Generation of Shared RSA Keys by Two Parties

Guillaume Poupard and Jacques Stern

École Normale Supérieure, Laboratoire d'informatique
45 rue d'Ulm, F-75230 Paris Cedex 05, France
email: {Guillaume.Poupard,Jacques.Stern}@ens.fr

Abstract. At Crypto'97 Boneh and Franklin proposed a protocol to efficiently generate shared RSA keys. In the case of two parties, the drawback of their scheme is the need of an independent third party. Furthermore, the security is guaranteed only if the three players follow the protocol. In this paper, we propose a protocol that enables two parties to evaluate any algebraic expression, including an RSA modulus, along the same lines as in the Boneh-Franklin protocol. Our solution does not need the help of a third party and the only assumption we make is the existence of an oblivious transfer protocol. Furthermore, it remains robust even if one of the two players deviates from the protocol.

1 Introduction

The general problem of private multi-party computation has motivated many solutions for ten years. In 1986, Yao [26] proved the existence of secure two-party protocols assuming the computational intractability of factoring large integers. Goldreich, Micali and Wigderson [16] generalized this result and showed that trapdoor functions enable to evaluate any function whose inputs are privately owned by the parties, provided a majority of them is honest. The next year, Ben-Or, Goldwasser and Wigderson [2] and independently Chaum, Crépeau and Damgård [8] solved the same problem under an information-theoretic approach. Then, many papers improved the needed assumptions [18], the theoretical bounds for subclasses of functions [9] or the simplicity and the efficiency of the methods [13].

The aim of all those papers is to solve any multi-party computation problem. Accordingly the first step is always the description of the function to be privately evaluated as a logical circuit or as a polynomial over a finite field. This enables to reduce the problem to a very small set of elementary protocols, like the computation of the logical AND of two bits, at the cost of polynomial but unpractical solutions. Consequently, even if the problem of multi-party computation is theoretically solved, the design of more specific but also more efficient protocols appears necessary.

Boneh and Franklin [5] followed this "application oriented" approach to solve the problem of generating shared RSA keys. More precisely, some parties want to jointly generate an RSA modulus $N = pq$ where p and q are prime in such a way that, at the end of the computation, the parties are convinced that N is indeed a product of two large primes but none of them knows its factorization. They use the general protocol of Ben-Or, Goldwasser and Wigderson [2] to prove that the distributed computation of N by two parties can be efficiently done with the help of a third party, assuming the three players do not collude and follow the protocol. They also prove that the test $N = pq$ with p and q two prime numbers can be efficiently done by two honest parties alone, using a clever probabilistic algorithm variant of the Miller-Rabin and the Solovay-Strassen ones, under the assumption that the quadratic residuosity problem is computationally hard to solve. Finally, they show how two parties can generate shared secret keys, by themselves for small public exponents and with the help of a third party in the general case. An experimental evaluation of the performance can be found in [24]. It shows that a 1024-bit modulus N can be generated in only 10 minutes with Sparc 20 machines.

Independently Cocks [10,11] has proposed another solution for the same problem that only involves two honest players but assuming the computational intractability of a problem weaker than RSA. This protocol is analyzed and improved in [4].

More recently, Frankel, MacKenzie and Yung [15] have improved the security of the Boneh-Franklin protocol. Their generation scheme is efficient and robust even when a minority of parties are malicious. But in the case of only two parties, both of them have to be honest.

Our Results

For many applications, the protocol of Boneh and Franklin does not provide an accurate level of security for two reasons. Firstly, it needs an independent and honest third party. Secondly, the security is guaranteed only if the three players do not deviate from the protocol. In this paper, we show how two parties can efficiently generate shared RSA keys even if one of them is dishonest. From a theoretical point of view, we only assume the existence of oblivious transfer protocols.

From a practical point of view, our scheme is less efficient than the Boneh-Franklin protocol based on the Ben-Or, Goldwasser and Wigderson construction [2] which is itself based on arithmetical computation in finite fields. Anyway our protocol is much more efficient, especially when we focus on the number of rounds of communication, than those derived from general techniques.

The paper is organized as follows: we first recall the notion of an ANDOS protocol. Then we propose an efficient and general protocol for the distributed evaluation of algebraic expressions by two parties. We prove its security when the players are honest but also its robustness when one of them is malicious. Then, we use this protocol to generate shared RSA keys. Finally we compare the efficiency of our scheme with general 2-party computation protocols. We also propose in an appendix another solution, much more simple and efficient but less general, based on the higher residue cryptosystem of Naccache and Stern [19].

All-or Nothing Disclosure of Secrets Protocols

Oblivious transfer has been introduced in 1981 by Rabin [21] and a specific version, namely the oblivious transfer of one bit out of two [14], soon emerged as a very useful cryptographic primitive for many applications [18]. Brassard, Crépeau and Robert generalized this notion in various natural ways [7] and they proved the equivalence of those protocols in an information theoretic sense [6].

All-or Nothing Disclosure of Secrets (ANDOS) protocols [7] address the following problem: a *merchant* has n secret bit-strings and wishes to sell one of them to a *buyer* who has the ability to choose which one he wants. There are two privacy requirements: the merchant does not want the buyer to obtain information about any other secret and the buyer does not want the merchant to learn anything about the string he has chosen.

Oblivious transfer, and consequently ANDOS, can be based on various assumptions like the existence of trapdoor functions [16], of noisy channels [12] or of quantum channels [3]. From a practical point of view, efficient implementation can be based on the quadratic residuosity problem [7,25] or on the Diffie-Hellman assumption [1,23].

