A Distributed Certification System for Structured P2P Networks

François Lesueur, Ludovic Mé, and Valérie Viet Triem Tong

SUPELEC, SSIR Group (EA 4039)
Avenue de la Boulaie - CS 47601 - 35576 Cesson-Sévigné cedex - France
firstname.lastname@supelec.fr

Abstract. In this paper, we present a novel distributed certification system in which signing a certificate needs the collaboration of a fixed ratio of the nodes, hence a varying number of nodes. This number is dynamically adjusted to enforce the ratio in a fully distributed way, which is mandatory for decentralized varying-size P2P networks. A certificate allows then to link the key pair of a node to some rights granted to it.

Keywords: P2P, Security, Distributed Certification.

1 Introduction

P2P networks have been widely used for the last few years as they allow the design of low cost and high availability systems. These large networks are based on a fully distributed architecture, in which every user has an equivalent role.

There are two types of P2P networks: unstructured ones and structured ones. In unstructured P2P networks (Gnutella [1]), requests are broadcasted or routed through random walks. In structured ones (Chord [2]), requests are routed using generated routing tables. We consider here structured P2P networks security.

Since security techniques used in traditional centralized systems rely on trusted entities, directly using such techniques would break the P2P basics. It is thus crucial to provide fully distributed security mechanisms for P2P networks.

In this paper, we present a novel distributed certification mechanism. This certification mechanism fully distributes a Certification Authority: signing a certificate needs the collaboration of $t\%$ of the nodes. Then, considering such a certificate valid is similar to trusting $t\%$ of the nodes would not collude to create a false certificate. This mechanism relies on agreements by a static ratio of the nodes (and hence a dynamic number of nodes) and not by a static number of nodes as in [3]. Moreover, this adaptation is done in a fully distributed way, as opposed to [4]. A certificate contains the public key of its owner and any rights that may be needed by this owner (access rights, name ownership, ...). As far as we know, this is the first approach proposing a dynamic threshold in a large and distributed environment.

We also briefly present three applications for this distributed certification, mitigating the sybil attack [5], excluding misbehaving nodes and providing intelligible names for P2P Voice over IP rather than cryptographic key hashes.

D. Hausheer and J. Schönwälder (Eds.): AIMS 2008, LNCS 5127, pp. 40–52, 2008.

In Section 1, we present some related work. Then, in Section 2, we present our distributed certification system. In Section 3, we discuss the security of our system against attackers. In Section 4, we present experimental results. In Section 5, we present the suggested applications. Finally, we conclude and propose some future work.

2 Related Work

We first introduce structured P2P networks and we then present previous work on distributed certification.

2.1 Structured P2P Networks

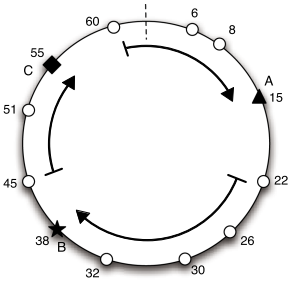


Fig. 1. Simplified representation of Chord with identifiers going from 0 to $2^6 - 1 = 63$. ▲, ♦ and ★ are nodes (PC) and ∘ are resources (files). Node A, which *nodeId* is 15, is responsible for resources 60, 6 and 8.

Structured P2P networks provide a virtual space called *overlay*. Each node (PC) is uniquely identified by a $nodeId \in \mathcal{KEYIDS}$; in the same way, each *resource* (file, ...) is uniquely identified by a key identifier $keyId$, $keyId \in \mathcal{KEYIDS}$ (for a file, the key is usually its SHA1 fingerprint). Nodes and resources thus share the same identifier space $\mathcal{KEYIDS}$, which is finite but supposed large enough (often 2^{160} elements due to the size of SHA1 fingerprints). Each node is responsible for the management of a part of the resources and the overlay provides routing facilities with a logarithmic cost for a node to access a specific identifier.

In Chord for instance, the identifiers space is represented as a ring going from 0 to $2^{160} - 1$ and each node is responsible for all the resources between it and its preceding node in the overlay. This organization is illustrated in Figure 1.

2.2 Distributed Certification

We first present threshold cryptography which is the basis of previous work on distributed certification as well as of our proposition. We then study previous work on distributed certification in peer groups.

