cases}$$

or with matrix $I.A = D \bmod M$

$$\begin{pmatrix} 1 & i_1 & i_1^2 & \ldots & i_1^t \\ 1 & i_2 & i_2^2 & \ldots & i_2^t \\ \ldots & & \ddots & & \ldots \\ 1 & i_t & i_t^2 & \ldots & i_t^t \\ 1 & 0 & 0 & \ldots & 0 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_t \end{pmatrix} = \begin{pmatrix} d_{i_1} \\ d_{i_2} \\ \vdots \\ d_{i_t} \\ d \end{pmatrix} \bmod M$$

The matrix I is a Vandermonde matrix. The determinant of such matrix is $\det(I) = \prod_{1 \le j < k \le t+1} (i_k - i_j) \bmod M$, where $i_{t+1} = 0$. As the i_j's are distinct in $\mathbb{Z}_M$, $i_k - i_j \neq 0 \bmod M$ since $\ell < B$. Hence, each factor $(i_k - i_j)$ is invertible modulo M, so $\det(I)$ is invertible modulo M. Therefore, all values of d are possible. Hence a group of t players cannot get information on d from shares of d. We see here that the sieving algorithm performed is important to avoid leakage of information on d. Consequently, $\ell < B$.

Information revealed from the verification keys. For all $u \in [1, k]$, the verification keys $v_{u,i}$'s of non-corrupted servers do not reveal any information as they can easily be simulated from the parts of the corrupted servers. The simulator chooses at random $y_u \in \mathbb{Z}_N{}^*$, computes $v_u = y_u^{2\Delta e} \bmod N$. Hence, $v_u^d = y_u^{2\Delta}$. We can note that $\Delta d_i = \sum_{k=1}^{t+1} \lambda_{i,k}^S d_j \bmod \varphi(N)$ if we denote by S a group of $t+1$ values and if we define the Lagrange coefficients as $\lambda_{i,j}^S = \Delta \times \prod_{j' \in S \backslash j} \frac{i-j'}{j-j'} \in \mathbb{Z}$. The simulator can then compute for all $u \in [1, k]$, $v_u^{\Delta d_i} = y_u^{2\Delta\lambda_{i,0}^S} \times \prod_{j=1}^{t} v_u^{\lambda_{i,i_j}^S d_{i_j}} \bmod N$, where $S = \{0, i_1, \ldots, i_t\}$. Hence a group of t players cannot get information from the validation keys of non-corrupted servers.

5.2 Signing protocol

To generate a signature on message m, each server i computes $x = H(m)$, $s_i = x^{2\Delta d_i} \bmod N$ and sends s_i to the combiner without any proof because we are in the honest-but-curious model. The combiner selects a set S of $t+1$ values and computes $w = \prod_{j=1}^{t+1} s_{i_j}^{2\lambda_{0,i_j}^S} \bmod N$. It then follows that $w^e = x^{4\Delta^2}$ as

$s_{i_j}^2 = x^{4\Delta d_{i_j}}$. Applying a well-known lemma, we can extract e-root of x from w which is a e-root of $x^{4\Delta^2}$ as $4\Delta^2$ is a known value using the extended Euclidean algorithm on e and $4\Delta^2$. As we are in the honest-but-curious model, the signature s is always correct.

We can prove the following theorem :

Theorem 3. *In the random oracle model, the signing protocol described above is a secure threshold signature scheme (non-forgeable) assuming the standard RSA signature scheme is secure.*

Proof. Similar to [37]. □

6 Enhancing the Shoup scheme against active adversary

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 section 4. This uses a method by which we generate the entire group of squares with few random elements with high probability.

6.1 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 ${s_i}^2$ in basis g exists, where $s_i = x^{2\Delta d_i}$ and $\Delta = \ell!$. As we will see in section 6.3, we can denote by $v_1, \ldots, v_k$ 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 $(a_1, \ldots, a_k) \in [0, M[^k$ such that $x = \prod_{i=1}^k {v_i}^{a_i} \bmod N$.

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

$$\log_{v_1}(v_{1,i}) = \ldots = \log_{v_k}(v_{k,i}) = \log_{x^{4\Delta}}({s_i}^2)(= d_i \bmod M)$$

The value ${s_i}^2$ 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 protocol for one round ($h = 1$).

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

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

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

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

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

6.2 Security analysis of the proof of correctness

Proof of Completeness.

Theorem 4. *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 $\geq (1 - \frac{B'M}{A})^h \geq 1 - \frac{B'Mh}{A}$. $\square$

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 ${v_1}^\sigma = {v_{1,i}}^\tau, \ldots, {v_k}^\sigma = {v_{k,i}}^\tau, x^{4\Delta\sigma} = {s_i}^{2\tau}$.*

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 ${v_1}^\sigma = {v_{1,i}}^\tau, \ldots, {v_k}^\sigma = {v_{k,i}}^\tau, x^{4\Delta\sigma} = {s_i}^{2\tau}$. $\square$

Theorem 5. ***(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 $x^{4\Delta d_i} = {s_i}^2$ and so s_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 $x^{4\Delta\sigma} = {s_i}^{2\tau}$.