In this paper we use ANDOS as a cryptographic primitive. In order to formalize its properties, let us consider that Alice sells a secret to Bob. Using the terminology of [17], we define the *view* of Alice to be everything she sees during the execution of the protocol. Let $View_A$ be the random variable whose value is this view. It depends on the secrets $s_1, ...s_n$ sold by Alice, on the index i_B of the secret bought by Bob and on the random tape ω_A of Alice considered as a polynomial time Turing machine. We also use the three well-known notions of indistinguishability of random variables: the perfect one, the statistical one and the computational one (see [17] for complete definitions). In the paper, we just talk about indistinguishability without any other precision for simplicity reasons but all the definitions and proofs hold in the three models and the choice of one of them only depends on the properties of the underlying ANDOS.

We model the ANDOS protocol as a scheme that enables Alice to sell one secret out of n to Bob in such a way that:
(ANDOS$_1$) Alice does not learn anything about the index i_B of the secret she has sold:

$$\begin{gathered}\forall j \in [1,n] \quad View_A(\omega_A, s_1, ...s_n, i_B) \\ \text{is indistinguishable from } View_A(\omega_A, s_1, ...s_n, j)\end{gathered}$$

(ANDOS$_2$) Bob does not learn anything about the other secrets:

$$\begin{gathered}\forall s'_1, ...s'_n \text{ such that } s'_{i_B} = s_{i_B} \quad View_B(\omega_B, s_1, ...s_n, i_B) \\ \text{is indistinguishable from } View_B(\omega_B, s'_1, ...s'_n, i_B)\end{gathered}$$

2 Efficient Two-Party Evaluation of Algebraic Expressions

The General Problem

Let us consider two players, Alice and Bob, modelled as polynomial time Turing machines, who have private randomly chosen data $d_A \in \mathcal{E}_A$ and $d_B \in \mathcal{E}_B$. We want to design a two-party computation protocol which enables Alice and Bob to compute a public function $f(d_A, d_B)$ (whose result is encoded as an integer value) modulo a prime public modulus P.

This protocol has to meet two main properties. Informally, it must be *correct*, i.e. the result of the computation must be $f(d_A, d_B) \bmod P$. It must also be *private*, i.e. a party must not be able to learn information about the other's secret. The exact meaning of those two properties will be made precise further on. At the moment, let us stress that we do not develop a general two-party computation protocol but just an efficient scheme suitable to the generation of shared RSA keys. Consequently we need weaker notions of privacy and correctness than those described in more general papers [2,8,16,26].

The Protocol

Let us consider polynomial size sets $\mathcal{E}_A$ and $\mathcal{E}_B$ and any prime modulus P. The following protocol enables Alice and Bob to compute $f(d_A, d_B) \bmod P$ without revealing there private inputs $d_A \in \mathcal{E}_A$ and $d_B \in \mathcal{E}_B$:

(1) Alice randomly chooses $(\alpha_A, \beta_A) \in \mathbb{Z}_P^* \times \mathbb{Z}_P$ (the coefficients of a secret line).
(2) Alice and Bob perform an ANDOS protocol where Alice sells $\{\gamma_d\}_{d \in \mathcal{E}_B} = \{\alpha_A \times f(d_A, d) + \beta_A \bmod P\}_{d \in \mathcal{E}_B}$ and Bob buys $\gamma_{d_B} = y_B$.
(3) Bob randomly chooses $(\alpha_B, \beta_B) \in \mathbb{Z}_P^* \times \mathbb{Z}_P$.
(4) Alice and Bob perform an ANDOS protocol where Bob sells $\{\delta_d\}_{d \in \mathcal{E}_A} = \{\alpha_B \times f(d, d_B) + \beta_B \bmod P\}_{d \in \mathcal{E}_A}$ and Alice buys $\delta_{d_A} = y_A$.
(5) Alice and Bob broadcast (simultaneously) (α_A, β_A, y_A) and (α_B, β_B, y_B).
(6) They verify $\alpha_A \in \mathbb{Z}_P^*$, $\alpha_B \in \mathbb{Z}_P^*$ and $(y_B - \beta_A) \times {\alpha_A}^{-1} = (y_A - \beta_B) \times {\alpha_B}^{-1} \bmod P$. If this equality holds, $f(d_A, d_B) = (y_A - \beta_B) \times {\alpha_B}^{-1} \bmod P$; we say that the protocol ends *successfully*. Otherwise, the protocol fails and the players stop cooperation.

Security Analysis for Honest Players

We first consider that Alice and Bob behave honestly, i.e. follow the protocol.