Threshold Cryptography. In threshold cryptography, based on Shamir secret sharing [6], the enciphering of a message can only be achieved through the collaboration of a given number of entities. Threshold cryptography consists in splitting a secret key and distributing the resulting shares on different entities. We present here threshold cryptography based on [7].

(t, n)-threshold cryptography allows for enciphering a message with any t shares chosen among those issued to n entities, each entity usually owning one distinct share. t and n are predefined constants, set up at the initialization. If nobody knows the secret key, this key is better protected from misbehaving people. t shares are needed to encipher a message, but $t - 1$ shares hold *no* information on the secret key. An attacker must thus obtain t shares of the secret key to be able to recover the full key.

When some entity wants to obtain the signature of some data d, it asks t other entities to sign d with their own share. The signature of d by the secret key is then a combination of the t partial signatures. For instance, if the signature function f is homomorphic, i.e., $f(x+y) = f(x) \times f(y)$ (the RSA function is homomorphic), the combination of the partial signatures is simply their multiplication. Only the partial signatures are public and not the key shares. Such threshold cryptography schemes have been previously used to provide distributed certification.

Distributed Certification in Peer Groups. Distributing a certification process can be achieved through threshold cryptography. In [3], Kong *et al.* propose a distributed certification based on the cooperation of t nodes, t being a fixed number of nodes during the whole life of the system. The choice of t is a problem since $t = 3$ might for instance be a correct value for a network composed of 10 nodes but is clearly too small when the same network has grown to 1000 nodes: t should be a ratio of the number of nodes.

In [4], Saxena *et al.* propose to adapt the threshold t *dynamically* using algorithms proposed by Frankel *et al.* in [8]. A server manages a counter of the network size and detects the need for changing the threshold. Besides scaling and robustness problems in threshold changing algorithms, the reliance on a server is opposed to P2P bases and lowers the availability of the network. None of these schemes is thus sufficient for distributing certification in a P2P network.

In this paper, we propose a novel distributed certification scheme. In this scheme, the threshold is dynamically adjusted in a fully distributed way, which is mandatory for decentralized varying-size networks such as P2P ones.

3 A Distributed Certification System

In this section, we present our distributed certification mechanism in a structured P2P network. First, we describe the principle of the proposed certification system. Then, we present the sharing of the secret key used to distribute the certification process. Finally, we explain the distributed certification algorithm.

3.1 Principle

The network is characterized by an RSA public/secret key pair (P, S): P is publicly known and S is shared among the nodes (no node knows S). This key pair is generated by founding members using a distributed algorithm as Boneh and Franklin proposed in [9]. Certificates follow the X.509 format and contain

in particular the public key of their owners and some rights granted to these owners. They are signed with S through a novel threshold cryptography scheme.

Each solicited node of the P2P network decides locally if it should participate in the certification process. Such decisions are based on local observations, security policy or proofs included in the certification request (see Section 6). Obtaining a valid certificate requires the agreement and cooperation of a fixed *ratio* t of the members of the network. Then, trusting the validity of this certificate is equivalent to trusting that such a ratio of node would not collude to lie.

In the following, we make the assumption that there is no successful Sybil Attack [5] in the network, i.e., each person has only one connected node and its identifier is truly random. We discuss in Section 6.1 a self-healing node admission control, rejecting sybil nodes using our distributed certification operated by already accepted ones (which are thus non-sybil).

3.2 Sharing the Network Secret Key

The sharing of the network secret key is based on the homomorphic property of the RSA enciphering function. Generally, if $S = (e, m)$ denotes the RSA network secret key, we can pick s *shares* $e_1, \ldots, e_s$ such that $e = \sum_{i=1}^{s} e_i$ and then for any data d:

$$d^e[m] = d^{\sum_{i=1}^{s} e_i}[m] = \Big(\prod_{i=1}^{s} d^{e_i}[m] \Big)[m]$$

In other words, the RSA signature of d with S, which is $d^e[m]$, is equal to the product of the signatures with each share modulo m, m being publicly known since it appears in the network public key $P = (d, m)$.

If the security policy of the network defines that a certification has to be controlled by a ratio t of the nodes ($t \in [0, 1]$) and if n is the size of the network, then the secret key must be split in $s = t \times n$ shares. To provide a share to each member, each share e_i is then replicated on $g = \frac{1}{t}$ members composing a *sharing group*, each member knowing the list of his group. In order to sign a certificate, each share has to be involved and so a ratio t of the nodes must cooperate.

