Practical Threshold RSA Signatures without a Trusted Dealer

Ivan Damgård and Maciej Koprowski

BRICS*, Aarhus University

Abstract. We propose a threshold RSA scheme which is as efficient as the fastest previous threshold RSA scheme (by Shoup), but where two assumptions needed in Shoup's and in previous schemes can be dropped, namely that the modulus must be a product of safe primes and that a trusted dealer generates the keys. The robustness (but not the unforgeability) of our scheme depends on a new intractability assumption, in addition to security of the underlying standard RSA scheme.

1 Introduction

In a threshold public-key system we have a standard public key (for the RSA system, for instance), while the private key is shared among a set of servers, in such a way that by collaborating, these servers can apply the private key operation to a given input, to decrypt it or sign it, as the case may be. If there are l servers, such schemes typically ensure that even if an active adversary corrupts less than $l/2$ servers, he will not learn additional information about the private key, and will be unable to force the network to compute incorrect results. Thus threshold cryptography is an important concept because it can improve substantially the reliability and security of applications in practice of public-key systems.

Threshold schemes based on the discrete log problem are relatively straightforward to build, and have been known for a long time. It is even possible to make efficient schemes where also the key generation phase is done by the servers in a distributed way [13,9]. This way we can completely avoid assuming any fixed trusted parties in the system.

Basing threshold schemes on RSA is technically more difficult because we have to work in a group of non-prime and unknown order (Z_n^* rather than a prime order subgroup of Z_p^* for a prime p). Nevertheless RSA-based schemes have been known for some time, see [10,14] for the first reasonably efficient and robust solutions. However, due to the technical difficulties mentioned, they tend to be more complex and less efficient in comparison to the discrete log schemes. One concrete reason is that they use secret sharing "in two levels", i.e. server i knows a number d_i, such that $\sum_i d_i = d$, the secret RSA exponent. In addition, each d_i is a verifiable secret shared among the servers. In such a scenario, testing

* Basic Research in Computer Science, Centre of the Danish National Research Foundation.

if servers have behaved correctly is more complex than in the discrete log case, and if faults do occur, interaction between the servers is necessary to recover.

Recently, however, Shoup[16] proposed a threshold RSA signature scheme which is essentially as efficient as possible: the scheme uses only one level of secret sharing, to sign a message, each server simply sends a single response to a signature request, and must do work that is equivalent up to a constant factor to computing a single RSA signature. No further interaction is needed to recover from faults. Unfortunately, that scheme - like any previous efficient RSA-based scheme - needs to assume a trusted dealer to generate keys. This is caused by the fact that it relies on a special property for the RSA modulus, namely it must be the product of two so called *safe primes* (i.e. the modulus n is the product of primes p, q, where $p' = (p-1)/2, q' = (q-1)/2$ are also prime). The problem now is that although reasonably efficient distributed RSA key generation protocols are known[1,6], none of these protocols can ensure that the modulus is a product of safe primes[1]. One attempt to overcome this was made by Miyazaki et al. [11], who build a threshold RSA scheme that can use the key generation protocol from [6]. Unfortunately that scheme is significantly less efficient than Shoup's. It uses two-level secret sharing and needs interaction between servers for each message signed, even if no faults occur. Fouque and Stern [8] present independently of our work a distributed threshold RSA scheme in which they modify the distributed generation of RSA keys from [1],[6] and combine this with Shoup's scheme [16]. The security of their protocol is based only on the underlying standard RSA scheme, but is less efficient than ours by a factor of $\Omega(k^2)$, where k is the security parameter (Fouque and Stern estimate a factor of 30 for a realistic setting of the parameters).

In this paper, we overcome the problem in a more efficient way by constructing a new threshold RSA scheme which may be seen as a generalization of Shoup's, is essentially as efficient as that scheme, follows the same communication pattern, but does not need the assumption about safe primes. As we shall see, this implies that the distributed RSA key generation protocol from [6] can be used to generate keys for our scheme. Note that there may be good reasons to avoid safe primes, other than the distributed key generation issue: first, we do not even know if there are infinitely many safe primes, and second it may turn out to be the case that safe primes are not "safe" at all: although it currently looks as if safe prime products are as hard to factor as RSA moduli in general, this may eventually turn out to be false, indeed most experts agree that choosing the primes as randomly as possibly gives the best security.

On the technical level, one difficulty that arises when safe primes are not assumed, relates to the efficient zero-knowledge protocols used in [16] to verify the behavior of servers. These protocols seem to fail if safe primes are not used, primarily because the group we are working in is no longer cyclic, and may have small prime factors in its order. We get around this by showing that with

[1] Of course, generic multiparty computation methods could be used to generate and share such keys in a distributed fashion, but this would be extremely inefficient and completely unsatisfactory in practice

small modifications to the protocols and under an appropriate intractability assumption, the adversary will not be able to exploit the "deficiencies" of the group. Concretely, we show that zero-knowledge proofs of equality of discrete logarithms over a general RSA modulus can be done very efficiently (i.e. without resorting to binary challenge proofs) as long as the prover does not know the factorization. This may be of independent interest, and was previously only known if the modulus was a safe prime product.