Theorem 1 (Correctness). *If the two players follow the protocol, it always succeeds and both of them obtain the correct value $f(d_A, d_B) \bmod P$.*

Proof. The correctness of the protocol when the two players are honest is obvious according to the graphical representation of figure 1. □

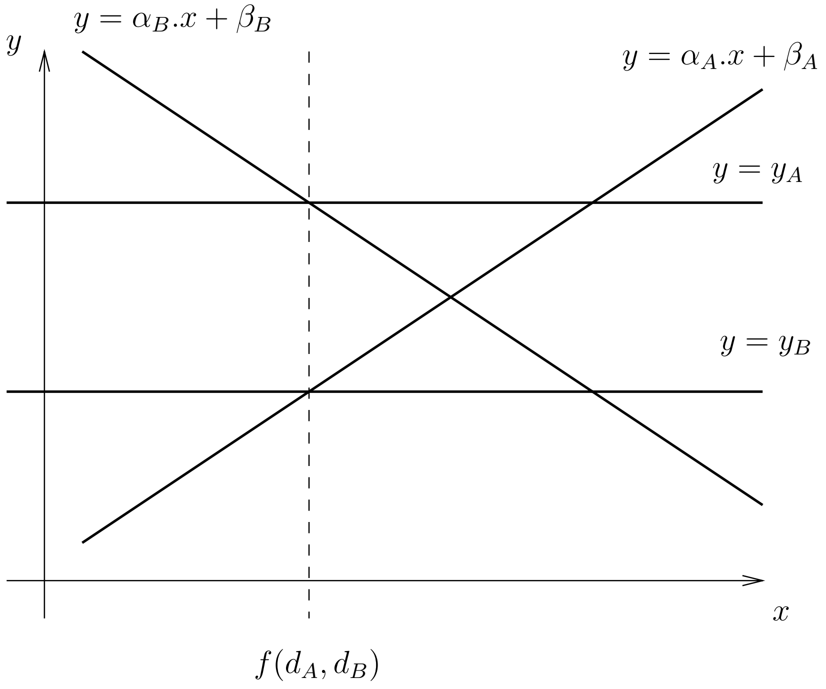

Fig. 1. Graphical representation of the two-party computation protocol

Theorem 2 (Privacy). *Given* $f(d_A, d_B) \bmod P$ *and their own private data, Alice and Bob can each simulate the transcript of the protocol. Consequently they learn nothing more than the value* $f(d_A, d_B) \bmod P$.

Proof. We show how to simulate Alice's view but the same proof holds for Bob. The simulator randomly chooses $\alpha_B \in \mathbb{Z}_P^*$, $\beta_B \in \mathbb{Z}_P$, $\delta_d \in \mathbb{Z}_P$ for all $d \in \mathcal{E}_A - \{d_A\}$. It computes $\delta_{d_A} = y_A = \alpha_B \times f(d_A, d_B) + \beta_B \bmod P$ and $\gamma_d = \alpha_A \times f(d_A, d) + \alpha_B \bmod P$ for all $d \in \mathcal{E}_B$ (including $y_B = \gamma_{d_B}$). It then randomly choose $d \in \mathcal{E}_B$ and simulates the buying of the secret γ_d by Bob. The property (Andos$_1$) shows that the view of Alice during the simulation is indistinguishable from what she sees when Bob really buys γ_{d_B}. It also simulates the buying of $\delta_{d_A} = y_A$ by Alice. Property (Andos$_2$) proves that the view of Alice is indistinguishable from her view when the secrets $\{\delta_d\}_{d \in \mathcal{E}_A - \{d_A\}}$ are really computed by Bob. Finally, the simulator reveals α_A, β_A, y_A, α_B, β_B, y_B whose distribution is the same as in a real interaction between Alice and Bob. □

Security Analysis when one Player is Malicious

We only consider the situation where Alice is malicious and Bob honest. For the reverse case, even though the protocol is not symmetrical for the two players, the following proofs remain valid because they do not use the order of the steps (1) to (4).

Let us first recall a useful probabilistic lemma [20]:

Lemma 3. *Let* $A \subset X \times Y$ *such that* $\Pr\{A(x,y)\} = \varepsilon$ *and, for any* $\alpha < \varepsilon$, *let* $X_0 = \{a \in X / \Pr\{A(x,y)/x = a\} > \varepsilon - \alpha\}$. *Then* $\Pr\{x \in X_0\} \geq \alpha$.

Proof. Using the Bayes law, the probability $\varepsilon = \Pr_{(x,y)\in X\times Y}\{A(x,y)\}$ is equal to $\Pr\{x \in X_0\}\Pr\{A(x,y)/x \in X_0\} + \Pr\{x \notin X_0\}\Pr\{A(x,y)/x \notin X_0\}$ and this is less than $\Pr\{x \in X_0\} + \sum_{a \notin X_0}\Pr\{x = a\}\Pr\{A(x,y)/x = a\}$.
So $\varepsilon \leq \Pr\{x \in X_0\} + \sum_{a \notin X_0}\Pr\{x = a\}(\varepsilon - \alpha)$. □

For the analysis of the security of the protocol when Alice is malicious, we note $\widetilde{\gamma}_d$ what she sells to Bob during the first ANDOS and $\widetilde{y_A}$ the value she broadcasts at step (5). Such a notation enables to distinguish potentially false values from those computed according to the protocol. The tuple $(\{\widetilde{\gamma}_d\}_{d\in\mathcal{E}_B}, \alpha_A, \beta_A, \widetilde{y_A})$ is simply noted t and Δ denotes the size of $\mathcal{E}_B$. Finally, we often omit the modular reduction mod$P$ for simplicity reasons but all the computation are performed in $\mathbb{Z}_P$.

Definition 4. *For fixed values of* d_A, d_B *and* t, *Alice is said to be* pseudo-honest *(*PH*) if* $\widetilde{\gamma}_{d_B} = \alpha_A \times f(d_A, d_B) + \beta_A \bmod P$ *and* $\widetilde{y_A} = y_A$.

Definition 5. *For fixed values* d_A *and* d_B, *the predicate* $success_{d_A,d_B}(t, \alpha_B, \beta_B)$ *is true if the protocol ends successfully i.e. if* $\alpha_A \neq 0$ *and* $(\widetilde{\gamma}_{d_B} - \beta_A) \times {\alpha_A}^{-1} = (\widetilde{y_A} - \beta_B) \times {\alpha_B}^{-1} \bmod P$.