When the network grows (resp. shrinks), the number s of shares must grow (resp. shrink) in order to maintain $s = t \times n$, since t is a static ratio. However, each share should still be replicated on $g = \frac{1}{t}$ members, which is independent of the network size n. So, a sharing group can detect without knowing n (and hence without a central counter) if it is too small or too large.

When nodes join (resp. leave) the network, if a sharing group detects it is too large (resp. too small) to maintain $s = t \times n$, i.e., this group is composed of more (resp. less) than $\frac{1}{t}$ members, it splits and generates a new share (resp. merges with another group and discards one share). To create two shares e_{i0} and e_{i1} from a single share e_i, e_{i0} and e_{i1} must simply be chosen such that $e_{i0} + e_{i1} = e_i$ and e_i must be discarded. To prevent an attacker from keeping e_i and thus having a larger share than others, e_i is rendered useless by mixing e_{i0} and e_{i1} with two other randomly chosen shares e_x and e_y: a chosen random value Δ_0 (resp. Δ_1) is added to e_{i0} (resp. e_{i1}) and subtracted from e_x (resp. e_y).

After this operation, the sum of shares is still e but e_i is not part of the sharing anymore. Creating one share from two shares is exactly the opposite operation of splitting. These operations only involve members of the concerned sharing groups and not the whole network.

However, only creating two sharing groups from one (resp. one from two) does not allow to have each group composed of exactly $\frac{1}{t}$ members. In fact, we define two bounds g_{min} and g_{max} for merging and splitting sharing groups, which are respectively the minimum and maximum size of a sharing group. We thus have $\frac{1}{g_{max}} < t < \frac{1}{g_{min}}$ and we expect s to roughly equal $t \times n$. Finally, when a sharing group grows to g_{max} members, it is split in two groups of size $g_{new} = \frac{g_{max}}{2}$. If g_{new} was lower than g_{min}, then these two groups would both have to join another group right after having split, so we must have $g_{max} > 2 \times g_{min}$.

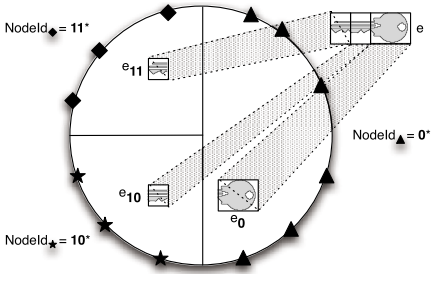


Fig. 2. Distribution of three shares e_0,e_{10} and e_{11} with $e = e_0 + e_{10} + e_{11}$. Each node knows the list of members of his sharing group.

Each share is uniquely identified by a binary *shareId* and is known by nodes which identifiers are such that $nodeId = shareId*$ in binary form (i.e., *shareId* is a binary prefix of *nodeId*). Each node knows only one share and nobody knows S entirely. Splitting a group knowing e_i (resp. merging two groups knowing e_{i0} and e_{i1}) creates two groups knowing respectively e_{i0} and e_{i1} (resp. one group knowing e_i).

In Figure 2, there are three shares e_0, e_{10} and e_{11}. Nodes which identifiers begin with 0 know e_0, those with 10 know e_{10} and those with 11 know e_{11}. To get some data signed with S, one must obtain and multiply this data signed with e_0, e_{10} and e_{11}, and thus need the cooperation of three nodes which identifiers begin respectively with 0, 10 and 11.

3.3 Distributed Certification Process

We first present an efficient certification algorithm and we then calculate the probability of success of this algorithm in presence of attackers.