Following Shoup, we describe and prove our scheme assuming the random oracle model, however, we rely on it only for robustness of the scheme (and not for unforgeability). At the expense of an additional round of interaction when signing a message, we can avoid using random oracles. The details of this are are omitted since they are standard and straightforward.

To prove the security of our scheme, we need an intractability assumption in addition to the standard RSA assumption. Informally speaking, we assume that given the public key n, e:

– The adversary cannot compute an element $a \neq 1, -1 \bmod n$ such that a has "extremely small order". More precisely the adversary cannot compute an $a \neq 1, -1 \bmod n$ whose order is not divisible by q, where q is the largest prime factor in $\phi(n)$.
– The adversary cannot distinguish a random square modulo n from a random square of maximal order.

As evidence in favor of this assumption, we first note that it is well known that computing the order of a random element is equivalent to factoring. Specifically w.r.t. the first item, for a random RSA modulus n, there is overwhelming heuristic evidence that the prime q will be large (superpolynomial) with overwhelming probability. And so a suitable a cannot be found by choosing randomly. Indeed, it seems that one would need to raise a randomly chosen element to the q'th power to find such an a, however, guessing q is very unlikely to be feasible if factoring is difficult at all. Also, we note that if $n = pq$ is chosen such that $(p-1)/2$ and $(q-1)/2$ have no prime factors less than some number B, then finding $a \neq 1, -1$ of order less than B is as hard as factoring n, since the only possibilities for a are the two non-trivial square roots of 1. In [8], Fouque and Stern show a distributed protocol for generating such RSA moduli efficiently when B is (essentially) a constant (so this does not quite suffice to show that our assumption is equivalent to factoring for such n). The second item can be seen as a generalization of the Quadratic Residuosity Assumption, which can be interpreted as stating that it is difficult to decide if a given element has a maximal power of 2 dividing its order. Our conjecture makes a similar statement for other prime factors.

For the version of our scheme we describe here, we actually need that this assumption holds, even if the adversary is given an oracle for RSA signatures, i.e. , the adversary can specify an M and will be given the e'th root modulo n of $\tilde{H}(M)$, where $\tilde{H}$ is a secure hash function. While this extra condition does not seem to help the adversary in computing orders of elements, it can be removed completely if we are willing to assume that $\tilde{H}$ can be modelled as a random oracle.

In [4] and [7], Damgård/Jurik and Fouque et al. construct threshold versions for (generalizations of) Paillier's probabilistic public key system [12] using the basic techniques from Shoups scheme. These protocols all assume a trusted dealer. Using similar constructions, but starting from our scheme instead of Shoup's, threshold versions of Paillier's scheme without a trusted dealer are easily obtained.

2 Model

Here we describe the model for threshold signature schemes we use, rather informally, due to space limitations. In the type of schemes we consider there are l *servers*. In the *generation phase* on input a security parameter k the public key pk and *secret key shares* $s_1, ..., s_l$ are created, where s_i belongs to server number i. There is a *signing protocol* defined for the servers which takes a message M as input and outputs (publically) a signature σ. Finally, there is a *verification predicate* V, which is efficiently computable, takes pk, message M and signature σ as inputs, and returns *accept* or *reject*. Both the signing protocol and the verification predicate may make use of a random oracle.

To define security, we assume a polynomially bounded static and active adversary $\mathcal{A}$, who corrupts initially $t < l/2$ of the l servers. Thus, the adversary always learns pk and the s_i's of corrupted servers. As the adversary's algorithm is executed, he may issue two types of requests:

– An *oracle request*, where he queries the random oracle used, he is then given the oracle's answer to the query he specified.
– A signature request, where the adversary specifies a message M. This causes the signing protocol to be executed on input M, where the adversary controls the behaviour of corrupted servers (and will of course see whatever information is made public by honest servers).

At the end, $\mathcal{A}$ outputs a message M_0 and a signature σ_0.

We say that $\mathcal{A}$ *wins*, if any of the signing requests resulted in an invalid signature being output, or if he produced a forged signature on a new message, i.e. M_0 was not used in a previous signature request, and $V(pk, M_0, \sigma_0) = accept$.

We say that the scheme is *secure*, if every adversary wins with probability negligible in k[2].

3 The Honest Dealer Scheme

In this section we first describe our scheme assuming an honest dealer that will generate and distribute the keys. The algorithm we specify for the dealer looks

[2] unlike the definition in [10], we treat robustness and unforgeability together - this does not make any essential difference.

rather strange, taken by itself. However, the dealer is designed in such a way that the information he distributes matches the output that can be generated by the distributed RSA key generation protocol of Frankel et al. [6]. Therefore, once we prove the security of the honest dealer scheme, a secure (and efficient) scheme without an honest dealer follows easily. We return to this issue in Section 5 [3].