Before proving the correctness of the protocol, let us state a lemma whose proof comes from elementary algebra arguments and that essentially says that the intersection of two non-parallel lines is reduced to one point in $(\mathbb{Z}_P)^2$.

Lemma 6. *For fixed values of* d_A, d_B, y_A *and for a given tuple* t, *if there exists two different pairs* (α_B^1, β_B^1) *and* (α_B^2, β_B^2) *such that* $y_A = \alpha_B^1 \times f(d_A, d_B) + \beta_B^1 = \alpha_B^2 \times f(d_A, d_B) + \beta_B^2$ *and* $success_{d_A,d_B}(t, \alpha_B^i, \beta_B^i)$ *for* $i \in \{1, 2\}$, *then Alice is pseudo-honest.*

Lemma 7. *If Alice has a cheating strategy such that the protocol ends successfully with probability* ε, *she is pseudo-honest with probability greater than* $\varepsilon - \frac{1}{1-P}$.

Proof. The probability distribution of $t = (\{\widetilde{\gamma}_d\}_{d\in\mathcal{E}_B}, \alpha_A, \beta_A, \widetilde{y_A})$ a priori depends on d_A, d_B, α_B and β_B. Property ANDOS$_2$ applied to the second ANDOS (where Alice buys $y_A = \alpha_B \times f(d_A, d_B) + \beta_B \bmod P$ to Bob) shows that for a fixed value y_A, the distribution of t does not depend of α_B and β_B such that $y_A = \alpha_B \times f(d_A, d_B) + \beta_B \bmod P$. Consequently, this distribution only depends on $(d_A, d_B, y_A) = s$. Let us note $\mathcal{D}$ the distribution of the pairs (α_B, β_B) such that $y_A = \alpha_B \times f(d_A, d_B) + \beta_B \bmod P$ and $\mathcal{T}$ the (non uniform) distribution of tuple t for fixed values of d_A, d_B and y_A. Let ε be the probability of success of the protocol according to the cheating strategy of Alice and ε_s be this probability for fixed values of d_A, d_B and y_A; $\varepsilon = \sum_s \Pr\{s\}\varepsilon_s$ and $\varepsilon_s = \Pr_{(\alpha_B,\beta_B)\in\mathcal{D},t\in\mathcal{T}}\{success_{d_A,d_B}(t,\alpha_B,\beta_B)\}$. Let $X_s$be the set of the tuples t such that $\Pr_{(\alpha_B,\beta_B)\in\mathcal{D}}\{success_{d_A,d_B}(t,\alpha_B,\beta_B)\} > \frac{1}{P-1}$. Lemma 3, with $\alpha = \varepsilon_s - \frac{1}{P-1}$, proves that $\Pr_{t\in\mathcal{T}}\{t \in X_s\} \geq \varepsilon_s - \frac{1}{P-1}$.

For fixed values of d_A, d_B and y_A, there are exactly $P-1$ pairs (α_B, β_B) such that $y_A = \alpha_B \times f(d_A, d_B) + \beta_B \bmod P$ and the distribution $\mathcal{D}$ of those pairs is uniform since Bob chooses them randomly. If t belongs to X_s, the probability $\Pr_{(\alpha_B,\beta_B)\in\mathcal{D}}\{success_{d_A,d_B}(t,\alpha_B,\beta_B)\}$ is greater than $\frac{1}{P-1}$ so there exists two different pairs (α_B^1, β_B^1) and (α_B^2, β_B^2) such that $y_A = \alpha_B^i \times f(d_A, d_B) + \beta_B^i \bmod P$ and $success_{d_A,d_B}(t, \alpha_B^i, \beta_B^i)$ for $i \in \{1, 2\}$. According to lemma 6, this proves that Alice is pseudo-honest.

We can now evaluate the probability for Alice to be pseudo-honest:

$$\sum_s \Pr\{s\} \Pr_{t\in\mathcal{T}}\{t \in X_s\} \geq \sum_s \Pr\{s\}\left(\varepsilon_s - \frac{1}{P-1}\right) = \varepsilon - \frac{1}{P-1}$$

□

Theorem 8 (Correctness). *Assume Alice has a cheating strategy such that the protocol ends successfully with probability ε. If an execution of the protocol is successful, the probability for the result to be $f(d_A, d_B)$ is greater than $\frac{\varepsilon - \frac{1}{P-1}}{\varepsilon}$.*

Proof. The probability for Alice to be pseudo-honest conditioned by the knowledge that the protocol ends successfully is $\Pr\{success(t, \alpha_B, \beta_B)/\text{Alice PH}\} \times \Pr\{\text{Alice PH}\} / \Pr\{success(t, \alpha_B, \beta_B)\}$.

Lemma 7 proves $\Pr\{\text{Alice is pseudo-honest}\} \geq \varepsilon - \frac{1}{P-1}$, by definition the probability of success is ε and finally $\Pr\{Success/\text{Alice PH}\} = 1$ because when Alice is pseudo-honest the protocol is successful so the result of a successful execution is correct with probability $\geq 1 - \frac{1}{\varepsilon(P-1)}$. □

Dealing with multiparty computation, an important characteristic is how fair the protocol is. During the shared generation of RSA keys, neither Alice nor Bob can take advantage to stop the interaction before the normal end because such keys cannot be used alone. So our protocol is unfair but it does not matter since our aim is not to design a general multiparty computation scheme but rather to obtain a scheme with no more properties than those needed for the generation of shared RSA keys.

