AN EFFICIENT PROACTIVE RSA SCHEME FOR LARGE-SCALE AD HOC NETWORKS

Ruishan Zhang and Kefei Chen
Department of Computer Science, Shanghai Jiaotong University
{zhang-rs,kfchen@sjtu.edu.cn}

Abstract In this paper, we present an efficient proactive threshold RSA signature scheme for large-scale ad hoc networks. Our scheme has two advantages. Firstly, the building blocks of the whole scheme are proven secure. Secondly, the whole scheme is efficient.

Keywords: Ad hoc networks, Threshold signature, Proactive secret sharing, RSA

Introduction

Proactive threshold signature is very important to tolerate a more powerful, mobile adversary [OY][HJKM].Large-scale ad hoc networks have hundreds or even thousands of network nodes. Generally, all network nodes have shares of the secret key (private key), and only a small number of nodes could be corrupted. That is, n is very large and far larger than t. Most current proactive RSA schemes are not designed for this purpose [FGMYa][FGMYb][Rab]. To the best of our knowledge, the only proactive RSA scheme is URSA [LKZL]. Unfortunately, the scheme has proved faulty [JSY].

In this paper, we present an efficient proactive threshold RSA signature scheme for large-scale ad hoc networks. Our scheme includes four protocols: the key distribution protocol, the signature generation protocol, the share refreshing protocol and the share distribution protocol. Our scheme has two advantages. Firstly, the building blocks of the whole scheme are proven secure. Secondly, the whole scheme is efficient. The efficiency of our scheme is approximate to that of the scheme of Wong et al.

In our scheme,an ad hoc network consists of $P_1, P_2, ..., P_n$ nodes . There are two types of nodes: R (refreshing) nodes and S (signing) nodes. There are $2t + 1$ R nodes, which perform the share refreshing protocol. All nodes are S nodes, which perform signing operations.

The remaining paper is organized as follows. The initial key distribution protocol, the share refreshing protocol, the share distribution protocol and the

Input: secret key $d_i \in Z_{\phi(N)}$,N,g,n,t, RIDList, PIDList
1. Choose and hand P_i $d_i \in_R [-nN^2, nN^2]$ for $i \in RIDList$, set $d_{public} = d - \sum d_i$.
2. Compute and broadcast the witness $w_i = g^{d_i} mod N$ for $i \in RIDList$.
3. Share the value d_i using the sharing Z_n-VSS on input d_i, N,g,n,t, RIDList, PIDList.

Figure 1. Initial key distribution

signature generation protocol are presented in Section 1,2,3,4 respectively. In Section 5, some discussions are given.

1. Initial key distribution

The key distribution protocol is used to distribute the initial secret shares to $2t + 1$ R nodes. Before distributing the secret key, we assume that a set-up process has been carried out in which the RSA key generation took place and the RSA key pair has been computed. Denote the public key by (e, N) where $N = pq$ and p, q are primes of the form $p = 2p' + 1$,$q = 2q' + 1$ and p',q' themselves prime. The private key is d where $ed \equiv 1 mod \phi(N)$. In addition, an element g of high order is chosen as $g = g_0^{L^2}$ where g_0 is an element of high order and $L = n!$. As shown in Figure 1, the protocol consists of three steps. First, the private RSA key d is shared by generating additive secret shares $d_i$such that $d = d_{public} + \sum_j d_j$. Then, the witnesses for the additive shares are generated in the second step. Finally, each additive share is backed-up using a protocol Z_n-VSS, which is depicted in Figure 2. Here polynomial secret shares of the additive share are sent to $4t+2$ nodes, of which $2t+1$ nodes are R nodes, the other nodes $2t + 1$ are one-hop or two-hop neighbors. Note that not d_i , but $d_i L^3$ is shared in Z_n-VSS. After the initial key distribution protocol, every party achieves their polynomial shares of $d_i^{poly} = \sum f_j(i, 0)$ of $(d - d_{public})L^3$, where d_j is shared by $f_j(x, y)$. Here $(d - d_{public})L^3$ is shared by $F(x, y) = \sum f_j(x, y)$, where $F(x, 0) = \sum f_j(x, 0)$. We call $F(x)$ Joint-Z_n-VSS. For differentiation, we call d_i^{poly},d_i the polynomial secret share and the additive secret share, respectively. After the initial key distribution protocol, a group of $2t + 1$ nodes within one-hop or two-hop distance hold the polynomial secret shares of the secret key. (RIDList, PIDList are R node ID list and node ID list of $2t + 1$ nodes within one-hop or two-hop distance).