In the threshold scheme to be described, an RSA public key n, e will be selected. We will assume that there exists some method, represented by a function $\tilde{H}$ for mapping an input message M to an element $\tilde{H}(M) \in Z_n^*$. Then the signature we will compute is just the standard RSA signature $\tilde{H}(M)^d \bmod n$, where d is the private exponent corresponding to e. We will refer to this as *the underlying RSA scheme*. The function $\tilde{H}$ can be a hash function, a redundancy scheme, or a combination of both, our construction will work fine in any case. But we do assume throughout *that the underlying RSA signature scheme is secure* - more precisely that it is not existentially forgeable under a chosen message attack. This is clearly a necessary assumption, since without it, no threshold scheme we build from the underlying scheme can be secure. Note that, assuming $\tilde{H}$ can be modelled as a random oracle, security of RSA signatures using $\tilde{H}$ follow from only the standard RSA assumption.

The dealer. Let $\Delta = l!$. The dealer chooses at random $p_1, \ldots, p_l, q_1, \ldots, q_l \in_R [2^{k-1}, 2^k]$ until $p = (p_1 + \cdots + p_l)$ and $q = (q_1 + \cdots + q_l)$ are prime numbers and $gcd((p-1)/2, \Delta) = gcd((q-1)/2, \Delta) = 1$. The RSA modulus is $n = pq$. The dealer also chooses a public exponent e as a prime $e > l$. The public key is $pk = (n, e)$.

Next the dealer executes generation of private keys from [6] to compute $d\Delta^2 = d_1 + \cdots + d_{t+1} \in \mathbb{Z}$ such that $de \equiv 1 \bmod \Phi(n)$, $\Delta | d_1, \ldots, \Delta | d_{t+1}$ and $|d_1| < Cl^{l+1}\Delta^{11}n^2, \ldots, |d_{t+1}| < Cl^{l+1}\Delta^{11}n^2$ for some constant $C > 1$. [4]

The dealer performs secret sharing over the integers, which was introduced in [5] and presented in a modified version in [6]. For $1 \leq i \leq t+1$ a random polynomial $f_i(x) = \sum_{j=0}^{t} f_{i,j}x^j$ is chosen such that $f_{i,0} = d_i$ and for $1 \leq j \leq t$ we have $f_{i,j} \in_R \{0, \Delta, 2\Delta, \ldots, \Delta^{10}n^2 \cdot \Delta\}$. We define a polynomial $f(x) = f_1(x) + \cdots + f_{t+1}(x)$. We can observe that $f(0) = d\Delta^2$ and $f(x)$ is a multiple of Δ for all integers x.

For $1 \leq i \leq l$, the dealer computes $s_i = f(i) = f_1(i) + \cdots + f_{t+1}(i)$, which is a secret key of server i. If we define $\alpha(k, l) = 4k + (12l + 4)\log l$, it is easy to verify that $0 \leq s_i < 2^{\alpha(k,l)}$.

The dealer chooses $v \in Z_n^*$ as a random square. For $1 \leq i \leq l$, the dealer computes verification key $v_i = v^{s_i \Delta^2}$ of server i.

[3] We note that the description in [6] in some places leaves open alternatives for how details in their key generation protocol are executed. Choosing different options lead to minor differences in the output distribution. We stick to one option here for simplicity. Any of the other options could easily be accommodated here by adjusting the description of the honest dealer.

[4] In case of a small public exponent, the protocol from [6] instead generates the private exponent $d\Delta^2$ as a sum of l shares. Our construction could also be based on this method. The protocol and the proofs would be analogous.

Signing protocol.

1. When a message M is requested to be signed, we set $x = \tilde{H}(M)$, where $\tilde{H}$ is a hash function, a redundancy scheme, or a combination of both, and use the scheme to compute $x^d \bmod n$.
2. We define the signature share x_i of server i by
$$x_i = x^{2\Delta^2 s_i}.$$
3. The server i can prove that the discrete logarithm of x_i^2 to base $\tilde{x} = x^{4\Delta^2}$ is the same as the discrete logarithm of v_i to the base v^{Δ^2}.
We construct the proof of correctness. Let H be a hash function modelled as a random oracle, whose output is an L_1-bit integer, where L_1 is a secondary security parameter (e.g. $L_1 = 128$).
Each server i chooses at random a number $r \in \{0, \ldots, 2^{\alpha(k,l)+2L_1} - 1\}$. Let
$$c = H(v, \tilde{x}, v_i, x_i^2, v^{r\Delta^2}, x^{4r\Delta^2}), z = s_i c + r.$$
The proof of correctness produced by server i is (z, c).
4. To verify this proof of correctness, one should check that
$$c = H(v, \tilde{x}, v_i, x_i^2, v^{z\Delta^2} v_i^{-c}, \tilde{x}^z x_i^{-2c}).$$
5. Suppose that valid shares were generated by honest servers from a set $S = \{i_1, \ldots, i_{t+1}\} \subset \{1, \ldots, l\}$.
For all $j \in S$ we define the Lagrange coefficients multiplied by Δ:
$$\lambda_{0,j}^S = \Delta \cdot \prod_{i \in S \setminus \{j\}} \frac{i}{i-j}.$$
Clearly the coefficients $\lambda_{0,j}^S$ are integers and
$$d\Delta^3 = f(0)\Delta = \sum_{j \in S} \lambda_{0,j}^S f(j) = \sum_{j \in S} \lambda_{0,j}^S s_j.$$
Therefore to combine shares, we can compute
$$\omega = \prod_{j \in S} x_j^{2\lambda_{0,j}^S} = x^{4\Delta^2 \sum_{j \in S} \left(s_j \lambda_{0,j}^S\right)} = x^{4\Delta^5 d}.$$
6. We can observe that
$$\omega^e = x^{4\Delta^5}.$$
Since e is prime to $4\Delta^5$, we can obtain such integers a and b from the extended Euclidian algorithm that $a4\Delta^5 + be = 1$. Finally we have a signature $y = w^a x^b$, because
$$y^e = \left(\omega^a x^b\right)^e = x.$$