Lemma 9. *The knowledge of $f(d_A, d_B) \bmod P$ enables Alice to simulate the transcript of successful executions of the protocol.*

Proof. We already said in lemma 7 that the probability distribution of t depends on y_A. Furthermore, for randomly chosen $(\alpha_B, \beta_B) \in \mathbb{Z}_P^* \times \mathbb{Z}_P$ and for any fixed value of $f(d_A, d_B)$, the distribution of $y_A = \alpha_B \times f(d_A, d_B) + \beta_B \bmod P$ is uniform.

Most of the simulation of Alice's view is the same as in the case of honest players (theorem 2). For fixed value of $f(d_A, d_B)$, the view of Alice during the first ANDOS is simulated by the buying of $\widetilde{\gamma}_{d_0}$ for a random value d_0. Her view during the second ANDOS is simulated by the buying of $y_A = \delta_{d_A} = \alpha_B f(d_A, d_B) + \beta_B \bmod P$ for randomly chosen α_B, β_B and δ_d for $d \neq d_A$. Then the simulator broadcasts α_A, β_A, $\widetilde{y_A}$, α_B, β_B, y_B, where $y_B = \widetilde{\gamma}_d$ for a randomly chosen d. Finally the simulator is reset until the verification succeeds. □

Theorem 10 (Privacy). *Assume Alice has a cheating strategy such that the protocol ends successfully with probability ε. After a successful execution of the protocol, Alice cannot learn more than $\frac{1}{\varepsilon(P-1)} \log \Delta - \frac{\varepsilon - \frac{1}{P-1}}{\varepsilon} \log\left(\varepsilon - \frac{1}{P-1}\right)$ bits of information about d_B in addition to the result $f(d_A, d_B)$.*

Proof. Let $\nu_{d_A,t}$ be the random variable equal to the number of $\widetilde{\gamma}_d$ correctly computed by Alice according to the revealed line $y = \alpha_A.x + \beta_A \bmod P$, i.e. such that $\widetilde{\gamma}_d = \alpha_A f(d_A, d) + \beta_A \bmod P$. For fixed d_A and t, the probability for Alice to be pseudo-honest is exactly ν/Δ if she reveals $\widetilde{y_A} = y_A$ so the probability ε' for Alice to be pseudo-honest is less than $\sum_{d_A,t} \Pr\{d_A, t\}\nu/\Delta$. Furthermore, if the protocol is successful, she exactly learns that $\widetilde{\gamma}_{d_B}$ has been correctly computed and consequently learns that d_B belongs to a set of size ν. In order to estimate the information Alice learns in addition to the result $f(d_A, d_B)$, we evaluate the expected value of $\log \nu$ in case of success, $E(\log \nu/success) = \frac{1}{\varepsilon} \sum_{d_A,t} \Pr\{d_A, t\} \frac{\nu}{\Delta} \log \nu$. A convexity inequality applied to the function $F(x) = x \log x$ shows that

$$E(\log \nu/success) \geq \frac{1}{\varepsilon\Delta} F(\varepsilon'\Delta) \geq \frac{\varepsilon - \frac{1}{P-1}}{\varepsilon} \left(\log(\varepsilon - \frac{1}{P-1}) + \log \Delta\right)$$

□

Furthermore, it is important to notice that, if the final verification fails, Bob is convinced that Alice has tried to cheat because the protocol is always successful when the players behave honestly.

Theorem 10 does not prove strict privacy because, with non negligible probability, a malicious player can obtain a few bits about the other player's secret without being caught. But, if we consider Alice and Bob as polynomial time Turing machines and if the probability of success is non-negligible, Alice does not learn much more information than what she could have guessed. More precisely, for example in the case of the generation of an RSA modulus N, if the knowledge of about $-\log \varepsilon$ bits of information enables Alice to factorize N in polynomial time, we can use her to factorize N in polynomial time without any other information.

When P is large enough, the previous results are simpler:

Theorem 11. *Assume Alice has a cheating strategy such that the protocol ends successfully with probability ε. If $\frac{1}{P-1} = o(\varepsilon)$, the result of a successful execution is correct and Alice cannot learn more than $-\log \varepsilon$ bits of Bob's secret in addition to $f(d_A, d_B)$.*

Special Case of Algebraic Expressions

The protocol has been stated for polynomial size sets $\mathcal{E}_A$ and $\mathcal{E}_B$. When f is an algebraic expression in $\mathbb{Z}_M$ and the inputs d_A and d_B are tuples of elements of $\mathbb{Z}_M$, if M can be factored in small relatively prime factors $M = \prod_{i=1}^{k} m_i$, with k and m_i polynomial in the security parameter, the protocol can also be used, even though M is not polynomial.

Instead of performing the protocol previously described with the large modulus M, we can use it k times with each modulus m_i. Finally, if $P > m_i$, Alice and Bob obtain $f(d_A, d_B) \bmod m_i$ for all i and the result $f(d_A, d_B) \bmod M$ is computed with the Chinese remainder theorem. The more M can be factored in relatively prime factors, the more the protocol is efficient. Consequently, as much as possible, we use a modulus M equal to the product of the first k prime numbers. Notice that theorem 10 can be generalized because if Alice learns less than $-\log \varepsilon_i$ bits of information with probability ε_i at round i, she learns less than $\sum_i -\log \varepsilon_i = -\log\left(\prod_i \varepsilon_i\right) = -\log \varepsilon$ with probability $\prod_i \varepsilon_i = \varepsilon$.