The details of Z_n-VSS are shown in Figure 1. In sharing stage, the dealer computes a two-dimensional sharing of the secret $s \in [-nN^2, nN^2]$ by choosing a random bivariate polynomial $f(x, y)$ of degree at most t with $f(0, 0) = SL^3$. It commits to $f(x, y) = \sum_{j,l=0}^{t} f_{jl} x^j y^l$ by computing a matrix $C =$

Input: secret value $s \in [-nN^2, -nN^2]$ and N,g,n,t , RIDList, PIDList
Sharing steps:
1.The dealer chooses $f_{00}, f_{01}, ..., f_{t-1t}, f_{tt} \in [-nL^3N^3, nL^3N^3]$ and $L|f_{00}, f_{01}, ..., f_{t-1t}, f_{tt}$, then defines $f(x,y) = \sum_{j,l=0}^{t} f_{jl}x^j y^l$ with $f_{00} = 0$.
2. Compute $a_i(y) = f(i,y), b_i(x) = f(x,i)$ for $i \in (RIDList \cup PIDList)$, and $C_{iy} = g^{f_{iy}} mod N$ for $i, y \in [0,t]$.
3. Hand node P_i the polynomial $a_i(y), b_i(x)$ and broadcast C_{iy} for $i, y \in [0,t]$.
Verification steps:
1. Node P_i use $verify{-}poly(a,i,C)$ $verify{-}poly(b,i,C)$to verify if $a_i(y), b_i(x)$ is correct. If the verification fails, P_i requests that the dealer make $a_i(y), b_i(x)$ public.
2. The dealer broadcasts all polynomial requested in the previous step. If the dealer fails to do so, he is disqualified.
3. Node P_i carried out the verification of Step 1 for all public polynomial. If the verification fails, the dealer is disqualified.

Figure 2. Z_n-VSS

$\{C_{jl}\}$ with $C_{jl} = g^{f_{jl}} mod N$ for $j, l \in [0,t]$. Then the dealer sends to every node P_i the share polynomials and broadcast the commitment matrix C. When node P_i receives $a_i(y), b_i(x)$ and C, it use $verify - poly(a,b,i,C)$ to verify if $a_i(y), b_i(x)$ are correct. If the verification is ok, P_i computes and keeps $a_i(0) := f(i,0)$ as its share. The reconstruction stage is straightforward and omitted.

In Z_n-VSS, $verify{-}poly(a,b,i,C)$, $verify{-}point(\alpha,\beta,i,m,C)$, $verify{-}share(\sigma,m,C)$ are employed to verify that the given polynomial, the given point and the given share are correct.

The message complexity and the communication complexity is $O(t^3)$ and $O(t^5k)$ (k is the security parameter), respectively.

2. Share refreshing

The essence of the share refreshing protocol is that each party splits his additive-share d_i into sub-shares d_{ij} which sum up to d_i , and gives each party P_j such a sub-share d_{ij}. The details are shown in Figure 3, including seven steps. The message complexity and the communication complexity of protocol are $O(t^3)$ and $O(t^5k)$, respectively.

3. Share distribution

In the share distribution protocol, S nodes obtain their secret shares from their neighbors. First, those $2t + 1$ nodes, which have been refreshed in the

Public information: N,g,n,t and w_i (for $i \in RIDList$), RIDList, PIDList
Input of party P_i : secret share d_i such that $g^{d_i} = w_i$
1. Party P_i randomly chooses $d_{ij} \in_R [-N^2, N^2]$ for (for $j \in RIDList$), set $d_{i,public} = d_i - \sum d_{ij}$, computes and broadcast $g_{ij} = g^{d_{ij}} mod N$.
2. P_i sends to P_j the value d_{ij} .
3. Verification of distribution of proper share size and public commitments: P_j verifies that $d_{ij} \in [-N^2, N^2]$ and $g_{ij} = g^{d_{ij}} mod N$ if not then he requests that d_{ij} be made public and set g_{ij} to g raised to this public value.
4. If P_i does not cooperate in Step 3 then d_i is reconstructed.
5. Verification that the sub shares in fact sum up to the previous share of P_i : P_j verifies that $w_i = g^{d_{i,public}} \prod g_{ij} mod N$ if not then d_i is reconstructed.
6. P_i computes his new share d_i^{new} and shares it. This results in a value g^{sL^3} where s is the secret that shared.
7. If P_i fails to share his secret or $(g^{d_i^{new}})^{L^3} \neq g^{sL^3} mod N$ then each party P_j exposes d_{ij} . If P_j fails to expose d_{ij} , then d_j is reconstructed by all parties.