This shows that y is obtained if the signature shares x_i are computed by honest servers only. In real life, we will only know that the x_i's are values that allow the servers to produce acceptable proofs of correctness. We will later show that this (with overwhelming probability) is sufficient.

4 Proof of Security for the Honest Dealer Scheme

We start by stating our intractability assumption more formally. To this end, we define a *signing oracle* $O(n, e, \tilde{H})$ to be an oracle that on input a message M returns the signature $\tilde{H}(M)^d \bmod n$.

Conjecture 1. – Consider any probabilistic polynomial time algorithm who gets as input n, e (as chosen by the honest dealer on input k), gets access to a signing oracle $O(n, e, \tilde{H})$, and outputs a number a. For any such algorithm, the probability that $a \neq 1, -1 \bmod n$ and q does not divide the order of a, where q is the largest prime factor in $\phi(n)$, is negligible in k.

– Let $D = \{D(k) | \; k = 1, 2..\}$ be the family of distributions where $D(k)$ is the distribution of n, e, v generated by our honest dealer on input k. Define D' to be the same, except that v is chosen as a random square of maximal order. Then D and D' are polynomial time indistinguishable, where distinguishers are given access to a signing oracle $O(n, e, \tilde{H})$.

This assumption was already discussed in the introduction. Note that if we are willing to assume that $\tilde{H}$ can be modelled as a random oracle, then the signing oracles can be removed from the conjecture by a standard argument[5].

A number of preliminary observations:

Lemma 1. *The proofs of correctness for signature shares produced by honest servers can be simulated with a statistically close distribution, given the public key and the message to be signed.*

Proof. We construct a simulator which can simulate the proof of correctness generated by server i without knowing the value of secret s_i. Recall that we invoke the random oracle for the hash function H. The simulator controls the random oracle. Whenever the adversary queries the random oracle, if it has not been defined yet at the given point, the simulator picks a random value and sends it to the adversary. When a honest server is expected to produce a proof of correctness for given x, x_i, the simulator picks random $c' \in \{0, \ldots, 2^{L_1} - 1\}$ and $z' \in \{0, \ldots, 2^{\alpha(k,l)+2L_1} - 1\}$. We declare the value of the random oracle at the point $(v, \tilde{x}, v_i, x_i^2, v^{z'\Delta^2} v_i^{-c'}, \tilde{x}^z x_i^{-2c'})$ to be c'. With overwhelming probability, the random oracle has not been defined at this point before. The simulated proof is (z', c'). The only difference to a real proof (z, c) is that in a real execution, we have $z = r + cs_i$, where r is a random $\alpha(k, l) + 2L_1$-bit number. But since r and z' are L_1 bits longer than cs_i, the distance between the distributions of z and z' is exponentially small in L_1. □

Lemma 2. *Let q be the largest prime factor in $\phi(n)$, and consider a signature share x_i (for an input x) produced by a corrupt server. Assume that the element v produced by the honest dealer has maximal order (among all squares modulo n),*

[5] Since under this assumption, a signing oracle is easy to implement: if the adversary wants to see a signature on message M, choose a random $\sigma \in Z_n^*$, define the output of $\tilde{H}$ on input M to be $\sigma^e \bmod n$, so that σ now is the signature on M

and that x_i is incorrect, i.e., $x_i^2 \neq (x^{4\Delta^2})^{s_i} \bmod n$. Then, either q does not divide the order of $x_i^2 \cdot (x^{4\Delta^2})^{-s_i} \bmod n$, or the probability that the adversary can construct an acceptable proof of correctness for x_i is negligible. Furthermore, a correct signature can be computed from $t+1$ correct signature shares.

Proof. Let (z, c) be an acceptable proof produced by a corrupt server i. Therefore

$$c = H(v, \tilde{x}, v_i, x_i^2, v^{z\Delta^2} v_i^{-c}, \tilde{x}^z x_i^{-2c}).$$

We can reinterpret this proof as an application of the following interactive protocol, where the verifier is replaced by a call to the random oracle:

Let G be a group of squares in $\mathbb{Z}_n^*$. We have elements $\tilde{v}, w = v^s \in G$, where $\tilde{v}$ has maximal order in G and the prover knows s. The prover P makes elements α, β, guaranteed to be in G as well, and wants to convince us that $\alpha^s = \beta$.