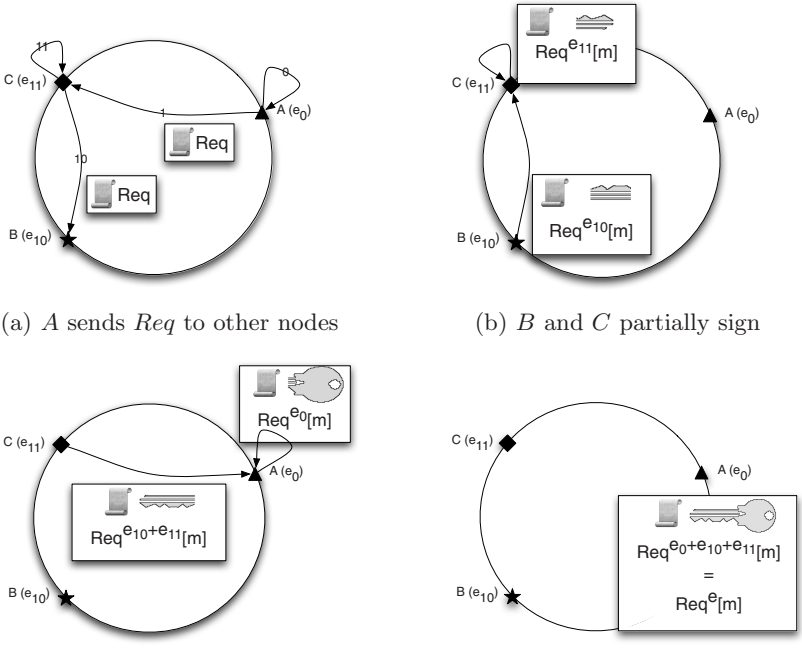
Certification Algorithm. Given the sharing illustrated in Figure 2, we now explain the associated distributed certification process. This explanation is illustrated in Figure 3, on which the node A wants to obtain a certificate for a request *Req*. It has to obtain the cooperation of one node of each sharing group to obtain the signature $Req^e[m]$. This certification is realized in two steps.

The first step is to deploy a covering tree on the sharing groups. The node A requests two nodes whose binary identifiers begin respectively with 0 and 1: A is itself eligible for 0 and finds C to handle the *shareId* 1. Since A owns the

share e_0, A stops here; C does not handle the share e_1 (which does not exist) so it forwards the request to two nodes B and C whose binary identifiers begin respectively with 10 and 11. B (resp. C) owns e_{10} (resp. e_{11}) so both nodes stop here (Figure 3(a)).

The second step is to create the certificate. B and C partially sign with e_{10} and e_{11} and send their results $Req^{e_{10}}[m]$ and $Req^{e_{11}}[m]$ to C (Figure 3(b)). C multiplies these two partial signatures, obtains $Req^{e_1}[m] = Req^{e_{10}+e_{11}}[m]$ and sends it back to A. A partially signs with e_0 to obtain $Req^{e_0}[m]$ (Figure 3(c)) and finally multiplies the partial signatures with e_0 and e_1, obtaining $Req^{e}[m]$ which is the signature of Req with $S = (e, m)$ (Figure 3(d)).

Since the public key corresponding to each share is unknown and partial signatures are thus unverifiable, each node involved in the certification can corrupt the signature. Misbehaving nodes can produce wrong exponentiations with owned share or wrong multiplications. We propose to ask $nbAsks$ nodes instead of one for a partial signature, following the hypothesis that there are less misbehaving nodes than well behaving ones. Asking several nodes, one can decide and return the most likely partial signature.



(a) A sends Req to other nodes

(b) B and C partially sign

(c) C multiplies the two partial signatures and A partially signs

(d) A obtains the signature of Req with $S = (e, m)$

Fig. 3. Certification of Req with $S = (e, m)$. Shares do not transit.

Probability of Success of the Certification Algorithm. We calculate here the probability of success of a legitimate certification in the case where every honest node accepts to participate in the certification and every attacker tries to corrupt this process. Let $k \in [0, 1]$ be the rate of attackers in the network. Each request for a partial signature is sent to $nbAsks$ nodes, $nbAsks$ odd, and the used result is the value returned by more than half of the nodes (we suppose the worst case where attackers collude to return the same incorrect partial signature).

Let us consider the tree representing the recursive calls to partially sign with a given $shareId$. The leafs of this tree correspond to the shares of the secret key which identifiers are prefixed by the given $shareId$. The internal nodes of this tree correspond to the share identifiers that are not present in the network and which provoke two recursive calls with longer identifiers, so this tree is binary. Each node of this tree coincide with one or several P2P members (if $nbAsks > 1$) asked for the corresponding $shareId$. We define the probability of success of a partial signature recursively on the height of the tree representing its recursive calls: $P(h)$ is the probability for a partial signature to succeed in a tree of height h (note that it includes the probability that the root node is honest).