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

Consequently if we raise this equation to the power σ, we obtain $x^{4\Delta\sigma} = {v_1}^{\sigma\beta_1} \times \ldots \times {v_k}^{\sigma\beta_k}$. But, $x^{4\Delta\sigma}$ is equal to ${s_i}^{2\tau}$ 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, ${s_i}^{2\tau} = (x^{4\Delta})^{\tau d_i}$ with $|\tau| < B'$. We can simplify this equation by τ if τ is coprime with $p'q'$. So we obtain $x^{4\Delta d_i} = {s_i}^2$ 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 6.3. 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

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

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. □

6.3 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 6. *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.

See appendix section 9.4 for a proof of this theorem.

7 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 $\Pr_{p'}[p'$ 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 $\Pr_{p'}[p'$ 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 the first phase of Boneh-Franklin's 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.

8 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 and make Shamir Secret Sharing Scheme secure
- 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. O. Baudron, P.A. Fouque, D. Pointcheval, G. Poupard, and J. Stern. Practical Multi-Candidate Election System. In *PODC '01*. ACM, 2001.
2. 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.
3. S. Blackburn, S. Blake-Wilson, S. Galbraith, and M. Burmester. Shared Generation of Shared RSA Keys. Technical report, University of Waterloo, Canada, February 1998. CORR-98-19.
4. D. Boneh and M. Franklin. Efficient Generation of Shared RSA keys. In *Crypto '97*, LNCS 1233, pages 425–439. Springer-Verlag, 1997.
5. 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.
6. R. Canetti, R. Gennaro, A. Herzberg, and D. Naor. Proactive Security : Long-term Protection Against Break-ins. *CryptoBytes*, 3(1), Spring 1997.
7. R. Canetti and S. Goldwasser. An Efficient Threshold Public Key Cryptosystem Secure Against Adaptive Chosen Ciphertext Attack. In *Eurorypt '99*, LNCS 1592, pages 90–106. Springer-Verlag, 1999.
8. 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.

9. C. Cocks. Split Knowledge Generation of RSA Parameters. In *Cryptography and Coding : 6th IMA Conference*, LNCS 1355, pages 89–95. Springer-Verlag, 1997.
10. C. Cocks. Split Generation of RSA Parameters with Multiple Participants. Technical report, CESG, 1998. Available at `http://www.cesg.gov.uk`.
11. I. Damgård and M. Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *PKC '01*, LNCS 1992, pages 119–136. Springer-Verlag, 2001.
12. I. Damgård and M. Koprowski. Practical Threshold RSA Signatures Without a Trusted Dealer. In *Eurocrypt '01*, LNCS 2045, pages 152–165. Springer-Verlag, 2001.
13. Y. Desmedt and Y. Frankel. Shared Generation of Authenticators and Signature. In *Crypto '91*, LNCS 576, pages 457–469. Springer-Verlag, 1991.
14. P. A. Fouque, G. Poupard, and J. Stern. Sharing Decryption in the Context of Voting or Lotteries. In *Financial Crypto '00*, LNCS. Springer-Verlag, 2000.
15. P. A. Fouque and J. Stern. One Round Threshold Discrete-Log Key Generation without Private Channels. In *PKC '01*, LNCS 1992. Springer-Verlag, 2001.
16. Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Optimal Resilience Proactive Public-Key Cryptosystems. In *FOCS '97*, pages 384–393, 1997.
17. Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Proactive RSA. In *Crypto '97*, pages 440–454, 1997.
18. Y. Frankel, P. MacKenzie, and M. Yung. Robust Efficient Distributed RSA Key Generation. In *STOC '98*, pages 663–672, 1995.
19. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and Efficient Sharing of RSA Functions. In *Crypto '96*, LNCS 1109, pages 157–172. Springer-Verlag, 1996.
20. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust Threshold DSS Signatures. In *Eurocrypt '96*, LNCS 1070, pages 425–438. Springer-Verlag, 1996.
21. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In *Eurocrypt '99*, LNCS 1592, pages 295–310. Springer-Verlag, 1999.
22. 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.
23. N. Gilboa. Two Party RSA Key Generation. In *Crypto '99*, LNCS 1666. Springer-Verlag, 1999.
24. L. C. Guillou and J.-J. Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. In *Eurocrypt '88*, LNCS 330, pages 123–128. Springer-Verlag, 1988.
25. B. King. Improved Methods to Perform Threshold RSA. In *Asiacrypt '00*, LNCS 1976, pages 359–372. Springer-Verlag, 2000.
26. S. Miyazaki, K. Sakurai, and M. Yung. On Threshold RSA-signing with no dealer. In *ICICS '99*, LNCS 1787. Springer-Verlag, 1999.
27. T.P. Pedersen. A Threshold Cryptosystem without a Trusted Party. In *Eurocrypt'91*, LNCS 547, pages 522–526. Springer-Verlag, 1991.
28. C. Pomerance. On the distribution of pseudoprimes. In *Mathematics of Computation*, 37(156), pages 587–593, 1981.
29. C. Pomerance. Two methods in elementary analytic number theory. pages 135–161. Kluwer Academic Publishers, 1989.
30. G. Poupard and J. Stern. Generation of Shared RSA Keys by Two Parties. In *Asiacrypt '98*, LNCS 1514, pages 11–24. Springer-Verlag, 1998.

