Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 171 (2007) 93–105
www.elsevier.com/locate/entcs

Efficient and Adaptive Threshold Signatures for Ad hoc networks

Roberto Di Pietro[1,2]

Dipartimento di Matematica
Università di Roma Tre
Rome, Italy

Luigi Vincenzo Mancini[3]

Dipartimento di Informatica
Università di Roma "La Sapienza"
Rome, Italy

Giorgio Zanin[4]

Dipartimento di Informatica
Università di Roma "La Sapienza"
Rome, Italy

Abstract

In this paper, we propose a secure, flexible, robust and fully distributed signature service, for ad hoc groups. In order to provide the service, we use a new threshold scheme, that allows to share a secret key among the current group members. The novelty of the scheme is in that it easily and efficiently enables dynamic increase of the threshold, according to the needs of the group, so that the service provides both adaptiveness to the level of threat the ad hoc group is subject to, and availability. We prove the correctness of the protocol and evaluate its efficiency. The changes to the threshold are performed by using a protocol that is efficient in terms of interactions among nodes and per-node required resources, resulting suitable even for resource-constrained settings. Finally, the same proposed scheme allows to detect nodes that attempt to disrupt the service, providing invalid contributions to the distributed signature service.

Keywords: threshold cryptosystem, signature scheme, secret sharing, peer-to-peer network, ad hoc networks, wireless network, ubiquitous service.

[1] The authors have been partially funded by the WEB-MINDS project, supported by the italian MIUR under the FIRB program.

[2] Email: dipietro@mat.uniroma3.it

[3] Email: mancini@di.uniroma1.it

[4] Email: zanin@di.uniroma1.it

1571-0661 © 2007 Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.entcs.2006.11.012

1 Introduction and Background

Mobile Ad Hoc Networks (MANETs) are inherently self-organized, large-scale and fault-tolerant networks, made of miniature wireless devices such as PDAs, cellular phones, Pocket PCs. They generally exhibit dynamic membership and topology, i.e. nodes can join and leave the network, as well as fail, over time. In particular, their size may change drastically in a short time period. MANETs, as in common ad hoc groups, can be characterized by a not-hierarchical structure, such that their nodes are autonomous *peers*. Due to their ability to provide a wide range of services, their use can be appealing for both military and civilian applications. Further, they are expected to be employed for providing pervasive services, playing a crucial role in any aspect of everyday life, impacting the way people work, and interact with each other, and the way business, education, entertainment, and health-care are operating, depicting the vision of anytime ubiquitous computing.

However, MANETs introduce new security challenges, compared to traditional wired or cellular wireless networks, due to: dynamic topology, produced by mobility and change in the number and distribution of active nodes in the network; severe resource constraints, that call for power-efficient protocols; absence of a trusted infrastructure; necessity for group oriented distributed services and protocols. These challenges have stimulated considerable research interest in recent years, because ad hoc networks cannot be adopted widespread, without adequate security. In particular, features such as decentralized operation and dynamic membership, bring a bulk of challenges from the perspective of security, such as member authentication, access control, secure group communication, and secure routing, to cite a few.

In this paper, we present a distributed signature scheme for ad hoc networks, that improves over current solutions. Basically, a *signature* $Sig(m)$ of a message m is an encryption of m using a secret key sk, that is $Sig(m) = [m]_{sk}$. In practice, any signature scheme is generally applied to hashed values of messages, rather than to the messages themselves; the hashed values are computed by using a strong hash function, such as SHA-1. Here, for ease of presentation, we assume that m is an already such hashed value. The signature needs to be verifiable by using a public key pk, that is, for any valid signature $Sig(m)$, it is required that $[Sig(m)]_{pk} = m$ holds.

In order to distribute a signature service, two tools are necessary: one for generating the $<sk, pk>$ pair, and one for distributing, among a set of distinct individuals, the ability to sign messages under sk. In the following, we assume the MANET to have associated an RSA key pair $<sk, pk>$. Current solutions, for distributing the signing ability to a group of individuals, are based on standard threshold schemes [4], in which sk is divided into a given number of *shares* that are distributed to as many cosigners.

The main limitation of standard threshold schemes is that they tolerate only a fix number of corruptions, regardless the dimension of the network. Indeed, in ad hoc settings, the members of the network typically exhibit physical vulnerabilities and they are exposed to attacks by adversaries able to compromise and/or corrupt likely

many of them. This is likely to occur for large-scale long-lived networks, because the probability to compromise a sufficiently large number (up to the threshold) of shares increases with the number of nodes. In order to limit the corruption power of a mobile adversary, generally *proactive secret sharing* [1] is employed. However, such a defence is not effective against *silent* adversaries that behave honestly, as long as they control a number of nodes below the threshold. We claim that, in order to cope with such a kind of adversary, it is necessary to increase the robustness of the secret sharing scheme employed, i.e. to increase the threshold. Unfortunately, typical secret sharing schemes found in the literature are affected by a major limitation, that is their robustness does not scale with the number of members in the network; in fact, increasing the robustness of these schemes generally requires several messages exchange among all the nodes of the network, that can be too expensive for MANET environments.