If $h = 0$ (leaf), the certification succeeds if and only if this node is honest: $P(0) = 1 - k$. If $h > 0$, the certification succeeds if and only if the root node is honest and the two recursive calls succeed. Since the root node is eligible for one of the calls, the probability of this node being honest and this first call succeeding is $P(h-1)$. The second call is sent to $nbAsks$ nodes and this partial certification succeeds if the majority of calls succeed. A call succeeds with the probability $P(h-1)$ and fails with $1 - P(h-1)$: i calls thus succeed with the probability $C^i_{nbAsks} P(h-1)^i (1-P(h-1))^{nbAsks-i}$. This second partial signature succeeds when more than half of the requests succeed, yielding a probability of $\sum_{i=\frac{nbAsks+1}{2}}^{nbAsks} C^i_{nbAsks} P(h-1)^i (1-P(h-1))^{nbAsks-i}$. Putting it all together gives

$$P(h) = P(h-1) \times \sum_{i=\frac{nbAsks+1}{2}}^{nbAsks} C^i_{nbAsks} P(h-1)^i (1-P(h-1))^{nbAsks-i}$$

To obtain the complete signature, the root of the tree is called on the empty share identifier and the leafs correspond thus to the s shares of the secret key present in the network. Since this tree is binary, its height is $h = log_2(s)$. If n is the number of nodes and g_{min} (resp. g_{max}) is the minimal (resp. maximal) size of a sharing group, then the average number of shares is $s = \frac{2n}{g_{min}+g_{max}}$. The probability of success of the certification is thus $P\big(log_2\big(\frac{2n}{g_{min}+g_{max}}\big)\big)$. We can conclude that the bigger the network is, the harder the certifications are. This probability is graphed in Section 5.

4 Security Analysis

In this section, we discuss the robustness of our system against attackers.

4.1 Obtaining a Fake Certificate

How to Obtain a Fake Certificate ? An attacker can create a fake certificate through obtaining the secret key of the network S. Since S is initially generated through a distributed algorithm, no member knows S at any moment. So, the attacker has to obtain every share to rebuild S, which means corrupting a node or inserting himself or an accomplice in every sharing group. Due to the large number of sharing groups (more than $\frac{n}{g_{max}} = \frac{n}{40}$ with the values proposed in Section 5), we think that corrupting a node in each group is quite hard. Inserting himself in each sharing group involves creating multiple identities which is a sybil attack (we made the assumption there is no such attack in Section 3.1). The only other possibility is a group of attackers getting every share and we thus calculate the probability of such a successful attack.

Probability for Colluding Attackers to get Every Share. Let k be the ratio of attackers. If a given sharing group is composed of g_i members, then the probability that there is no attacker in this group is $(1-k)^{g_i}$. The probability that there is at least one attacker in this group is thus $1-(1-k)^{g_i}$.

Consider now a network composed of s sharing groups of resp. $g_1, \ldots, g_s$ members (hence the number of nodes is $\sum_{i=1}^{s} g_i$). Then the probability that there is an attacker in each sharing group is $\prod_{i=1}^{s} 1-(1-k)^{g_i}$. Since $1-(1-k)^{g_i} < 1$, this probability decreases when the number of shares s increases, i.e., when the network grows. Moreover, for a given network size, reducing the size of the groups g_i and hence increasing the number of shares s results in a lower probability of such an attack. This probability is graphed in Section 5.

4.2 Attacking an Honest Certification

How to Attack an Honest Certification ? An attacker can prevent a certification process through intercepting the certification request. This attack is prevented by requesting several nodes instead of one. An attacker can also try to make a share unavailable, through crashing all the nodes in a given sharing group or creating enough nodes or having enough accomplices to control all the nodes in such a group. We exclude here the case of crashing $g_{min} = 20$ nodes (value proposed in Section 5) and leave a potential countermeasure to future work. Creating enough nodes to control a full sharing group is a sybil attack, which is not handled here as stated in Section 3.1. The only other case is a group of attackers controlling all the nodes in a sharing group, which is discussed below.

Probability for Colluding Attackers to Control all the Nodes in a Sharing Group. Let k be the ratio of attackers. If a given sharing group is composed of g_i members, then the probability that there is only attackers in this group is k^{g_i}. The probability that there is at least one well-behaving node is thus $1-k^{g_i}$.