3 Computation of Shared RSA Keys

The computation of shared RSA keys by two parties can be efficiently performed using the protocol of the previous section. The first step consists in computing a candidate $N = (p_A + p_B) \times (q_A + q_B)$ and then to test whether N is the product of two prime numbers. Such a test has be proposed by Boneh and Franklin. Then, the second part of the generation consists in computing a shared secret key associated with a public exponent e.

3.1 Computation of the Modulus N

Let n be the size of the modulus we want to generate and $\mathcal{E}_A = \mathcal{E}_B = [0, 2^{n/2-1}[^2$ be the range where Alice and Bob randomly choose their private input $d_A = (p_A, q_A)$ and $d_B = (p_B, q_B)$. They want to compute $f((p_A, q_A), (p_B, q_B)) = (p_A + p_B) \times (q_A + q_B) = N$. We choose M as the smallest product of the first prime numbers greater than 2^n. Consequently, the result of the computation modulo M is the same as if the computation were done with integers. The function f is an algebraic expression so that we can use the efficient protocol described in section 2. This solves the problem of the efficient computation a shared RSA modulus by only two parties, even if one of them is malicious.

3.2 Trial Division Test

Since Alice and Bob first choose their private data, compute N and, only afterwards, test that $p_A + p_B$ and $q_A + q_B$ are indeed prime numbers, the generation procedure has to be repeated about $n^2/4$ times in order to obtain an RSA modulus N. Boneh and Franklin have proposed to perform a trial division test just after the random choice of p_A and p_B to check that $p_A + p_B$ is not divisible by a small prime number. This allows a reduction of the number of trials and consequently of the complexity of the generation.

We can use our protocol again to test if a small prime number p divides $p_A + p_B$, just taking $d_A = p_A \bmod p$, $d_B = p_B \bmod p$, $\mathcal{E}_A = \mathcal{E}_B = \mathbb{Z}_p$ and $f(x, y) = 0$ if $x + y = 0 \bmod p$ and $f(x, y) = 1$ otherwise. If during one trial division test the result 1 is obtained, Alice and Bob try again with new values p_A and p_B. Consequently, if the test succeeds, Alice only learns that $p_A + p_B \neq 0 \bmod p$, and she would have learned it anyway after the test $N = pq$.

In addition to the generic cheating strategy analyzed in theorem 10, Alice can use an input value $\widetilde{p_A}$ different from $p_A \bmod p$ as input. If she does this, she learns that $\widetilde{p_A} + p_B \neq 0 \bmod p$, i.e $\log(p)/p$ bits of p_B. Since Bob cannot know if she tried to cheat, Alice can make the protocol restart until she learns as much information as possible. If we note $\mathcal{P}$ the set of tested prime numbers, the information learned by Alice if she is malicious is less than $\sum_{p \in \mathcal{P}} \log(p)/p$ bits. If $\mathcal{P}$ is the set of the first ℓ prime numbers, this leads to a maximal amount of information less than $\log(\ell \ln \ell)$. As an example, for $n = 1024$ one can test the first 200 prime numbers as it is adviced in [24]. With our protocol, Alice can learn at most 9 bits of information about p_B.

3.3 Efficiency Improvement

A more efficient and more secure way to choose secret data that have more chance to lead to an RSA modulus consists in choosing p_A and p_B (resp. q_A and q_B) such that $p_A + p_B$ is not divisible by a very small prime number. More precisely, let M' be a product of the first odd prime numbers such that $M' \approx 2^{n/2-1}$. The choice of p_A and p_B by Alice and Bob is performed as follows:

(1) Alice randomly chooses $p'_A \in \mathbb{Z}^*_{M'}$, $p_A \in \mathbb{Z}_{M'}$,
(2) Bob randomly chooses $p'_B \in \mathbb{Z}^*_{M'}$, $\rho_B \in \mathbb{Z}_{M'}$,
(3) Alice and Bob perform a protocol as described in section 2 with
$d_A = (p'_A, p_A)$, $d_B = (p'_B, \rho_B)$ and
$f((p'_A, p_A), (p'_B, \rho_B)) = p'_A \times p'_B - p_A - \rho_B \bmod M'$,
(4) Bob obtains the value δ and computes $p_B = \delta + \rho_B \bmod M'$.

This protocol enables Alice and Bob to privately and efficiently obtain p_A and p_B such that none of the first prime numbers divides $p_A + p_B$.

Alice could try to cheat using $p'_A \notin \mathbb{Z}^*_{M'}$ but the design of the two-party computation protocol obliges Alice and Bob to input data in $\mathbb{Z}^*_{M'} \times \mathbb{Z}_{M'}$. Furthermore, the knowledge of $p_B = p'_A \times p'_B - p_A \bmod M'$ does not help Bob to learn more than $p_A + p_B \in \mathbb{Z}^*_{M'}$ because $\forall p'_B \in \mathbb{Z}^*_{M'}$ $p'_B \times \mathbb{Z}^*_{M'} = \mathbb{Z}^*_{M'}$. After this preliminary step, Bob could use a different p_B but this would just reduce the efficiency of the protocol and cannot be used as a way to cheat. The aim of this computation is just to help Bob in choosing a reasonable p_B.