We propose a novel secret sharing scheme that additively shares sk among all the current members of the network, so that limited subgroups of them, provided with different shares, can act on behalf of the group, and collectively issue a signature. The scheme is basically a (t,t)-threshold scheme in which shares are *replicated*, and the value of the threshold t can be enlarged, in order to increase the robustness of the signature scheme. Each member belonging to the subgroup, separately and independently issues a partial signature, and all the partial signatures are combined afterwards by the applicant, in order to have a standard signature. Further, the dimension of the produced signature does not depend on the size of the subgroup, and it is possible to easily verify the signature correctness. Finally, it is possible to detect cosigners attempting to cheat.

The rest of the paper is organized as follows: in Section 2, we review the related literature. In Section 3, we propose the novel additive scheme for sharing an RSA secret key, and the share replication policy enforced. In Section 4, we present the protocols executed to issue a distributed signature and to increase the robustness of the scheme. In Section 5 we provide the proof of correctness of the signature scheme. In Section 6 we evaluate the efficiency of the scheme in terms of both space and computational requirements. Finally, in Section 7, we summarize the results.

2 Related Work

The general notion of threshold signature has been introduced in [4]. In [5] a non-robust and non-interactive RSA scheme is presented, with no security analysis. The scheme is extended in [8], with a new synchronized scheme, and further in [3] with a non-robust and non-synchronized scheme, in which the share size grows linearly with the number of cosigners. The linear growth of the share size limits its scalability. In [10, 9, 16] robust threshold RSA schemes that require synchronization are considered, and rigorously analyzed. These schemes are *proactive* threshold schemes [1], i.e. they periodically refresh the shares, without changing the signing key. The current best RSA-based scheme seems to be [16], however, if N is the RSA modulus and n is the number of cosigners, the size of the key used for the signature

generation is $2\log(nN)$ and the memory requirement is $2n\log(nN)$. Some schemes such as [2, 11] are *verifiable*, i.e. they permit to check whether the partial signatures and the newly created shares have been correctly computed. In [22] a robust, verifiable, and non-interactive scheme based on RSA is presented; the size of any individual signature is bounded by a constant times the size of the RSA modulus; however it assumes a static corruption model: the adversary must choose which cosigners to attack at the very beginning of the attack. All the previous schemes assume a limited number of cosigners: this number is considered fixed, and the system needs to restart if it changes. In [14] a scalable threshold RSA signature scheme is proposed: new shares can be computed from existing shares but, in order to keep secret the initial shares, an expensive synchronized interaction among existing cosigners is required. Moreover, as it has been shown in [15], the scheme is not robust. A recent work [18] permits to avoid any interaction among the existing cosigners. Other works [19, 20, 13] propose solutions based on threshold signatures for ad hoc and peer-to-peer environments, however they do not address the problem of increasing the threshold. In [7] a distributed signature scheme, specifically suitable for MANETs in which connectivity is partial, is presented. The scheme is robust against Byzantine adversaries, and does not need any trusted dealer; moreover, it is efficient, requiring little interaction for the signature issuance/verification phases.

Among the above schemes, only the two protocols in [9] permit to change the threshold, but this is achieved with multiple communication rounds, and needs the simultaneous on-line presence of all the cosigners. Another scheme that provides dynamic threshold is presented in [6]; the scheme does not require the intermediate reconstruction of the secret, and permits to transform a given threshold scheme into another, even between disjoint sets of cosigners; however, the scheme is not robust and compromised cosigners can distribute incorrect shares to newly admitted members. The same shortcoming can be referred to [9].

3 Distributed Signature Service

In this section, we assume the reader familiar with the RSA [17] signature scheme. The proposed additive secret sharing scheme is based on the following observation: $\forall m \in \mathbb{N}^+, v \odot \alpha \mod m \equiv v \mod m$, provided that α is the neutral element with respect to the operation $\odot$. Further, if $r_0 \odot r_1 \odot \ldots \odot r_{t-1} \equiv \alpha \mod m$, and $\odot$ is commutative and associative, it follows that:

$$(v \odot r_0) \odot \ldots \odot (v \odot r_{t-1}) \equiv \underbrace{v \odot \ldots \odot v}_{t \text{ times}} \mod m.$$

In $\mathbb{Z}_m$, by choosing $\odot = +$, than 0 is the neutral element. Hence, if we restrict t to be a positive odd number, and bind the sign of v to the index i, we have: $\sum_{i=0}^{t-1}((-1)^i v + r_i) \equiv v \mod m$. In order to ensure that $\sum_{i=0}^{t-1} r_i \equiv 0 \mod m$, it is enough to choose $r_i, i = 0, \ldots, t-2$ as random values in $\mathbb{Z}_m$ and $r_{t-1} \equiv -\sum_{i=0}^{t-2} r_i \mod m$, so that $\sum_{i=0}^{t-1} r_i \equiv 0 \mod m$, by construction.

