Partially Interactive Threshold RSA Signatures

H.L. Nguyen

Dept. of Computer Science,
University of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB,
United Kingdom.
hn2503@bristol.ac.uk

Abstract. In this paper, a new threshold signing scheme for RSA is proposed. The scheme does not require a trusted third party and no secure information is leaked throughout the protocol. The time and storage complexity of the protocol is linear in the number of parties and no restriction is placed on the RSA moduli. Combined with the n-out-of-n key generation protocol of Boneh and Franklin, one has a complete solution for the threshold RSA problem with no trusted dealer.

1 Introduction

Threshold decryption has been studied a lot for the last two decades. It is a branch of public key cryptography in general, and multi-party computation in particular. Essentially, in a k-out-of-n threshold crypto-system, denoted (k, n) where $1 < k < n$, for the RSA function, our aim is to split the secret decryption/signing exponent d into n different pieces, which are then distributed privately to n parties. This enables:

- Any k or more out of n total parties, when they come together, they can "reconstruct" the secret d in a way which enables them to decrypt or sign a message. This should be done in a way that does not reveal the value of d and its shares to any one in the scheme.
- Secondly, signing or decryption will be totally impossible in the circumstance where less than k parties are present.

The reason why threshold decryption/signing is very useful in practice is because not only does it provide secrecy and reliability but also flexibility.

One of many typical examples is in digital signature. Assuming there are 10 managers in a bank, if every manager has his or her own copy of the secret key then the system can be easily suffered from single point of failure or misuse due to compromise and machine break down. In contrast, if a valid signature requires the signatures of all ten managers in the company then it will be very secure but might not be convenient to use in real life. Therefore the best solution for this problem might be that as long as a document is digitally signed by any 5 or more out of 10 managers then it will be valid and that is exactly what a (5,10) threshold signing scheme tries to achieve. In addition, if an adversary wants to obtain a signature on a message, he must compromise at least 5 people in the scheme and that is a much harder thing to do compared to a traditional public protocol.

As another example one can consider a PKI implementation where the CA is distributed amongst many sites, so as to avoid a single point of failure. But the system must be robust against occasional system downtime or network problems. Hence, distributing the RSA signing key amongst a number of boxes which implement the CA in a threshold manner one can obtained security against a single point of failure and robustness against errors.

The area of threshold cryptography has been pioneered by Adi Shamir in his 1978 paper [11], however the idea only took off when the problem was formally stated by Desmedt in [5]. Since then there has been much work devoted to the topic such as Desmedt and Frankel [6], Pedersen [9], Gennaro et. al. [8], and many more. However, the majority of these solutions are only for discrete logarithm based system that has a direct application to the Elgamal encryption and decryption algorithm. The reason why discrete logarithm based threshold systems are easier to design is because the group in which one works has a publicly known order. Whereas, in the RSA signature scheme, the group we are working in has an unknown group order and so various technical problems arise. For example, standard polynomial interpolation over the ring $\mathbb{Z}_{\phi(N)}$ is hard as no party knows $\phi(N)$.

Another problem is that it is relatively easy to generate a shared discrete logarithm public/private key pair, but it is harder to generate a shared RSA public/private key pair without the presence of a trusted third party. However, there was in breakthrough in the area of shared RSA key generation when both Boyd [3] and Frankel [7] independently proposed a simple and elegant solution for distributed RSA. The decryption key d is additively shared among n parties, $d = d_1 + d_2 + \cdots + d_n$, signing is simply done as follows:

$$s = m^d = m^{d_1} \cdots m^{d_n} \pmod N,$$

and each $s_i = m^{d_i} \pmod N$ is called the partial signature or signature share.

Extending this idea, Boneh and Franklin [2] built a protocol, which does not require a trusted third party, and which can efficiently generate shared RSA keys that satisfy the above property. The drawback of the scheme is that it only gives us a n-out-of-n threshold decryption that cannot be switched easily into a k-out-of-n threshold scheme.

A number of threshold schemes for RSA have been proposed in the literature, most notable are Rabin's [10] and Shoup's [12] schemes. In Rabin's protocol, the author uses Shamir secret sharing to share the secret but on signing the k signing parties need to interact so as to recover the secrets of the non-signing parties. This removes the problem of working in a group of unknown order, but means the scheme leaks information about the additive shares of various parties. To get around this problem a *share-refreshing* protocol is given. All parts of Rabin's scheme require a large amount of interaction between the various parties.