Consider now a network composed of s sharing groups of resp. $g_1, \ldots, g_s$ members. Then the probability that there is at least one well-behaving node in each sharing group is $\prod_{i=1}^{s} 1-k^{g_i}$. The probability that there is one group

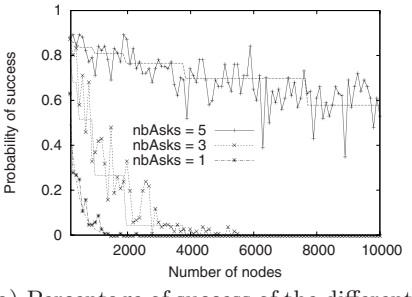
containing only attackers is thus the complementary probability $1-\prod_{i=1}^{s} 1-k^{g_i}$. Since $1-k^{g_i} < 1$, this probability increases when the network grows. Moreover, for a given network size, reducing the size of the groups g_i and hence increasing the number of shares s results in a higher probability of such an attack. Given the probability presented in 4.1, the size of the groups implies thus a trade-off between these two possible attacks. This probability is graphed in Section 5.

5 Experimental Results

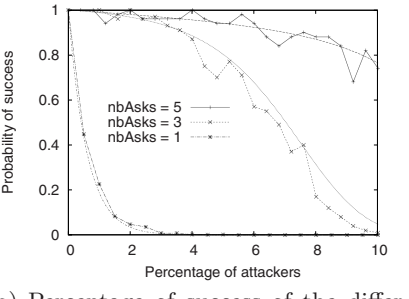
In this section, we present simulations results and compare them to theoretical results provided in Sections 3.3 and 4. We simulated our distributed certification system using *PeerSim* [10], an extensible P2P simulator. Parameters of our simulations are the number of nodes in the network, the percentage of attackers and the number of nodes *nbAsks* asked for each partial signature. Sharing groups are composed of $g_{min} = 20$ to $g_{max} = 40$ members, yielding $0.025 < t < 0.05$ (certification is only possible through the collaboration of 2.5 to 5% of the nodes).

In these simulations, we do not handle attacks on nodes. For instance, a *worm* attack could allow some attacker to take control of a large number of honest nodes. However, we think that protecting against such attacks is out of our current scope. Considered nodes are thus well-behaving or misbehaving depending only on the local user choice.

In Figure 4(a), we vary the number of nodes in the network with a constant percentage of attackers (10%, which is already a very large part since we do not consider worm attacks). As stated in Section 3.3, the larger the network is, the harder it is to achieve a successful certification. With 500 nodes, 20%



(a) Percentage of success of the different algorithms in function of the number of nodes. Each experiment contains 10% of misbehaving nodes and each node asks 1, 3 or 5 other nodes for partial signatures. Corresponding theoretical curves are also drawn.



(b) Percentage of success of the different algorithms in function of the percentage of attackers. Each experiment contains 5000 nodes and each node asks 1, 3 or 5 other nodes for partial signatures. Corresponding theoretical curves are also drawn.

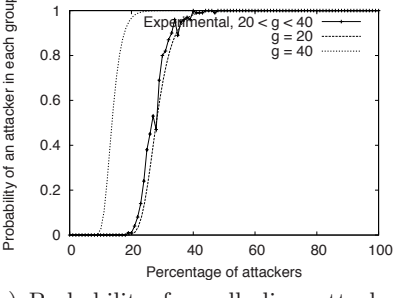
Fig. 4. Certification results

of the certifications with $nbAsks = 1$ succeed. With more than 1500 nodes, no certification with $nbAsks = 1$ succeeds. Even if certification with $nbAsks = 5$ allows to handle more attackers, it is clear that it will be important to exclude attackers to maintain an efficient certification service.

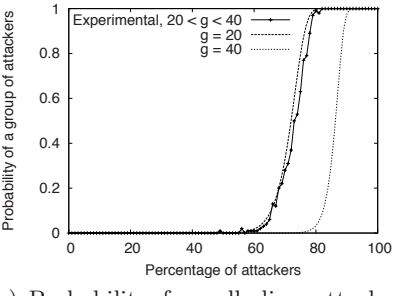
In Figure 4(b), we vary the percentage of misbehaving nodes with a constant total number of nodes (5000, which is a median value of Figure 4(a)). With 1% of attackers, only 20% of certifications with $nbAsks = 1$ succeed. With more than 2% of attackers, this algorithm is not usable with $nbAsks = 1$. Certifications with $nbAsks = 3$ or $nbAsks = 5$ are able to tolerate much more attackers.