Hence, a secret RSA exponent d can be shared among t cosigners (t odd), by using *shares* of kind:

$$s_i \equiv (-1)^i d + r_i \mod \phi(N), \quad i = 0, \ldots, t-1$$

where r_i is an integer such that, if t is the required threshold to issue a signature, then: $\sum_{i=0}^{t-1} r_i \equiv 0 \mod \phi(N)$. Moreover, in order to provide verifiability, each share s_i can be bound to a corresponding public *verifier*:

$$w_i \equiv (-1)^i + e \cdot r_i \mod \phi(N), \quad i = 0, \ldots, t-1$$

where e is the public RSA exponent, corresponding to d. Upon receipt of a share s_i, any cosigner can verify s_i's correctness, by ensuring that $(a^{s_i})^e \equiv a^{w_i} \mod N$, for an arbitrary a. Indeed $(a^{s_i})^e \equiv (a^{(-1)^i d + r_i \mod \phi(N)})^e \equiv a^{(-1)^i + e \cdot r_i \mod \phi(N)} \equiv a^{w_i} \mod N$. Note that, as long as $\phi(N)$ is unknown, w_i does not provide any information about the corresponding share, or about the secret signing key.

Similar arguments hold by working in $\mathbb{Z}$ instead of $\mathbb{Z}_m$, that is performing the operations showed above without computing the modulus. In particular, by considering $\mathbb{Z}$, it is possible to increase the robustness of the scheme of an arbitrary value k, by means of the following *share-split* procedure: a) by using a round robin policy, select a share s_i to split; b) choose k as a positive odd number; c) randomly select a share $s_j, j \neq i$; then, generate $k+1$ new shares: $s'_i = s_i + u_0$, $s_{t+g-1} = (-1)^g s_i + u_g, g = 1, \ldots, k$, and an update factor $s''_i = s_i + u_{k+1}$, such that $\sum_{j=0}^{k+1} u_j = 0$. Finally, replace s_i with s'_i and update s_j with $s'_j = s_j + s''_i$. At the end of the share-split procedure, k brand new shares $s_{t+g-1}, g = 1, \ldots, k$ have been created, and two shares s'_i, s'_j have been *proactively* updated. The corresponding verifiers are set to: $w'_i = w_i + e \cdot u_0$, $w_{t+g-1} = (-1)^g + e \cdot u_g, g = 1, \ldots, k$, and $w'_j = w_j + w_i + e \cdot u_{k+1}$. Note that we used equalities ($=$) instead of congruences ($\equiv$) since we were operating in $\mathbb{Z}$ instead of $\mathbb{Z}_m$. It can be proved that, unless the adversary controls s_i, the robustness of the scheme increases of an arbitrary odd factor $k \geq 1$, upon the completion of the share-split procedure. Note that this is performed by updating only two existing shares, namely s_i and s_j, without facing interpolation over $\mathbb{Z}_{\phi(N)}$ as in [14]. Moreover, the scheme can be extended to provide a proactive renewal of all the shares: it is enough to update each share s_i by a single update factor u_i, where the update factors are chosen such that $\sum_{i=0}^{t-1} u_i = 0$, and t is the current number of different shares.

The drawback of the proposed solution could be the growth of both the share and verifier size, because their values are computed over $\mathbb{Z}$, instead of $\mathbb{Z}_{\phi(N)}$. However, as we show in Section 6, this growth in size, that could have a severe impact on both communications and computations overhead, is very limited, making the solution feasible and efficient.

Unlike typical (t,n)-threshold schemes, in which each node is provided with a *different* share, here there are only t different shares, that are replicated: different cosigners are provided with the same share, according to the following distribution constraint: let x be the unique identifer of node D_x, s_i is distributed to D_x if and

only if:

$$F(x) \equiv i \pmod{t} \tag{1}$$

where F is a hashing function mapping the set of cosigner identifiers into $\{0, \ldots, \mathrm{t}-1\}$. As long as F is chosen uniformly at random from a universal hash family, the probability that two cosigners have the same share is less than or equal to $1/t$. This property is essential if we want the ad hoc network to provide an ubiquitous service: different shares will be evenly distributed among the nodes, then the service could be provided at each node's locality. Basically, enforcing the distribution constraint (1) partitions the set of cosigners into t classes, $C_0^t, \ldots, C_{t-1}^t$, corresponding to shares $s_i,\ i = 0, \ldots, t-1$. With notation C_i^t, we denote the class of nodes corresponding to the i^{th} share, when t is the necessary number of shares for reconstructing the signing key, and with D_{j_i} we denote a member of C_i^t; in other words, each $D_{j_i} \in C_i^t$ is provided with s_i. Note that, in such a scheme, the minimum effort required to the adversary for denying the service is to compromise all the replicas of a given share, that is the effort to corrupt all the nodes into a given class. Moreover, since the shares are distributed by enforcing (1), it can be shown that, as the number of cosigners increases, all the classes have asymptotically the same size.