In conclusion, an efficient strategy to compute a good N consists in generating p_A and p_B with this protocol, possibly testing a few more trial divisions, doing the same with q_A and q_B, computing N and finally testing if N is actually the product of two large prime integers.

It would be interesting to generate an RSA modulus N such that the prime factors p and q are *strong* primes. We do not know how to achieve this but we can test if $(p-1)/2$ is divisible by small prime numbers or not. To do this, we just use the protocol designed for the trial division test, with the function $f(x, y) = 0$ if $(x + y - 1)/2 = 0 \bmod p$ and $f(x, y) = 1$ otherwise.

3.4 Generation of the Shared Private Keys

When N has been generated and tested, the last step is the choice of a public exponent e and of a secret one d. More precisely, we want Alice to know d_A and Bob d_B such that $e \times (d_A + d_B) = 1 \bmod \phi(N)$.

Let ϕ_A be $N - p_A - q_A + 1$ and ϕ_B be $p_B + q_B$. Let M'' be the smallest product of the first prime integers greater than $2e \times 2^n$.

(1) Alice randomly chooses $\zeta_A \in \mathbb{Z}_e$,
(2) Alice and Bob privately compute $(\phi_A + \phi_B)^{-1} - \zeta_A \bmod e$ and only Bob obtains the result ζ_B, as in the protocol of section 3.3,
(3) Alice randomly chooses $T_A \in \mathbb{Z}_{M''}$,
(4) Alice and Bob privately compute $(\phi_A + \phi_B) \times (\zeta_A + \zeta_B) + 1 - T_A \bmod M''$ and only Bob obtains the result T_B,
(5) Alice computes its secret share $d_A = \lfloor T_A/e \rfloor$,
(6) Bob computes its secret share $d_B = \lceil T_B/e \rceil$,
(7) Alice and Bob verify that $e(d_A + d_B) = 1 \bmod \phi(N)$.

In order to verify if d_A and d_B has been correctly computed, Alice chooses a random message m and sends $c = m^{e \times d_A} \bmod N$ to Bob who replies the original message m to prove that he knows d_B. Then Bob verifies in the same way that Alice owns a correct exponent d_A.

This protocol is based on the algorithm Boneh and Franklin used to compute $e^{-1} \bmod \phi(N)$. They have noticed that this computation can be done without reduction modulo $\phi(N)$ but just with reductions modulo e. Their algorithm is the following: first compute $\zeta = -\phi(N)^{-1} \bmod e$ and then take $T = \zeta\phi(N) + 1$. Since e divides T, $d = T/e$ verifies $ed = 1 \bmod \phi(N)$.

4 Comparison with General 2-Party Computation Schemes

We said in the introduction that, from a theoretical point of view, there already exist general protocols that enable to privately evaluate expressions like $N = (p_A + p_B) \times (q_A + q_B)$ in polynomial time [26,16,18,13]. All those schemes transform 2-party computations into secure evaluation of logical circuits. This enables to reduce any computation to the combination of a very small set of elementary protocols, like the computation of the logical AND of two bits, at the cost of polynomial but unpractical solutions.

If we focus on the multiplication $N = (p_A + p_B) \times (q_A + q_B)$, with p_A, p_B, q_A and q_B, four $(n/2 - 1)$-bit integers, the most practical logical circuit able to evaluate N needs $O(n^2)$ gates and is depth is $O(n)$. Using the results of [13], we obtain a protocol that enable to privately compute N with a communication complexity of $O(n^2)$ and with at least $O(n)$ rounds of communication (for a fixed value of the security parameter).

In order to compare this complexity with our scheme's, we need to choose an ANDOS protocol. The one described in [25] has a communication complexity $C(t) = 2^{\alpha(\sqrt{\log t})}$ when one secret out of t is sold and needs a constant number of rounds of communication ($\alpha \approx 1.1$). The global communication complexity of our scheme is $2\sum_{i=1}^{k} C(p_i^2)$ with $2^n \approx \prod_{i=1}^{k} p_i$ and p_i the i^{th} prime number. Consequently, this complexity is about $2\int_2^n 2^{\alpha\sqrt{\log t}}/\log t dt$ (see for example [22]) and this expression is about $2n \times 2^{\alpha\sqrt{\log n}}/\log n = o(n^\beta)\ \forall\beta > 1$. So, asymptotically, our solution is about $O(n)$ times more efficient than general ones in term of communication complexity. Furthermore, the k ANDOS can be parallelized so the resulting protocol as a constant number of rounds of communication while general solutions need at least $O(n)$ rounds.

¿From a more practical point of view, using the results of [25], we estimate the communication to 2MB when $n = 768$ bits. A general solution would clearly be much less efficient since it would need at least $(n/2)^2 \approx 150.000$ Rabin oblivious transfer [21] and a few hundred rounds of communication.

In conclusion, our scheme is much more practical than those derived from general solutions while it is still based on very general security assumptions. But the secure computation of a shared RSA keys always seems to need efficient computers linked by high rate networks. We propose in appendix an alternative solution, less general since it is based on a specific number theoretical problem but that enables very efficient computations and transmissions.