Figure 5(a) shows the probability for colluding attackers to obtain every share of the secret key in function of the percentage of attackers. The experimental curve with sharing groups composed of $g_{min} = 20$ to $g_{max} = 40$ members is bounded by the two theoretical curves where all groups are composed of 20 (resp. 40) nodes. In the worst case, 2.5% of the nodes ($\frac{1}{g_{max}} = 0.025$) should be able to obtain every share. However, obtaining every share requires not only to have more attackers than shares but also to specifically have an attacker in *each* sharing group. The theoretical probability of such an attack is in fact infinitesimal for less than 10% of attackers (theoretical curve with $g = 40$). Moreover, the experimental curve is much closer to the theoretical one with $g = 20$ than to the one with $g = 40$. This is due to the fact that, in the experiment, there are groups of different sizes between 20 and 40. It is then less probable for an attacker to get into those containing only 20 members: these small groups have a critical impact on this probability. Given that, the effective probability of such an attack is infinitesimal for less than 20% of attackers.

Figure 5(b) shows the probability for colluding attackers to be the only members of a sharing group and thus to be able to make this share unavailable. The experimental curve is also bounded by the two theoretical curves where



(a) Probability for colluding attackers to obtain every share of the secret key in function of the percentage of attackers. Corresponding theoretical curves are also drawn.



(b) Probability for colluding attackers to be the only members of a sharing group in function of the percentage of attackers. Corresponding theoretical curves are also drawn.

Fig. 5. Robustness results in a 10,000 nodes network

all groups are composed of 20 (resp. 40) nodes. It appears that the probability of such an attack is infinitesimal for less than 60% of attackers. For the same reasons as in the previous figure, the experimental curve is much closer to the theoretical one with $g = 20$ than to the one with $g = 40$.

6 Applications of Distributed Certification

In this section, we briefly introduce three applications of our distributed certification. These applications aim at preventing sybil attacks, excluding misbehaving nodes and finally providing a secure naming service.

6.1 Sybil Protection through Admission Control to the Network

In the sybil attack [5], an attacker creates many node identifiers and possibly picks a specific subset. With many node identifiers, an attacker can alter the overall performance of the network. Moreover, even with a few well-chosen identifiers, an attacker can isolate nodes or censor resources. First, he can isolate a victim node from the network and filter his requests by choosing precisely the node identifiers the victim node uses in its routing table: this victim node then sends all his requests to this attacker. Second, an attacker can take the control of a resource and of all its replicas by choosing identifiers close to the attacked resource identifier (if replication is done on nearby nodes). The problem is thus not only to limit the number of identifiers a user can create but also to enforce truly random identifiers.

Relying on friendship relations, *SybilGuard* [11] allows each node to decide whether another node is genuine or not and so limits the number of sybil nodes an attacker can create. However, the identifier of a node is the hash of its public key. An attacker can thus generate many key pairs and choose a specific one which hashes to an identifier in the desired part of the overlay: *SybilGuard* does not enforce random identifiers. We propose thus to combine our distributed certification with *SybilGuard*. To join the network, a new member must obtain a certificate containing his public key signed with the network secret key and thus needs the cooperation of a fixed ratio t of the nodes. Each of these nodes tests the new node with *SybilGuard* and cooperates only if the new node is detected as genuine: if the newcomer is detected as sybil, he does not obtain a certificate which prevents an attacker from creating many identifiers. Then, node identifiers are derived from the unpredictable signatures of the certificates. Each new member thus generates a key pair, registers his public key, and finally obtains his node identifier: a user cannot predict his identifier which ensures random ones. Accepted members are checked as non-sybil and the network is self-protected from sybil attacks.

6.2 Detection and Exclusion of Misbehaving Nodes

To prevent some adversarial behaviors in the P2P network, it is interesting to detect and exclude misbehaving nodes. For instance, in our certification algorithm,

nodes ask several other nodes for the same partial signature and compare the results to cope with attackers returning fake partial signatures. Also, to obtain a resource, nodes may use redundant routing to prevent an attacker present on a route to this resource from forging a fake response. However, asking several nodes for a partial signature or using redundant routing generates overhead on the network. Excluding such attackers allows to reduce the number of nodes asked or redundant routes used for an identical success probability.