4 Protocols

In this section we present the protocols executed by the nodes of the network in order to issue a signature, and to increase the threshold. We make standard assumptions: each member D_x of the group has a unique nonzero identifier x, such as a MAC address; D_x is connected via a broadcast communication channel to the neighbors in its transmission range; D_x can fail, dynamically join or leave the network, and has a source of randomness. Moreover, we assume that there is a routing layer for reliable message delivery. Messages are exchanged through secure channels.

4.1 Setup

We assume that the network is initialized by a trusted dealer D, that chooses a group secret key $sk = (d, N)$, and shares sk in such a way that any set of t nodes, one for each class, is able to issue a signature. For each share s_i, D also computes a corresponding *verifier* w_i that can be used for misbehavior detection. Note that D needs only to initialize t initial nodes, while other nodes can be initialized with the cooperation of that initial set. After the setup phase, the dealer is not necessary any longer. D performs the following actions:

(i) create an RSA key-pair $<sk = (d, N), pk = (e, N)>$;
(ii) choose the original threshold t; t is a positive odd number;
(iii) generate t random numbers $r_i \in \left[-\frac{N-1}{2}, \frac{N-1}{2}\right]$ such that: $\sum_{i=0}^{t-1} r_i \equiv 0 \mod \phi(N)$;
(iv) compute $s_i \equiv (-1)^i d + r_i \mod \phi(N)$;
(v) compute $w_i \equiv (-1)^i + e \cdot r_i \mod \phi(N)$;
(vi) distribute shares to at least t members, enforcing (1);
(vii) publish $<pk,\ t,\ w_0, \ldots, w_{t-1}>$;
(viii) destroy any computed value.

4.2 Signature Issuance and Verification

Once shares $s_i,\ i = 0, \ldots, t-1$, have been distributed to $n \geq t$ initial cosigners, the signature service can start. Any applicant D_x can obtain a signature $Sig(m)$ for a message m with the consent of at least t cosigners belonging to different classes. The protocol consists of two rounds of communication:

(i) D_x sends a signature request to the group, specifying the message m;
(ii) upon receipt of the request, at least t cosigners $D_{j_i}, i = 0, \ldots, t-1$, one for each class, compute t partial signatures $pSig_{j_i}(m) = [m]_{s_i} \equiv (m)^{s_i} \mod N$, where $s_i \equiv (-1)^i d + r_i \mod \phi(N)$ is the cosigner's share. Then, each cosigner replies with the computed partial signature.

Note that each cosigner can perform its job asynchronously, i.e. it neither requires to coordinate with other nodes, nor to be aware of other node's identities or on-line presence.

Upon receipt of t partial signatures, D_x reconstructs $Sig(m) \equiv \prod_{i=0}^{t-1} pSig_{j_i}(m) \mod N$, and checks for its correctness, by ensuring that $Sig(m)^e \equiv m \mod N$; if the verification fails, D_x checks for correctness of each partial signature, ensuring that $pSig_{j_i}(m)^e \equiv m^{w_i} \mod N$. The action to be taken once a partial signature is recognized as fake is out of the scope of this paper.

4.3 Threshold increase

In order to increase the threshold, new shares are generated from existing ones, by using the share-split procedure, described in section 3, and a fraction of cosigners

are redistributed into the newly created classes. The protocol that we describe here permits to increase the scheme's robustness from value t to value $t + k$, with k a positive odd number, any time it is executed.

In accordance with a stated policy, a member D_{j_i} is *designated* to execute the split procedure on its share s_i. D_{j_i} distributes the new shares by means of the following protocol:

(i) D_{j_i} sends an increase threshold request to each member D_{l_i} in its current class, specifying the new share-verifier pair $< s'_{l_i}, w'_{l_i} >$;
(ii) D_{j_i} sends an update request to each member D_{l_j} of a randomly selected class, specifying the update factor $s''_i = s_i + u_{k+1}$;
(iii) finally, D_{j_i} broadcasts a list comprising of all the computed verifiers and $w'_{k+1} = e \cdot u_{k+1}$.