Taking a different approach, Shoup provides a framework that leads to the possibility of applying the protocol in practice, where dealing, signature share generation, signature share verification and signature share combining are separated from each other and only the first part, i.e. dealing, requires interaction of the various parties. The scheme Shoup proposes is then fully non-interactive, bar the initial dealing phase. However, the drawbacks of his scheme are that it requires both a trusted dealer and strong RSA moduli. Hence, Shoup's scheme cannot be applied with the Boneh and Franklin shared key generation protocol.

In this paper we give a RSA threshold scheme which does not require trusted dealers and which can be applied with the Boneh and Franklin shared key generation protocol. In addition we try to minimise the amount of interaction required between the parties and we eliminate the need for a share refreshing stage of the protocol.

2 Problem Definition and Notation

We assume that a set of n players wishes to generate a number of threshold RSA signatures. Using a scheme such as that in [2] the n players can generate a shared RSA modulus N, a public exponent e and n shares d_i of the secret exponent d, such that

$$d = d_1 + d_2 + \cdots + d_n.$$

This shared RSA key generation protocol can be executed without the need for a trusted dealer.

The parties now wish to use these shares so as to generate a threshold RSA signature scheme, with threshold value k. The resulting signature should be compatible with existing hash-and-sign RSA signatures such as RSA-FDH [1]. This means that we want any k parties to come together so as to be able to sign a document. We let $I = \{t_1, \ldots, t_k\} \subset \{1, \ldots, n\}$ denote the set of parties who wish to come together to sign the document and $I' = \{1, \ldots, n\} \setminus I = \{t_{k+1}, \ldots, t_n\}$ denote the other parties. There are essentially two existing ways of doing this, both with disadvantages.

In [10] Rabin gives a scheme which does not require a trusted dealer, as we require, but which works by the k signing parties, when signing a document, reconstructing the $n-k$ secrets d_{t_i} for $i = k+1, \ldots, n$. This means that after signing one message, if a different subset is going to sign for the second message, one needs to rekey in some way. Hence, Rabin defines a share refreshing protocol which occurs after a signing operation. The signing stage requires the interaction of all k signing parties, so as to reconstruct the $n-k$ missing secrets, and the share refreshing protocol requires the interaction of all n parties.

In [12] Shoup gives a scheme which is completely non-interactive in that each party signs the message independently and a seperate, public, share combining algorithm is used to combine k of the signature shares into a valid full RSA signature. There is no need for a share refreshing protocol and once the scheme is set up there is no need for any of the parties to interact. However, the set-up phase of the Shoup scheme requires a trusted dealer who knows the factors of the RSA modulus N.

Our scheme gives a threshold RSA scheme which combines some of the advantages of the previous schemes, and tries to reduce their disadvantages. In particular we define the following algorithms, with the following properties:

- **Dealing Algorithm:** An interactive protocol amongst the n players. Each player has an input d_i, which is their share of the unknown RSA private key d. At the end of this protocol the players agree on a threshold value k and some global public information S. In addition each player also obtains a public/private share (P_i, S_i) of the data needed to implement the threshold signature scheme. We note that this stage does not require a trusted third party in our scheme.
- **Subset Presigning Algorithm:** This is an interactive protocol amongst k members $I = \{t_1, \ldots, t_k\}$ of the n parties. The protocol results in public data D_I which is used by the share combining algorithm to generate a valid full RSA signature from the signature shares. The protocol results in each of the k parties holding some secret information S_{I,t_i} which depends on the subset I. This protocol is interactive, but only needs to be run once for each subset I.
- **Signature Share Generation Algorithm:** This algorithm takes as input a subset I as above, the secret information S_{I,t_i} and a message m. The result is a partial signature σ_{I,t_i} on the message m.
- **Signature Share Verification Algorithm:** This takes as input a signature share σ_{I,t_i} on the message m and verifies that it is validly formed using the public information D_I and P_{t_i}.
- **Share Combining Algorithm:** This takes as input the public information S, P_i and D_I, plus a message m and the partial signature shares σ_{I,t_i} for all $t_i \in I$, and then produces a valid full RSA signature σ. Or returns fail if one of the signature shares is invalid.