So $\alpha, \beta, \tilde{v}$ correspond to $\tilde{x}, x_i^2, v^{\Delta^2}$ above. Note that if v has maximal order, so does v^{Δ^2}, since n was chosen such that G has order prime to Δ.

The prover performs the following steps:

1. P chooses r in some large enough interval and sends $a = \tilde{v}^r, b = \alpha^r$.
2. P gets a random challenge c from the verifier.
3. P replies by sending $z = r + cs$
4. To check the proof, one verifies that $\tilde{v}^z = aw^c$ and $\alpha^z = b\beta^c$.

We can always write $G = G_1 \times .. \times G_u$, where the order of G_j is a power of q_j and $q_1, \dots, q_u$ are the distinct prime factors in the order of G. So then we can think of α as a u-tuple, $\alpha = (\alpha_1, ..., \alpha_u)$, $\alpha_j \in G_j$, and similarly for the other group elements. Now, of course, $\alpha^s = \beta$ iff $\alpha_j^s = \beta_j$ for all j.

Claim. If for some j, $\alpha_j^s \neq \beta_j$, then for any initial message (a, b) in the protocol, there is at most one value of $c \bmod q_j$, for which a satisfactory reply z to c exists.

To prove this, there are two cases we must look at, depending on whether $\beta_j \in <\alpha_j>$ or $\beta_j \notin <\alpha_j>$.

Assume first that $\beta_j \notin <\alpha_j>$. Suppose that for some initial message a, the prover can answer both c and c', where $c \neq c' \bmod q_j$. This means that the prover can send z and z' such that $\alpha_j^z = b\beta_j^c$ and $\alpha_j^{z'} = b\beta_j^{c'}$. Dividing one equation by the other we get $\alpha_j^{z'-z} = \beta_j^{c'-c}$. Since we assumed that $\beta_j \notin <\alpha_j>$, it must be the case that $<\beta_j> \cap <\alpha_j>$ is a proper subgroup of $<\beta_j>$. Hence the order of $\beta_j^{c'-c}$ must be strictly smaller than the order of β_j, but this is a contradiction since $c - c'$ is relatively prime to q_j by assumption.

Next, assume that $\beta_j = \alpha_j^{\tilde{s}}$ for some $\tilde{s}$, but nevertheless $\beta_j \neq \alpha_j^s$. So $\tilde{s} \neq s \bmod ord(\alpha_j)$, where $ord(\alpha_j)$ is some power of q_j. If we let $q_j^{\nu_j}$ be the order of $\tilde{v}_j$, we have $ord(\alpha_j) \leq q_j^{\nu_j}$ because $\tilde{v}$ has maximal order in G. Assume again that given some initial message a, the prover can answer both c and c', where $c \neq c' \bmod q_j$, by sending responses z, z'. From the equations the verifier checks, we get

$$\tilde{v}_j^{z'-z} = w_j^{c'-c}, \; \alpha_j^{z'-z} = \beta_j^{c'-c}$$

Now, $c' - c$ is relatively prime to q_j, we can set $d = (c' - c)^{-1} \bmod q_j^{\nu_j}$ and raise both equations to the d'th power. Since the order of α_j - and hence of β_j - is at most $q_j^{\nu_j}$, this gives us

$$\tilde{v}_j^{d(z'-z)} = w_j, \ \alpha_j^{d(z'-z)} = \beta_j$$

Hence $d(z - z') = s \bmod ord(\tilde{v}_j)$ and also $d(z - z') = \tilde{s} \bmod ord(\alpha_j)$, which implies $s = \tilde{s} \bmod ord(\alpha_j)$, a contradiction.

This finishes the proof of the claim.

We now return to the situation where we have given an incorrect signature share x_i. Recall that we defined q to be the largest prime factor in the order of Z_n^*, and hence in the order of G, so q is one of the q_j's, say $q = q_1$. Let ϕ be the natural homomorphism from G to G_1. We may assume that $\phi(x_i^2) \neq \phi((x^{4\Delta^2})^{s_i})$, i.e., x_i is " incorrect in G_1", since otherwise q does not divide the order of $x_i^2(x^{4\Delta^2})^{-s_i}$. It then follows from the claim we just proved that for each oracle call the adversary makes where x_i occurs as signature share, the probability that this results in an acceptable proof is at most $1/q$. (note that if the adversary attempts to make a proof without calling the oracle it is clear that it will be accepted with probability at most 2^{-L_1}). It follows from the first part of conjecture 1 that $1/q$ must be negligible, since otherwise a small order element could be found by guessing at random. Since the adversary can only make a polynomial number of oracle calls, it follows that the probability that he can make an acceptable proof for such an x_i is negligible.

Combining shares. Assume that we have $t+1$ correct signature shares $x'_{i_1}, \ldots, x'_{i_{t+1}}$. For $1 \leq j \leq t+1$ the signature shares satisfy a property

$$x'_{i_j} = \tilde{x}^{s'_{i_j}}, \text{ where } s'_{i_j} \equiv s_{i_j} \bmod \operatorname{ord}(v^{\Delta^2}).$$

Since v is an element of maximal order in the group of squares in $\mathbb{Z}_n^*$ and $\tilde{x} = x^{4\Delta^2}$, we have

$$x'_{i_j} = \tilde{x}^{s'_{i_j}} = \tilde{x}^{s_{i_j}} \bmod n.$$

Therefore $t+1$ correct signature shares allow us to compute a correct signature. □

Lemma 3. *Let n, e, distributed as the honest dealer chooses them, be given. Based on this, the information the adversary learns from the honest dealer initially can be simulated with a statistically close distribution.*

Proof. Suppose that the adversary corrupted t servers $i_1, \ldots, i_t$.