Upon receipt of the list and the new share-verifier pair, each D_{l_i} checks for the correctness of the received values, by ensuring that $w'_i + w'_j + \sum_{g=0}^{k-1} w_{t+g} = w_i + w_j$, and $(a^{s'_{l_i}})^e \equiv a^{w'_{l_i}} \mod N$, for an arbitrary a. If the check succeeds, D_{l_i} drops its current share, accepting s'_{l_i}, otherwise it raises a policy dependant exception; each D_{l_j} performs the same checks as D_{l_i}, and if they succeed, each D_{l_j} updates its current share $s'_j = s_j + s''_i$, otherwise it raises an exception.

Note that only those cosigners in C_i^t and in C_x^t take part in the protocol, that is, in order to increase the threshold, only a fraction of the cosigners need to cooperate. This permits to save network resources, by limiting the communication traffic and the overall computations performed by the nodes of the network.

At the end of the protocol, k new classes have been created, and they are filled with those cosigners who acquired the new shares, i.e. the members in C_i^t are reallocated into $C_i^{t+k}, C_t^{t+k}, \ldots, C_{t+k-1}^{t+k}$, where C_i^{t+k} replaces C_i^t. In order to evenly reallocate those cosigners, D_{j_i} enforces the following share distribution: a member with unique identifier l is assigned to class $C_{t+j}^{t+k}, 0 \leq j \leq k-1$, if and only if $F(l) \equiv j \mod k+1$, where F is a hashing function mapping the set of node identifiers into $\{0, \ldots, k\}$, otherwise it is assigned to C_i^{t+k}. Note that, as long as the selection of the share to split is performed with a round robin policy, and the original threshold is known by all the members, it is always possible to determine the class that a given member fell into at a given time, by determining how many executions of the protocol have been performed up to that time.

5 Correctness

The scheme is correct, as shown by the following lemma.

Lemma 5.1 *The cooperation among t cosigners belonging to different classes, where t is the current threshold, allows to issue a valid signature.*

Proof. *We consider the necessary cooperation among the cosigners in two cases: a) at any time between the setup phase and the first share-split; b) at any time between the q^{th} and the $(q+1)^{th}$ share-split, for an arbitrary q.*

a) If no share-split occurred, t partial signatures, with t odd, issued by using different shares, are necessary and sufficient to issue a signature $Sig(m)$. Indeed, since $\sum_{i=0}^{t-1} r_i \equiv 0 \pmod{\phi(N)}$ by construction, and t is odd:

$$\prod_{i=0}^{t-1} psig_{j_i}(m) \equiv \prod_{i=0}^{t-1} m^{(-1)^i d + r_i} \equiv m^d \cdot m^{\sum_{i=0}^{t-1} r_i} \equiv m^d \pmod{N} = Sig(m).$$

b) Assume now that q share-splits, $1 \leq q \leq t$, occurred, according to a round robin policy. Let $u_{i_g}, g = 0, \ldots, k+1$ (k odd), denote the random numbers chosen to split share s_i; note that, for any i, $\sum_{g=0}^{k+1} u_{i_g} = 0$ by construction; further, let s'_i the new value for s_i after the split, i.e. $s'_i = s_i + u_{i_0}$, and let $s'_j = s_j + s_i + u_{i_{k+1}}$ the new value of the proactively updated share s_j. After q share-splits, the number of shares required to issue a signature is $t' = t + qk$. In particular, without loss of generality, assume there are:

- *$t - 2q$ initial shares that are not affected by the share-splits: $s_h = (-1)^h d + r_h$, $h = 0, \ldots, t-2q-1$;*
- *q new shares: $s'_i = (-1)^i d + r_i + u_{i_0}$, $i = t-2q, \ldots, t-q-1$;*
- *q initial shares, proactively updated: $s'_j = (-1)^j d + r_j + s_i + u_{i_{k+1}}$, $i \in [t-2q, \ldots, t-q-1]$, $j = t-q, \ldots, t-1$;*
- *qk new shares $s_{i_g} = (-1)^g s_i + u_{i_g}$, $i = t-2q, \ldots, t-q-1$, $g = 1, \ldots, k$. Let l be the index of these shares, running as: $l = t, \ldots, t+qk-1$.*