Hence, the main advantages of our scheme over those of Rabin and Shoup are that we do not require a trusted dealer (as Shoup's scheme does) and we do not require to rekey or interact once the Subset Presigning algorithm has been implemented for a given subset I. The main disadvantage is that the signature share generation algorithm needs to know which subset of shares are going to be combined later on.

3 Scheme Definition

In this section we describe our scheme and justify that it works.

3.1 Dealing Algorithm:

The parties $i = 1, \ldots, n$ first agree on a number of parameters.

- A prime number M where $M > N$.
- The threshold value k where $1 < k < n$.
- An element g of high order in $\mathbb{Z}_N^*$.

Each party i picks a random degree $(k-1)$ polynomial $f_i \in \mathbb{Z}_M[x]$ where

$$f_i(x) = a_{i,k-1}x^{k-1} + \cdots + a_{i,1}x + d_i$$

This ith party then computes $f_{i,j} = f_i(j)$ and then privately sends $f_{i,j}$ to party P_j, for all $j = 1, \ldots, n$. Note that the $f_{i,j}$ for $j = 1, .., n$ are the standard k-out-of-n Shamir sharing [11] of d_i. In addition, the ith party also computes $b_{i,j} = g^{a_{i,j}} \pmod N$ for $j = 0, \ldots, (k-1)$, where we let $a_{i,0} = d_i$. He then broadcasts $b_{i,0}$, .., $b_{i,(k-1)}$ to all other parties, these are the commitments of all the coefficients of the polynomial $f_i(x)$.

At this point each party j receives $f_{1,j}, .., f_{n,j}$. Player j verifies that:

$$\begin{aligned} g^{f_{i,j}} &= g^{f_i(j)} \pmod N = g^{[a_{i,k-1}j^{k-1} + \ldots + a_{i,1}j + d_i]} \pmod N \\ &= \prod_{t=0}^{k-1} (g^{a_{i,t}})^{j^t} \pmod N \\ &= \prod_{t=0}^{k-1} b_{i,t}^{j^t} \pmod N. \end{aligned}$$

If this does not hold then the protocol is aborted, as one knows that player i is not honest.

We finally set

$$\begin{aligned} S &= \{k, M, g\}, \\ P_i &= \{\{b_{j,l}\}_{j=1\ldots n, l=0,\ldots,k-1}\}, \\ S_i &= \left\{d_i, \{a_{i,j}\}_{j=1}^{k-1}, \{f_{j,i}\}_{i \neq j}\right\}. \end{aligned}$$

3.2 Subset Presigning Algorithm:

Recall in this stage we have as input $I = \{t_1, \ldots, t_k\}$ and $I' = \{1, \ldots, n\} \setminus I = \{t_{k+1}, \ldots, t_n\}$. This protocol is executed amongst the k parties represented by the set I.

Each party $t_i \in I$ computes

$$\begin{aligned} \lambda_{t_i} &= \prod_{1 \le j \le k, j \neq i} \frac{t_j}{t_j - t_i} \pmod M, \\ s_{t_i} &= \left(\sum_{j=k+1}^{n} f_{t_j, t_i} \right) \lambda_{t_i} \pmod M. \end{aligned}$$

They also compute $h_{t_i} = g^{s_{t_i}} \pmod N$, which is the commitment to the share s_{t_i} of party t_i. This commitment will help other parties to verify the signature share generated by party t_i later on in the scheme.

Note that we have

$$\begin{aligned}
s_{t_1} + \cdots + s_{t_k} &= (\sum_{i=k+1}^{n} f_{t_i,t_1})\lambda_{t_1} + \cdots + (\sum_{i=k+1}^{n} f_{t_i,t_k})\lambda_{t_k} \pmod{M} \\
&= \sum_{i=1}^{k} (f_{t_{k+1},t_i}\lambda_{t_i}) + \cdots + \sum_{i=1}^{k} (f_{t_n,t_i}\lambda_{t_i}) \pmod{M} \\
&= d_{t_{k+1}} + \ldots + d_{t_n} \pmod{M}
\end{aligned}$$

The last equality follows from the properties of Shamir's secret sharing scheme and the fact that M is a large prime.