We choose at random $r \in \mathbb{Z}_{ln}$ and distribute $r\Delta^2$ randomly as a sum $r\Delta^2 = r_1 + \cdots + r_{t+1}$, where $\Delta | r_1, \ldots, \Delta | r_{t+1}$ and $|r_1| < Cl^{l+1}\Delta^{11}n^2, \ldots, |r_{t+1}| < Cl^{l+1}\Delta^{11}n^2$. We perform secret sharing over the integers to share r. For $1 \leq i \leq l$ a random polynomial $g_i(x) = \sum_{j=0}^{t} g_{i,j}x^j$ is chosen such that $g_{i,0} = r_i$ and for $1 \leq j \leq t$ we have $g_{i,j} \in_R \{0, \Delta, \ldots, \Delta^{10}n^2 \cdot \Delta\}$. We define a polynomial $g(x) = g_1(x) + \cdots + g_{t+1}(x)$. We can observe that $g(0) = r\Delta^2$.

The function g gives us a polynomial sharing over the integers of r, which was generated like in the sum-to-poly protocol [5] and by Lemma 3 from [5] it is almost t-wise independent. Since the adversary learns t shares $s_{i_1} = (f_1 + \cdots + f_{t+1})(i_1), \ldots, s_{i_t} = (f_1 + \cdots + f_{t+1})(i_t)$, he can not distinguish these shares from random values and from the shares generated for him by the honest dealer.

Let w be a random square in $\mathbb{Z}_n^*$. We define the verification key $v = w^e \bmod n$.

The verification key of a corrupted server i is $v_i = v^{s_i \Delta^2}$. For an uncorrupted server i, we define set $S = \{0, i_1, \ldots, i_t\}$. We can take the normal Lagrange coefficients and multiply them by Δ so they become integers. The results are called $\lambda_{i,j}^S$ and we have

$$\Delta f(i) = \sum_{j \in S} \lambda_{i,j}^S f(j).$$

Since the adversary can not distinguish our secret d from r, we can compute

$$\begin{aligned} v_i &= v^{s_i \Delta^2} = v^{\Delta(d\Delta^2 \lambda_{i,0}^S + \lambda_{i,i_1}^S s_{i_1} + \cdots + \lambda_{i,i_t}^S s_{i_t})} \\ &= w^{\Delta(\Delta^2 \lambda_{i,0}^S + e(\lambda_{i,i_1}^S s_{i_1} + \cdots + \lambda_{i,i_t}^S s_{i_t}))} \bmod n. \end{aligned}$$

The adversary's view consists from $n, e, s_{i_1}, \ldots, s_{i_t}, v, v_1, \ldots, v_l$. Since it was generated on the basis of the adversary's shares $s_{i_1}, \ldots, s_{i_t}$, which were statistically indistinguishable from the adversary's shares produced by the honest dealer, the adversary can not distinguish this view from the one given by the honest dealer. □

Lemma 4. *Assume we are given a set of values distributed by the honest dealer to the adversary, i.e., $n, e, v, v_1, v_2, ..., v_l$ and the s_i's sent to the corrupt servers. Let also a message M, and the signature $\tilde{H}(M)^d \bmod n$ be given. Based on this, the contributions from honest servers in the protocol where M is signed can be simulated with the correct distribution.*

Proof. Let $\{i_1, \ldots, i_t\}$ be the set of corrupted servers. Let $y \equiv \tilde{H}(M)^d \bmod n$ and $x \equiv y^e \equiv \tilde{H}(M) \bmod n$.

We define set $S = \{0, i_1, \ldots, i_t\}$. We can take the normal Lagrange coefficients and multiply them by Δ so they become integers. We can easily compute $x_i = x^{2\Delta^2 s_i}$ for an uncorrupted player i as

$$x_i = y^{2\Delta(\Delta^2 \lambda_{i,0}^S + e(\lambda_{i,i_1}^S s_{i_1} + \cdots + \lambda_{i,i_t}^S s_{i_t}))} \bmod n.$$

□

4.1 Proof Assuming v Has Maximal Order

As a first step, we prove:

Lemma 5. *Modify the honest dealer scheme described above such that the honest dealer chooses v to be a random element of maximal order (among the squares modulo n). Then the resulting scheme is secure under Conjecture 1 and assuming the underlying standard RSA signature scheme is secure.*

Proof. Assume we are given an adversary $\mathcal{A}$ that breaks the scheme, with probability at least $1/p(k)$, for some polynomial $p()$. We will then build an algorithm that with approximately the same probability either breaks the first part of Conjecture 1 or the underlying RSA scheme. So our algorithm gets n, e as input, and also gets a chosen message attack on the underlying RSA scheme, i.e., access to an oracle which on input M returns $\tilde{H}(M)^d \bmod n$. The algorithm now behaves as follows:

1. Invoke Lemma 3 to generate from n, e a simulation of the honest dealer (note that this produces a random square v which does not necessarily have maximal order - we deal with this problem below). Send the data produced to $\mathcal{A}$.
2. For every oracle request $\mathcal{A}$ issues, check if the input value to the oracle that $\mathcal{A}$ specified has been asked for before. If so, return the same answer that was returned earlier. Otherwise, return a fresh random value as an answer and record this value.
3. For every signature request (say, on message M) $\mathcal{A}$ issues, call the oracle to obtain the signature $\tilde{H}(M)^d \bmod n$. Use this and the data generated in step 1 to invoke Lemma 4 and compute the contributions from honest servers in the signing protocol where M is the input. Invoke Lemma 1 to simulate the proofs of correctness from honest players. Send all data produced in this step to $\mathcal{A}$, and receive signature shares x_i and proofs for the corrupt servers from $\mathcal{A}$.
4. If $\mathcal{A}$ produces an incorrect signature share x_i and an acceptable proof for this share, stop and output $x_i^2 \cdot (x^{4\Delta^2})^{-s_i} \bmod n$ (where $x = \tilde{H}(M)$ and M is the message that was signed).
5. If $\mathcal{A}$ stops and outputs M_0, σ_0, output this pair and stop.

To analyze this algorithm, note first that the simulation of the honest dealer in step 1 produces v as a random square, where the honest dealer we have assumed in this subsection chooses v as a random square of maximal order. However, for any prime p, there is a non-negligible probability that a randomly chosen number modulo p has maximal order, namely $p - 1$ (see [15]). This (and the Chinese Remainder Theorem) implies that if we let GOOD be the event that v is a square of maximal order, there is a non-negligible probability that GOOD occurs. It will therefore be sufficient to show that the probability that our algorithm breaks one of the two assumptions, given that GOOD occurs, is non-negligible.

Under this assumption, step 3 simulates our honest dealer with a statistically close distribution. Therefore, the simulations of the signing protocols are also statistically close to the real life distributions (by Lemma 4 and 1). The simulation of the random oracle is trivially perfectly indistinguishable from the real thing. It follows that the probability that $\mathcal{A}$ breaks the threshold signature scheme during our simulation is equal to the probability with which this happens in real life except for a negligible amount, and certainly is at least $1/p'(k)$ for some polynomial.

However, assume first that $\mathcal{A}$ does this by producing an incorrect signature share x_i and a valid proof for it (by Lemma 2 this is necessary to make the signing protocol output a bad signature). By Lemma 2, this means that $x_i^2 \cdot (x^{4\Delta^2})^{-s_i} \bmod n$ has order not divisible by q, except with negligible probability, and so we have broken the first part of Conjecture 1. On the other hand, if M_0 did not occur in any of $\mathcal{A}$'s signature requests, it did not occur in any of ours either, so if also $\sigma_0 = \tilde{H}(M_0)^d \bmod n$, i.e., is a valid signature, we have broken the underlying RSA signature scheme.

4.2 Proof in General

We are now ready for the main result:

Theorem 1. *Consider the original honest dealer scheme described above where the honest dealer chooses v to be a random square modulo n. This scheme is secure under Conjecture 1 and assuming the underlying standard RSA signature scheme is secure.*

Proof. Assume the result is false, i.e. there exists an adversary $\mathcal{A}$ that breaks the scheme with significant probability. We will then argue that this leads to a contradiction with the second part of Conjecture 1. So let us assume that we are given values n, e, v. We know that n, e are chosen as the honest dealer would choose them, and we will show how to use the assumed adversary $\mathcal{A}$ to decide if v is a random square or a square of maximal order.

Note first that we may as well try to decide if $v^e \bmod n$ is random or of maximal order, since raising to the e'th power preserves order and is a 1-1 mapping. So by replacing v by $v^e \bmod n$, we see that we may assume without loss of generality that we know the e'th root of v. With this in mind, a trivial modification of Lemma 3 shows how the honest dealer can be simulated given n, e and v (and the e'th root of v).

We now run the simulation algorithm that appears in the proof of Lemma 5, with two changes:

– In step 1, we run the modified simulation of the honest dealer we just described.
– Having finished, we output v *is random* if $\mathcal{A}$ broke the threshold signature scheme, and v *has maximal order* otherwise.

It is evident from this description that if v has maximal order, we will be producing a simulation that is statistically close to the view of A attacking the scheme with maximal order v, and similarly for random v. It now follows that if v is in fact random, we will output v *is random* with probability at least $1/p(k)$ for some polynomial $p()$, by assumption on $\mathcal{A}$, while this happens with negligible probability if v has maximal order, by Lemma 5.

5 Removing the Honest Dealer

By inspection, it is straightforward to check that the output data from the distributed key generation protocol of [6] matches the data we have assumed that the trusted dealer generates, with one exception : we have required that n is such that $\phi(n)/4$ is not divisible by any prime less than l, and this condition is not automatically satisfied using [6].