By combining t' partial signatures, issued by using different shares, since k is odd and for any i, $\sum_{g=0}^{k+1} u_{i_g} = 0$ by construction, then:

$$\prod_{c=0}^{t+qk-1} psig_{j_c}(m) \equiv \prod_{h=0}^{t-2q-1} m^{s_h} \prod_{i=t-2q}^{t-q-1} m^{s'_i} \prod_{j=t-q}^{t-1} m^{s'_j} \prod_{l=t}^{t+qk-1} m^{s_l} \equiv$$
$$\prod_{h=0}^{t-2q-1} m^{(-1)^h d + r_h} \prod_{i=t-2q}^{t-q-1} m^{(-1)^i d + r_i + u_{i_0}} \prod_{j=t-q, i=t-2q}^{t-1, t-q-1} m^{(-1)^j d + r_j + s_i + u_{i_{k+1}}} \prod_{i=t-2q, g=1}^{t-q-1, k} m^{(-1)^g s_i + u_{i_g}} \equiv$$
$$\left(\prod_{h=0}^{t-2q-1} m^{(-1)^h d + r_h}\right) \cdot \left(\prod_{i=t-2q}^{t-q-1} m^{u_{i_0}} \prod_{i=t-2q}^{t-q-1} m^{(-1)^i d + r_i}\right) \cdot \left(\prod_{i=t-2q}^{t-q-1} m^{s_i} \prod_{i=t-2q}^{t-q-1} m^{u_{i_{k+1}}} \prod_{j=t-q}^{t-1} m^{(-1)^j d + r_j}\right) \cdot$$
$$\cdot \left(\prod_{i=t-2q}^{t-q-1} m^{-s_i} \prod_{i=t-2q, g=1}^{t-q-1, k} m^{u_{i_g}}\right) \equiv$$
$$\prod_{h=0}^{t-1} m^{(-1)^h d + r_h} \prod_{i=t-2q, g=0}^{t-q-1, k+1} m^{u_{i_g}} \equiv m^d \cdot m^{\sum_{h=0}^{t-1} r_h \pmod{\phi(N)}} \equiv m^d \pmod{N} = Sig(m).$$

The same arguments hold if $q > t$, considering that every share is equal to an initial share, added to a random factor $u_{x_g}, g \in [0, k+1]$ for some $x \in [0, t+qk-1]$.□

6 Efficiency

In this section we show that the number of extra bits required to support our scheme is limited by a small constant, with overwhelming probability, hence the solution is pretty feasible in resource-constrained environments such as MANETs, as well as in other ad hoc settings. Note that this analysis is necessary, because the operations for increasing the threshold are performed over the integers, not in modulus, then the share/verifier bit-length could increase with the number of share-splits performed, affecting both storage requirements and computational costs of our solution.

Share and verifier bit length. The analysis is based on three facts from elementary probability [12]. Let X be a random variable with uniform distribution in the range $[-\frac{N-1}{2}, \frac{N-1}{2}]$, μ_X be its expected value and σ_X^2 its finite variance. Then:

Fact 6.1 *$\mu_X = 0$ and $\sigma_X^2 \leq \frac{(N-1)^2}{12}$*

Fact 6.2 *for any constant $a > 0$, the random variable $Z = aX$ has finite variance $\sigma_Z^2 = a^2\sigma_X^2$.*

Fact 6.3 *for any constant $k > 0$, $P(|X - \mu_X| \geq k) \leq \frac{\sigma_X^2}{k^2}$.*

Let us now consider q share-splits that affect a given share s_i. Without loss of generality, assume s_i is one of the initial shares provided in the initialization phase by the trusted dealer. After q share-splits, the new share is $s_i' = \pm d + r_i \pmod{\phi(N)} + u_1 + u_2 + \ldots + u_q$, where $r_i \in \left[-\frac{N-1}{2}, \frac{N-1}{2}\right]$ has been randomly selected by the trusted dealer, and $u_j \in \left[-\frac{N-1}{2}, \frac{N-1}{2}\right], j = 1, \ldots, q$, are the numbers randomly selected by the designated member at the j^{th} share-split. Indeed, the share could have value $s_i' = \pm m \cdot d + r \pmod{\phi(N)} + u_1 + u_2 + \ldots + u_q, m \geq 0$, as effect of the split of other shares, in the event of s_i to be chosen for the proactive update. However, the expected value for m is 1. Let each u_j be modeled by a random variable U_j. Since U_j has uniform distribution over $\left[-\frac{N-1}{2}, \frac{N-1}{2}\right]$, from fact 6.1, it follows that $\mu_{U_j} = 0$ and $\sigma_{U_j}^2 = \frac{(N-1)^2}{12}$. Let be $U = \sum_{j=1}^{q} U_j$. By the linearity of the expectation, $E[U] = E[\sum_{j=1}^{q} U_j] = \sum_{j=1}^{q} E[U_j] = 0$. Further, since the random variables U_j are independent: $\sigma_U^2 = \sigma_{U_1+\ldots+U_q}^2 = \sum_{j=1}^{q} \sigma_{U_j}^2 = \sum_{j=1}^{q} \frac{(N-1)^2}{12} = \frac{q(N-1)^2}{12}$. Hence, by fact 6.3, for a given constant $b > 0$:

$$P(|U - \mu_U| \geq b \cdot N) = P(|U| \geq b \cdot N) \leq \frac{\sigma_U^2}{(b \cdot N)^2} = \frac{q(N-1)^2}{12b^2N^2} < \frac{q}{12b^2} \qquad (2)$$