Over the integers we then have that

$$\sum_{i=1}^{k} s_{t_i} = d_{t_{k+1}} + \cdots + d_{t_n} + x_I M$$

for some integer $x_I \in [k-n, k]$. That x_I lies in such a small interval is because s_{t_i} and d_{t_i} are both positive and smaller than M for all i.

To determine the value of x_I the parties then produce their signature share σ_{I,t_i} on the dummy message, $m' = 2^e \pmod{N}$. That is they compute and broadcast,

$$c'_{t_i} = 2^{e(d_{t_i} + s_{t_i})} \pmod{N}.$$

plus a proof of its correctness. The signature share verification algorithm is then executed, to check that c'_{t_i} is valid.

The value of x_I can then be computed via exhaustive search by verifying which value makes the following equation hold,

$$\begin{aligned}
\prod_{t_i \in I} c'_{t_i} &= 2^{e\left(\sum_{t_i \in I} d_{t_i} + s_{t_i}\right)} \pmod{N} \\
&= 2^{e\left(x_I M + \sum_{i=1}^{n} d_i\right)} \pmod{N} \\
&= 2^{e x_I M + ed} \pmod{N} \\
&= 2 \cdot \left(2^{eM}\right)^{x_I} \pmod{N}.
\end{aligned}$$

We finally set

$$\begin{aligned}
D_I &= \left\{x_I, \{h_{t_i}, c'_{t_i}\}_{t_i \in I}\right\}, \\
S_{I,t_i} &= \left\{s_{t_i}\right\}.
\end{aligned}$$

Notice, that from c'_{t_i} the correctness of x_I can be verified publicly.

3.3 Signature Share Generation Algorithm:

The parties now want to obtain a signature on the message m. Each party t_i computes

$$c_{t_i} = m^{d_{t_i} + s_{t_i}} \pmod{N}$$

along with a (non-interactive) proof that

$$\begin{aligned}
\mathtt{D}Log_m c_{t_i} &= d_{t_i} + s_{t_i} \\
&= \mathtt{D}Log_g \left(g^{d_{t_i}+s_{t_i}}\right) \\
&= \mathtt{D}Log_g \left(b_{t_i,0} \cdot h_{t_i}\right).
\end{aligned}$$

This proof can be produced using a standard adaption of the protocol of Chaum and Pederson [4] to a group with unknown order. We let this proof be denoted by $\mathcal{P}(m, c_{t_i}, t_i)$. The signature share is then given by

$$\sigma_{I,t_i} = \{c_{t_i}, \mathcal{P}(m, c_{t_i}, t_i)\}.$$

3.4 Signature Share Verification Algorithm:

Each signature share $\sigma_{I,t_i} = \{c_{t_i}, \mathcal{P}(m, c_{t_i}, t_i)\}$ can be verified by checking whether the proof $\mathcal{P}(m, c_{t_i}, t_i)$ holds.

3.5 Share Combining Algorithm:

At this point in the scheme, the signature shares

$$\sigma_{I,t_i} = \{c_{t_i}, \mathcal{P}(m, c_{t_i}, t_i)\}$$

have been published and verified to be correct. To produce the final signature on the message m we compute

$$\begin{aligned}
m^{-x_I M} \prod_{i=1}^{k} c_{t_i} &= m^{-x_I M} \prod_{i=1}^{k} m^{d_{t_i}+s_{t_i}} \pmod N \\
&= m^{-x_I M + \sum_{i=1}^{k} d_{t_i}+s_{t_i}} \pmod N \\
&= m^{\sum_{i=1}^{n} d_i} = m^d \pmod N \\
&= s.
\end{aligned}$$

4 Discussion of the Above Scheme

4.1 Interactiveness

If the k parties I want to sign a different message, they need to come back to the Signature Share Generation Algorithm. This is because the values of x_I, s_{t_i} etc can be reused. signature share generation and combining the signature shares.

If there is change in the threshold set parties, from I to I', then all the shares $s_{t'_i}$, for $t'_i \in I^{'}$, will have to be calculated again as their values depend on the set $I^{'}$. So that the parties need to come back to the Subset Presigning Algorithm stage. A new value of $x_{I'}$ needs to be determined. Note that the threshold value, k, still remains the same in this case.

If all parties agree to decrease the threshold value, k, then all of them need to run the protocol from the beginning, i.e. the Dealing phase. Also note that the threshold value can only be decreased, but not increased.