We propose thus to detect and exclude some types of misbehaving nodes. We make the assumption that the majority of the nodes are honest and we thus detect attackers which exhibit a minority behavior. Each node monitors some traffic and compares messages which should be the same. For instance, a minority partial signature or a minority response for a given resource requested through different routes is considered as an attack. If some traffic reveals an attack, then the victim sends the messages proving this attack to a ratio t of the nodes and these nodes revoke the membership certificate of the attacker with the network secret key (revocation is a special case of certification). This attacker is then globally excluded from the network.

6.3 Secure Naming of Resources

In [12], Bryan *et al.* propose to use a cheap and highly available P2P network as a VoIP directory. Each user inserts an entry in the P2P network mapping his name such as "John Smith" to his IP address. When a user wants to phone John Smith, he requests the resource identified by $h(\text{"}John\ Smith\text{"})$ and obtains the IP address of John Smith. However, they do not propose any mechanism to prevent an attacker from intersecting such a request and replying a fake IP address. The only solution, to our best knowledge, is to call John Smith by a hash of his public key rather than by his name. This is not convenient to remember.

We propose that each new user obtains a certificate binding his user name to his public key, using distributed certification. Each node involved in this certification first checks if there is already a user with the same name by requesting this username in the DHT. Cache mechanisms of DHT should be able to manage this peak of identical requests. If and only if the name is free, then this node proceeds with the distributed certification: a certificate for a given name can only be obtained if this name is free. Then, a node wanting to phone "John Smith" requests the resource identified by $h(\text{"}John\ Smith\text{"})$, obtains the certified public key of John Smith and his IP address, and can then challenge John Smith about his private key to authenticate him.

7 Conclusion and Future Work

We proposed here a distributed certification system for structured P2P networks. This mechanism provides the ability to leverage the local knowledge of a ratio t of the nodes to a global knowledge recognized by all the nodes in the network. Subsequent verifications of the certificates are simple checks on digital signatures.

We evaluated and validated this distributed certification in the presence of attackers through probabilities and simulations. We now plan on stressing a real implementation of our system under dynamics using the PlanetLab testbed.

We finally briefly presented three applications of distributed certification in structured P2P networks. These applications involve controlling the access of new nodes, excluding misbehaving nodes and providing a secure naming service. We now have to precise these applications and study interactions of the different steps.

Acknowledgments. The authors thank Orange Labs for partially funding this work and especially Hervé Debar for his involvement in the project and his advice and valuable comments.

References

1. Clip2: The gnutella protocol specification v0.4 (2000), `http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf`
2. Stoica, I., Morris, R., Karger, D.R., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the ACM SIGCOMM Conference (SIGCOMM). Computer Communication Review, pp. 149–160. ACM Press, New York (2001)
3. Kong, J., Zerfos, P., Luo, H., Lu, S., Zhang, L.: Providing robust and ubiquitous security support for mobile ad hoc networks. In: Proceedings of the 9th IEEE International Conference on Network Protocols (ICNP). IEEE Computer Society, Los Alamitos (2001)
4. Saxena, N., Tsudik, G., Yi, J.H.: Experimenting with admission control in P2P. In: Proceedings of the International Workshop on Advanced Developments in System and Software Security (WADIS) (2003)
5. Douceur, J.R.: The sybil attack. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002)
6. Shamir, A.: How to share a secret. Communications of the ACM 22(11) (1979)
7. Desmedt, Y.: Some recent research aspects of threshold cryptography. In: Okamoto, E. (ed.) ISW 1997. LNCS, vol. 1396, pp. 158–173. Springer, Heidelberg (1998)
8. Frankel, Y., Gemmell, P., MacKenzie, P.D., Yung, M.: Optimal-resilience proactive public-key cryptosystems. In: Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science (FOCS). IEEE Computer Society, Los Alamitos (1997)
9. Boneh, Franklin,: Efficient Generation of Shared RSA Keys. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 425–439. Springer, Heidelberg (1997)
10. Jelasity, M., Jesi, G.P., Montresor, A., Voulgaris, S.: PeerSim P2P Simulator (2004), `http://peersim.sourceforge.net/`
11. Yu, H., Kaminsky, M., Gibbons, P.B., Flaxman, A.: Sybilguard: Defending against sybil attacks via social networks. In: Proceedings of the ACM SIGCOMM Conference (SIGCOMM), pp. 267–278. ACM Press, New York (2006)
12. Bryan, D.A., Lowekamp, B.B., Jennings, C.: SOSIMPLE: A serverless, standards-based, P2P SIP communication system. In: Proceedings of the International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (AAA-IDEA) (2005)