We are interested in the probability that U is greater than bN and, in particular, we want this probability to be less or equal to a desired small value p. It turns out from (2) that the least value satisfying the inequality is $b = \frac{1}{2}\sqrt{\frac{qp^{-1}}{3}}$. Note that, in order to represent bN, at most $\lceil \log_2 bN \rceil = \lceil \log_2 b + \log_2 N \rceil$ bits are necessary. On the other hand, the initial share $s_i = d + r_i \pmod{\phi(N)}$ can be represented over at least $\lceil \log_2 N \rceil - 2$ bits. Hence, the necessary number of extra bits required for a

share by our solution is

$$extra\ bits = 2 + \lceil \log_2 b \rceil = 2 + \left\lceil \log_2 \left(\frac{1}{2} \sqrt{\frac{qp^{-1}}{3}} \right) \right\rceil \tag{3}$$

In order to figure out how (3) impacts on the length of the share, consider the following example: assume $q = 2^{30}$ share-splits affecting s_i have been performed, and $p = 2^{-30}$. Note that, by assuming a round robin policy for selecting the share to split, the actual total number of performed splits is much higher. It follows that $b = 2^{29}$, hence 31 extra bits are enough for any practical use.

Similar arguments hold for verifiers. After q splits of s_i, the corresponding verifier is: $w'_i = \pm 1 + e \cdot r_i \pmod{\phi(N)} + e(u_1 + u_2 + \ldots + u_q)$. Let be $W = \sum_{j=1}^{q} W_j = \sum_{j=1}^{q} e \cdot U_j = e \cdot U$. Note that e is a constant, then, by Fact 6.2, it follows that $\sigma_W^2 = e^2 \sigma_U^2$. Hence, by leveraging the results provided for the share, for a given constant c:

$$P(|W - \mu_W| \geq c \cdot N) = P(|U| \geq c \cdot N) \leq \frac{\sigma_W^2}{(c \cdot N)^2} < \frac{e^2 q}{12c^2} \tag{4}$$

By choosing $c = b \cdot e$, (2) and (4) yield the same value. It follows that, if $e = 2^{16} + 1$, as usual in common settings [21], a practical number of extra bits required for the verifier by our solution is $2 + \lceil \log_2 c \rceil = 2 + \lceil \log_2 (be) \rceil = 2 + \lceil \log_2 b \rceil + \lceil \log_2 e \rceil = 31 + 17 = 48$.

Table 1 summarizes the overhead incurred for each protocol execution: *prgs* is the number of pseudo-random generations required; *Applicant* is any individual requesting a signature; *Cosigner* is any node participating to the signature issuance; *Designated* is a current cosigner delegated to execute the share-split; t is the current threshold; s_i is the current share for members in class C_i^t, and w_i is the corresponding verifier; $|C_i^t|$ is the size of C_i^t; k is the number of newly created shares (that is, classes) at each share-split. We assume that any number a can be expressed in at most $\lceil \log_2 a \rceil$ bits, and the RSA exponentiations are performed with the Square-and-Multiply algorithm.

	Role	Operations			
		modular multiplications	Messages		prgs
			sent	received	
signature issuance	Applicant D_x	$t + 1.5\lceil \log_2 e \rceil + 1.5t\lceil \log(e \cdot w_i) \rceil$*	t	t	0
	Cosigner D_{j_i}	$1.5\lceil \log s_i \rceil$	1	1	0
threshold increase	Designated D_{j_i}	0	$\lvert C_i^t \rvert + \lvert C_j^t \rvert + 1$	0	k+1
	Member in C_i^t	$1.5\lceil \log(s_i \cdot e \cdot w_i) \rceil$	0	2	0
	Member in C_x^t	$1.5\lceil \log(s_i \cdot e \cdot w_i) \rceil$	0	2	0

Table 1
Overhead sustained by nodes in different roles.

7 Conclusions

In this paper, we have presented a secure, adaptive, robust and efficient mechanism to provide distributed signatures in ad hoc groups. In particular, we have proposed a dynamically adjustable signature threshold scheme. This feature, that allows to adapt to an increasing level of threat the ad hoc group is subject to, has been implemented by leveraging a peculiar property of RSA cryptosystem. Further, all the features of the proposed mechanism are achieved requiring just local communications and very little overhead, in terms of both memory and computations requirements. These appealing characteristics make these protocols particularly suitable for the mobile ad hoc environment, and appealing even for resource constrained devices. Finally, the proposed scheme allows to detect nodes that attempt to disrupt the distributed signature service by providing a non-valid contribution to the computation.