Hence, the scheme is partially interactive since of the five main stages of the protocol the only interactive stages are the Dealing and Subset Presigning Algorithms.

4.2 Share Refreshing

In Rabin's threshold scheme, [10][Figure 3], the additive shares of the absent parties must be reconstructed at one place or by a single party. The reason for this requirement arises from the difficulty in working with modulo $\phi(N) = (p-1)(q-1)$ that remains unknown to all parties.

As a consequence, all the n parties have to refresh or renew their additive shares of the secret exponent d whenever there is a change in either the threshold value of k or the set of the threshold parties. Note that the value of d still remains the same, what differs are the individual shares of each party. If they do not renew their shares, then a certain single party might end up to know the shares of all other parties in the scheme, for example, if we have three parties P_1, P_2, and P_3 and (2,3) threshold scheme:

- In the first signature, party P_2 is away and its share is reconstructed by party P_1. So P_1 now knows shares d_1 and d_2.
- In the second signature, party P_3 is away and its share is reconstructed by party P_1 again. So P_1 now knows shares d_1, d_2, and d_3. That means the first party knows the shares of all parties.

This problem leads Rabin's scheme to require a Share Refreshing stage, which is interactive and requires the presence of all parties who wish to continue in the scheme.

In contrast to Rabin's scheme, in our scheme each individual share, d_i, is reconstructed in a way that does not further reveal its value to any party after any number of changes in the set of threshold parties. As a result, the scheme not only does not leak any information about the share to any one else but also avoids the refreshing procedure that requires the interaction of all parties in the scheme.

4.3 Robustness

Even through the protocol is always $(k-1)$ private, i.e. no information is leaked when up to $(k-1)$ patries corrupt, it does not mean the combining signature process will be always successful.

If the number of dishonest parties is l, and $(n-k) < l < k$, then the number of honest parties is $(n-l)$ and $n-l < n-(n-k) = k$. That means a valid singature on a message cannot be obtained. As a result, to make sure that this never happens we require that $(n-k) > k$, i.e. $k < \lfloor \frac{n}{2} \rfloor$.

Acknowledgements

I would like to thank my supervisor, Professor Nigel Paul Smart, who has taught me cryptography throughout my three years at the University of Bristol. Many thanks to members of the security research group in Bristol for stimulating discussions on not only my results but also the threshold cryptography area.

References

1. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. *Proc. 1st ACM Conference on Computer and Communications Security, 1993*, 62–73, 1993.
2. D.Boneh and M. Franklin. Efficient generation of shared RSA keys. *Advances in Cryptography – CRYPTO '97*, Springer-Verlag LNCS 1233, 425–439, 1997.
3. C. Boyd. Digital Multisignatures. *Cryptography and Coding 1989*, 241–246, Clarendon Press, 1989.
4. D.Chaum and T. Pedersen. Wallet databases with observers. *Advances in Cryptology – CRYPTO '92*, Springer-Verlag LNCS 740, 89–105, 1992.
5. Y. Desmedt. Society and group oriented cryptography: an new concept. *Advances in Cryptography – CRYPTO '87*, Springer-Verlag LNCS 293, 120–127, 1987.
6. Y. Desmedt and Y. Frankel. Threshold Crypto-Systems. *Advances in Cryptography - CRYPTO '89*, Springer-Verlag LNCS 435, 307–315, 1989.
7. Y. Frankel. A practical protocol for large group oriented networks. *Advances in Cryptology – EUROCRYPT '89*, Springer-Verlag LNCS 434, 56–61, 1989.
8. R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Robust and efficient sharing of RSA functions. *Advances in Cryptology – CRYPTO '96*, Springer-Verlag LNCS 1109, 157–172, 1996.
9. T. Pederson. A threshold cryptosystem without a trusted dealer. *Advances in Cryptology – EUROCRYPT '91*, Springer-Verlag LNCS 547, 522–526, 1991.
10. T. Rabin. A simplified approach to threshold and proactive RSA. *Advances in Cryptology – CRYPTO '98*, Springer-Verlag LNCS 1462, 89–104, 1998.
11. A. Shamir. How to share a secret. *Communications of the ACM*, **22**, 612–613, 1979.
12. V. Shoup. Practical threshold signatures. *Advances in Cryptology – EUROCRYPT 2000*, Springer-Verlag LNCS 1807, 207–220, 2000.