References

1. M. Bellare and S. Micali. Non-Interactive Oblivious Transfer and Application. In *Crypto '89*, LNCS 435, pages 547–557. Springer-Verlag, 1990.
2. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *Proc. of the 20th STOC*, pages 1–10. ACM Press, 1988.
3. C.H. Bennett, G. Brassard, C. Crépeau, and M.-H. Skubiszewska. Practical Quantum Oblivious Transfer. In *Crypto '91*, LNCS 576, pages 351–366. Springer-Verlag, 1992.
4. S. Blackburn, S. Blake-Wilson, M. Burmester, and S. Galbraith. Shared Generation of Shared RSA Keys. Technical Report CORR 98-19, University of Waterloo, 1998. Available at `http://www.cacr.math.uwaterloo.ca`.
5. D. Boneh and M. Franklin. Efficient Generation of Shared RSA Keys. In *Crypto '97*, LNCS 1294, pages 425–439. Springer-Verlag, 1997.
6. G. Brassard, C. Crépeau, and J-M. Robert. Information Theoretic Reductions among Disclosure Problems. In *Proc. of the 27th FOCS*, pages 168–173. IEEE, 1986.
7. G. Brassard, C. Crépeau, and J-M. Robert. All-or Nothing Disclosure of Secrets. In *Crypto '86*, LNCS 263, pages 234–238. Springer-Verlag, 1987.

8. D. Chaum, C. Crépeau, and I. Damgård. Multiparty Unconditionally Secure Protocols. In *Proc. of the 20th STOC*, pages 11–19. ACM Press, 1988.
9. B. Chor and E. Kushilevitz. A Zero-One Law for Boolean Privacy. In *Proc. of the 21st STOC*, pages 62–72. ACM Press, 1989.
10. C. Cocks. Split Knowledge Generation of RSA Parameters. In *Cryptography and Coding: Proceedings of 6th IMA Conference*, LNCS 1355, pages 89–95. Springer-Verlag, 1997.
11. C. Cocks. Split Generation of RSA Parameters with Multiple Participants. Technical report, 1998. Available at `http://www.cesg.gov.uk`.
12. C. Crépeau and J. Kilian. Achieving Oblivious Transfer Using Weakened Security Assumptions. In *Proc. of the 29th FOCS*, pages 42–52. IEEE, 1988.
13. C. Crépeau, J. van de Graaf, and A. Tapp. Commited Oblivious Transfer and Private Multy-Party Computation. In *Crypto '95*, LNCS 963, pages 110–123. Springer-Verlag, 1995.
14. S. Even, O. Goldreich, and A. Lempel. A Randomized Protocol for Signing Contracts. *Communications of the ACM*, 28:637–647, 1985.
15. Y. Frankel, P. MacKenzie, and M. Yung. Robust Efficient Distributed RSA-Key Generation. In *Proc. of the 30th STOC*. ACM Press, 1998.
16. O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game. In *Proc. of the 19th STOC*, pages 218–229. ACM Press, 1987.
17. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM journal of computing*, 18(1):186–208, february 1989.
18. J. Kilian. Founding Cryptography on Oblivious Transfer. In *Proc. of the 20th STOC*, pages 20–31. ACM Press, 1988.
19. D. Naccache and J. Stern. A New Public Key Cryptosystem Based on Higher Residues. In *Proc. of the 5th CCCS*. ACM press, 1998.
20. D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In *Eurocrypt '96*, LNCS 1070, pages 387–398. Springer-Verlag, 1996.
21. M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.
22. J.N. Rosser and L. Schoenfeld. Approximate Formulas for some Functions of Prime Numbers. *Illinois Journal of Mathematics*, 6(1):64–94, march 1962.
23. K. Sakurai and H. Shizuya. A Structural Comparison of the Computational Difficulty of Breaking Discrete Log Cryptosystems. *Journal of Cryptology*, 11(1):29–43, 1998.
24. S. Spalding and R. Wright. Experimental Performance of Shared RSA Modulus Generation. In *proc. of Algorithms and Experiments 98*, pages 34–43, 1998.
25. J.P. Stern. A New and Efficient All-Or-Nothing Disclosure of Secrets Protocol. In *Asiacrypt '98*, LNCS. Springer-Verlag, 1998.
26. A. C. Yao. How to Generate and Exchange Secrets. In *Proc. of the 27th FOCS*, pages 162–167. IEEE, 1986.

A An Efficient Solution Based on Higher Residues Cryptosystem

Using a specific number theoretical problem, we can also propose a much more simple and efficient solution that does not need to perform many rounds of communication. It is based on a trapdoor version of the discrete logarithm problem. More precisely, Alice chooses parameters for the Naccache Stern cryptosystem [19] based on higher residues, i.e σ a squarefree odd B-smooth integer greater than 2^n, where B is a small integer, an RSA modulus N_A such that σ divides $\phi(N_A)$, g an element whose multiplicative order modulo N_A is a large multiple of σ.

The computation of $N = (p_A + p_B) \times (q_A + q_B)$ can be easily done with the following protocol that, on secret inputs x_A and x_B of Alice and Bob make them obtain y_A and y_B such that $y_A + y_B = x_A \times x_B \bmod \sigma$:

- Alice chooses a random x, computes $x^\sigma g^{x_A} \bmod N_A = c$ and sends it to Bob,
- Bob chooses $y_B \bmod \sigma$ and x', computes $c^{x_B} {x'}^\sigma g^{-y_B} \bmod N_A = d$ and sends d to Alice,
- Alice decrypts d and obtains $y_A = x_A \times x_B - y_B \bmod \sigma$.

The security analysis of this protocol is out of the scope of this appendix. We can just notice that a commitment of p_A, p_B, q_A and q_B and a verification of the correctness of the result have to be added (as in [4]). This can be done using modular exponentiation and its homomorphic property.