This is easily handled, however: the protocol from [6] contains a test division step where each candidate p for a prime factor in n is testdivided by small prime factors. At this point, p is shared additively among the players, so it is trivial to obtain an additive sharing of $p-1$, and testdivide $p-1$ by all primes less than l. This will of course slow down the protocol because more candidates will be rejected, however, by Mertens' theorem the cost will only be a factor proportional to $\log l$.

To show security of the combined scheme, we assume (for concreteness) that the protocol from [6] according to the definition of Canetti [2] is a secure protocol for computing the function F, which on input the security parameter k outputs to all players the values $n, e, v, \{v^{s_i} \bmod n\}$, and s_i as private output to server i[6]. Security of the entire combined scheme now follows from Canetti's composition theorem, provided we show that our protocol is secure given an "ideal implementation" of F, i.e., an oracle that on input k outputs to all players a set of output values for F chosen according to the correct distribution. But since such an oracle is equivalent to an honest dealer, the required proof is precisely what we have given in the previous sections [7].

6 Efficiency Analysis

It is straightforward to check that the number of bits sent by each server in order to sign a message, as well as the number of modular multiplications the server needs to perform, is proportional to the bit length of its share s_i of the secret key. From the estimates on s_i in Section 3 it therefore follows that the communication complexity per server is $O(k + l \log l)$ bits and the computation is $O(k + l \log l)$ modular multiplications, where l is the number of servers and k is the length of the modulus.

This is more than in Shoups[16] scheme which has complexity $O(k)$, however, in practice k must be 1000 or more for security reasons, while l is going to be much smaller, so this difference is hardly significant in practice. In the hidden constants, the main difference is a factor of 2 in Shoup's favor. As a concrete example, for a 1000 bit modulus and 32 servers, Shoups scheme will have shares of size 1 Kbit while our shares will be about 4 Kbits.

[6] [6] does not directly reference the definition of [2]. Nevertheless, the simulation based security proof they give fits with Canetti's definition

[7] Note that we allow the adversary to do a chosen message attack after seeing the public key. Strictly speaking, the model from [2] does not permit this because contributions from corrupted players must be chosen initially when the adversary is static. However, recent work by Canetti [3] does allow taking this into account

References

1. D. Boneh and M. Franklin *Efficient generation of shared RSA keys*, Proc. of Crypto' 97, Springer-Verlag LNCS series, nr. 1233.
2. R. Canetti, *Security and Composition of Multiparty Cryptographic Protocols*, Journal of Cryptology, vol.13, 2000. On-line version at http://philby.ucsd.edu/cryptolib/1998/98-18.html.
3. R. Canetti, *A unified framework for analyzing security of protocols* , Cryptology Eprint archive 2000/67, http://eprint.iacr.org/2000/067.ps
4. Damgård and Jurik: *A Generalization and some Applications of Paillier's Probabilistic Public-key System*, to appear in Public Key Cryptography 2001.
5. Yair Frankel, Peter Gemmell, Philip D. MacKenzie and Moti Yung *Optimal-Resilience Proactive Public-Key Cryptosystems* Proc. of FOCS 97.
6. Yair Frankel, Philip D. MacKenzie and Moti Yung *Robust Efficient Distributed RSA-Key Generation*, Proc. of STOC 98.
7. P. Fouque, G. Poupard, J. Stern: *Sharing Decryption in the Context of Voting or Lotteries*, Proceedings of Financial Crypto 2000.
8. Pierre-Alain Fouque and Jacques Stern: *Fully Distributed Threshold RSA under Standard Assumptions*, IACR Cryptology ePrint Archive: Report 2001/008, February 2001
9. Gennaro, Jarecki, Krawczyk and Rabin: *Secure Distributed Key Generation for Discrete-Log Based Cryptosystems*, Proc. of EuroCrypt 99, Springer Verlag LNCS series, nr. 1592.
10. Gennaro, Rabin, Jarecki and Krawczyk: *Robust and Efficient Sharing of RSA Functions*, J.Crypt. vol.13, no.2.
11. Shingo Miyazaki, Kouichi Sakurai and Moti Yung *On Threshold RSA-Signing with no Dealer*, Proc. of ICISC 1999, Springer Verlag LNCS series, nr.1787.
12. P.Pallier: *Public-Key Cryptosystems based on Composite Degree Residue Classes*, Proceedings of EuroCrypt 99, Springer Verlag LNCS series, pp. 223-238.
13. Pedersen: *A Threshold cryptosystem without a trusted third party*, proc. of EuroCrypt 91, Springer Verlag LNCS nr. 547.
14. T.Rabin: *A Simplified Approach to Threshold and Proactive RSA*, proc. of Crypto 98, Springer Verlag LNCS 1462.
15. J. B. Rosser and L. Schoenfeld: *Approximate formulas for some functions of prime numbers*, Ill. J. Math. 6 (1962), 64–94.
16. Victor Shoup *Practical Threshold Signatures*, Proceedings of EuroCrypt 2000, Springer Verlag LNCS series nr. 1807.