References

[1] Canetti, R., R. Gennaro, A. Herzberg and D. Naor, *Proactive security: Long-term protection against break-ins*, RSA CryptoBytes **3** (1997), pp. 1–8.

[2] Chor, B., S. Goldwasser, S. Micali and B. Awerbuch, *Verifiable secret sharing and achieving simultaneity in the presence of faults*, in: *Proceedings of the 26th IEEE Symposium on Foundation of Computer Science: FOCS'85*, 1985, pp. 383–395.

[3] De Santis, A., Y. Desmedt, Y. Frankel and M. Yung, *How to share a function securely*, in: *Proceedings of the 6th ACM Symposium on the Theory of Computing: STOC'94*, 1994, pp. 522–533.

[4] Desmedt, Y., *Society and group oriented cryptography: A new concept*, in: *Advances in Cryptology–CRYPTO'87, A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, Lecture Notes in Computer Science **293** (1988), pp. 120–127.

[5] Desmedt, Y. and Y. Frankel, *Shared generation of authenticators and signatures*, in: J. Feigenbaum, editor, *Advances in Cryptology–CRYPTO'91*, Lecture Notes in Computer Science **576** (1992), pp. 457–469.

[6] Desmedt, Y. and S. Jajodia, *Redistributing secret shares to new access structures and its applications*, Technical Report ISSE TR-97-01, George Mason University (1997).

[7] Di Crescenzo, G., R. Ge and G. R. Arce, *Improved topology assumptions for threshold cryptography in mobile ad hoc networks*, in: *SASN '05: Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks* (2005), pp. 53–62.

[8] Frankel, Y. and Y. Desmedt, *Parallel reliable threshold multisignature*, Technical Report TR-92-04-02, Department of EE & CS, University of Wisconsin-Milwaukee, Milwaukee, WI, USA (1992).

[9] Frankel, Y., P. Gemmell, P. Mackenzie and M. Yung, *Optimal resilience proactive public-key cryptosystems*, in: *Proceedings of the 38th Annual Symposium on Foundations of Computer Science: FOCS'97*, 1997.

[10] Frankel, Y., P. Gemmell, P. Mackenzie and M. Yung, *Proactive RSA*, in: *Advances in Cryptology: CRYPTO'97*, 1997.

[11] Gennaro, R., T. Rabin, S. Jarecki and H. Krawczyk, *Robust and efficient sharing of rsa functions*, Journal of Cryptology **13** (2000), pp. 273–300.

*The overhead $1.5t(\log e + \log w_i)$ is sustained only if the verification of the signature fails, in order to check for the correctness of the partial signatures.

[12] Haigh, J., "Probabilty Models," Springer Verlag, 2002.

[13] Jarecki, S. and N. Saxena, *Further simplifications in proactive rsa signatures.*, in: *TCC 2005: Theory of Cryptography, Second Theory of Cryptography Conference*, Lecture Notes in Computer Science **3378** (2005), pp. 510–528.

[14] Luo, H., J. Kong, P. Zerfos, S. Lu and L. Zhang, *URSA: Ubiquitous and Robust Access Control for Mobile Ad-hoc Networks*, IEEE/ACM Transactions on Networking (ToN) **12** (2004), pp. 1049–1063.

[15] Narasimha, M., G. Tsudik and J. H. Yi, *On the utility of distributed cryptography in p2p and manets: The case of membership control*, in: *ICNP '03: Proceedings of the 11th IEEE International Conference on Network Protocols* (2003), p. 336.

[16] Rabin, T., *A simplified approach to threshold and proactive RSA*, in: H. Krawczyk, editor, *Advances in Cryptology: CRYPTO '98*, Lecture Notes in Computer Science **1462** (1998).

[17] Rivest, R., A. Shamir and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM **21** (1978), pp. 120–126.

[18] Saxena, N., G. Tsudik and J. Yi, *Efficient node admission for short-lived mobile ad hoc networks.*, in: *Proceedings of the 13th International Conference on Network Protocols: ICNP'05*, 2005, pp. 269–278.

[19] Saxena, N., G. Tsudik and J. H. Yi, *Admission control in peer-to-peer: design and performance evaluation*, in: *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks* (2003), pp. 104–113.

[20] Saxena, N., G. Tsudik and J. H. Yi, *Identity-based access control for ad hoc groups.*, in: *Information Security and Cryptology - ICISC 2004, 7th International Conference*, 2004, pp. 362–379.

[21] Schneier, B., "Applied Cryptography: Protocols, Algorithms and Source Code in C," John Wiley & Sons, Inc., 1996, 2^{nd} edition, iSBN 0-471-12845-7.

[22] Shoup, *Practical threshold signatures*, in: B. Preneel, editor, *Advances in Cryptology: EUROCRYPT 2000*, Lecture Notes in Computer Science **1087** (2000), pp. 207–220.