31. G. Poupard and J. Stern. Short Proofs of Knowledge for Factoring. In *PKC '00*, LNCS 1751, pages 147–166. Springer-Verlag, 2000.
32. T. Rabin. A Simplified Approach to Threshold and Proactive RSA. In *Crypto '98*, LNCS 1462, pages 89–104. Springer-Verlag, 1998.
33. R. Rivest. Finding Four Million Large Random Primes. In *Crypto '90*, LNCS 537, pages 625–626. Springer-Verlag, 1991.
34. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
35. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *STOC '94*, pages 522–533. ACM, 1994.
36. A. Shamir. How to Share a Secret. *Communications of the ACM*, 22:612–613, November 1979.
37. V. Shoup. Practical Threshold Signatures. In *Eurocrypt '00*, LNCS 1807, pages 207–220. Springer-Verlag, 2000.
38. V. Shoup and R. Gennaro. Securing Threshold Cryptosystems against Chosen Ciphertext Attack. In *Eurocrypt '98*, LNCS 1403, pages 1–16. Springer-Verlag, 1998. cf. the extended version for the Journal of Cryptology, available at `http://www.shoup.net/papers/`.
39. R.D. Silverman. Fast Generation of Random, Strong RSA Primes. RSA Laboratories, May 1997.

9 Appendix

9.1 Robustness of the distributed sieving protocol

To resist against such players, we first run a "sum-to-poly" algorithm as described in [16, 18]. When the polynomial sharing of p is obtained, one can note that $\sum_{j\in S\setminus\{0\}} \lambda^S_{0,j} = 1$, where $\lambda^S_{0,j}$ denotes the Lagrange coefficient of the jth server. Therefore, server i can set its new polynomial share to $f(i) - 1$ if $f(i)$ denotes its polynomial share. Indeed, if $p = f(0) = \sum_{j\in S\setminus\{0\}} \lambda^S_{0,j} f(j)$, then

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

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

9.2 Proof of lemma 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 $f'^{\beta}|\gcd(a,b)$. □

9.3 Proof of security

Theorem 7. *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 [4], 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 8. *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 $1/n^\delta$. Then there exists an expected polynomial time algorithm $\mathcal{B}$ that factors $1/4(t+1)^3 n^\delta$ of the integers in $\mathbb{Z}_n^{(2)}$* [1].

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 $1/4(e^\gamma \ln(B))$. Hence, the result is still valid by replacing $1/4(t+1)^3 n^\delta$ by $1/16(e^\gamma \ln(B))(t+1)^3 n^\delta$.

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.

---

[1] We can note that the distribution of these numbers is not uniform in $[2^{\frac{n-1}{2}}, 2^{\frac{n}{2}}[$ since the random variable is the sum of ℓ independent and uniform random variables of mean $\frac{2^{\frac{n-3}{2}}/\ell+1}{2}$ and variance $\frac{2^{\frac{n-3}{2}}/\ell^2-1}{12}$. It is well-known that the sum of uniform random variables tends to a gaussian variable according to the Central Limit Theorem of variance $V = \frac{2^{\frac{n-3}{2}}/\ell-\ell}{12}$. We can compute its mean and variance to determine it. It can be shown that $V > 2^{128} \times \log(2^{128})$ which represents the size of an interval that contains at least 2^{128} primes.

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.

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

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

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

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 [18] where they make the Boneh and Franklin protocol robust against malicious t-adversary. The other result comes from the Catalano *et al.* proof of [8] 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. □

9.4 Proof of theorem 6

In this appendix, we prove the theorem 6.

Theorem 5. *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 $\sum_{d=B}^{+\infty} \frac{1}{d^k}$.

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

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

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 $\geq 1/26$, $\mathsf{Pr} \leq \frac{1}{2^{17}}$. 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 [31] 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 $\varphi_k(p')$.*

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

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, the number of k-tuples of $(\mathbb{Z}_{p'})^k$ such that $\gcd(\alpha_1, \ldots, \alpha_k, p') = 1$ is $\prod_{i=1}^{t'}(q_i{}^{kf_i} - q_i{}^{k(f_i-1)}) = \prod_{i=1}^{t'} \varphi_k(q_i{}^{f_i}) = \varphi_k(p')$ since φ_k is multiplicative. □

Now, we return back to the proof of the theorem 6. 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, $\mathsf{pr} = (\prod_{i \in S_{p'}} \sum_{\alpha_i=0}^{\infty} \frac{1}{p_i{}^{\alpha_i k}})^{-1}$ 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 $\mathsf{pr} > 1/\zeta(k)$.

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

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

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

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 $1 - 1/2^{80}$.