Figure 3. Share refreshing

share refreshing protocol, update the shares of their neighbors. Then their neighbor nodes update other nodes in a diffused way. A node can obtain his share polynomial from his $2t + 1$ neighbor nodes by interpolating. Each local member P_r sends a message containing point $a_i(r), b_i(r)$ to node P_i . Then P_i interpolates its polynomial $a_i(y), b_i(x)$ and obtains $a_i(0)$ as its share. The message complexity and communication complexity of the whole scheme are $O(t)$ and $O(tk)$, respectively.

4. Signature generation

A signature share on a message m is generated as follows. Let H and be a hash function. The signature share of P_j consists of $x_i = x^{4Ld_i^{poly}}$. Suppose we have valid shares from a set of I parties, where $I = \{P_1, P_2, ..., P_{t+1}\}$. Before combining shares, we define $\lambda_{i,j}^I = L \frac{\prod_{P_{j'} \in I/\{P_j\}} i - j'}{\prod_{P_{j'} \in I/\{P_j\}} j - j'} \in Z$ for any $P_i \in [1, n]/I$ and $j \in I$. Clearly, these values are integers. From the Lagrange interpolation formula, we have $LF(i) = \sum \lambda_{i,j}^I F(j)$. Then we compute $y' = x^{4L^4 d_{public}} x_1^{\lambda_{0,1}^I} ... x_{t+1}^{\lambda_{0,t+1}^I} mod N = x^{4L^4 d} mod N$ such that $y'^e = x^{4L^4} mod N$. Since e is a prime and larger than n, $gcd(4L^4, e) = 1$. Applying the extended Euclidean algorithm on e and $4L^4$ to compute a and b such that $4L^4 a + eb = 1$, then we achieve signature $y = y'^a x^b$ of the message m such that $y^e \equiv x mod N$

. The message complexity and the communication complexity are $O(t), O(tk)$, respectively.

5. Discussions

The initial key distribution protocol is carried out at the onset of the system to distribute shares to $2t + 1$ R nodes. The share refreshing protocol is carried out to update the old shares of $2t + 1$ R nodes at the beginning of the every phase. After the initial key distribution or the share refreshing protocol, the share distribution protocol is performed to distribute secret shares to all other nodes. After nodes obtain their secret shares, they can employ these secret shares to perform signing using the signature generation protocol. As we point out in Section 1, the only proactive RSA scheme for large-scale ad hoc networks is the URSA scheme. However, URSA is insecure. Compared to URSA, our scheme is proven secure. Furthermore, the efficiency (the message complexity and the communication complexity) of our scheme is approximate to that of URSA. Due to space limitation, we only give some brief discussions. Both signature generation are based on polynomial secret shares, so the efficiency is similar. Our share distribution protocol is more efficient than that of URSA. The share refreshing protocol and the initial key distribution protocol of URSA are more efficient than ours.

References

R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In Proc. 10th ACM Symposium on Principles of Distributed Computing (PODC), pages 51-59, 1991.

A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or how to cope with perpetual leakage. In Advances in Cryptology CRYPTO '95 (D. Coppersmith, ed. Springer.). 963:339-352, 1995

Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. SIAM J. Computing, 12(4), 1983.

Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Optimal-Resilience Proactive Public-Key Cryptosystems. In Foundations of Computer Science FOCS'97, pages 384-393, 1997.

Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Proactive RSA. In Proc. of Crypto'97, pages 440-454, 1997.

T. Rabin. A simplified approach to threshold and proactive RSA. in Proc. CRYPTO '98, pp. 89-104, Springer, 1998.

Haiyun Luo, Jiejun Kong, Petros Zerfos, Songwu Lu, and Lixia Zhang. URSA: Ubiquitous and Robust Access Control for Mobile Ad Hoc Networks, IEEE/ACM Transactions on Networking (ToN), pp.1049 - 1063,12(6),2004.

Stanislaw Jarecki, Nitesh Saxena, and Jeong Hyun Yi. Cryptanalyzing the Proactive RSA Signature Scheme in the URSA Ad Hoc Network Access Control Protocol. In ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN), October 2004.