A Framework Toward a Self-Organizing and Self-Healing Certificate Authority Group in a Content Addressable Network

Anuchart Tassanaviboon
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada
Email: atassana@uwaterloo.ca

Guang Gong
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada
Email: G.Gong@ece.uwaterloo.ca

***Abstract*—Public-key provision in on Internet scale is crucial for securing peer-to-peer (P2P) applications. This paper proposes a framework for a self-organizing and self-healing certificate authority (CA) in a Content Addressable Network (CAN) that can provide certificates without a centralized Trusted Third Party (TTP). In our framework, a CA group is initialized by bootstrapping nodes and then grows to a mature state by itself. Based on our group management policies, the membership in the CA group is dynamic and has a uniform distribution over the P2P community. Meanwhile, the honest majority of the CA group is maintained by a Byzantine agreement algorithm, and all shares of the CA group are refreshed gradually and continuously. A security analysis shows that the framework enables key registration and certificate issue with resistance to man-in-the-middle (MITM), collusion, and node impersonation attacks.**

I. Introduction

In a peer-to-peer (P2P) application model, each node is equivalent in term of functionality (i.e., can simultaneously be both a client and a server) and trustworthiness (i.e., has neither a security hierarchy nor a centralized TTP). Meanwhile, P2P applications are expected to gain reliability, efficiency, and performance from resources sharing among nodes in the same application community. The trust equality among nodes hinders security provision of authentication, integrity, non-repudiation, etc. This situation necessitates a self-organizing and self-healing security system without a centralized TTP for P2P applications.

Recent literature has proposed many self-organizing public-key based systems to secure P2P applications. Takeda *et al.* [13] suggested a distributed key management scheme that employs a Web of Trust and a Distributed Hash Table (DHT) to retrieve a required public key by iteratively searching authentic nodes. Pathak and Iftode [10] introduced a public-key authentication protocol in which each node maintains an individual trusted group to perform authentication for different end-points, and the honest majority of the trusted group is controlled by a Byzantine agreement algorithm. Avramidis *et al.* [1] offered a Public-Key Infrastructure (PKI) that applies a (t, n) threshold signature scheme to the Chord overlay network. Lesueur *et al.,* [7] [8] proposed a distributed certification system in which t percentage of nodes is required to sign a certificate collaboratively, and the ratio is maintained by increasing a number of shares according to the number of nodes.

This paper proposes a new framework for the self-organizing and self-healing CA group in CAN. Our framework leverages a CAN overlay network with overloading zones and an (n, n) threshold signature scheme to construct a CA group and addresses the lack of fault tolerance in an (n, n) threshold signature scheme by exploiting the redundancy of overloading zones. In a P2P community, the CA group is a single trusted group and issues certificates for every P2P node. To realize self-organizing and self-healing functionalities, the CA group itself maintains its CAN structure in a dynamic fashion by pre-defined group management policies, and eliminates malicious nodes from the CA group by a Byzantine Agreement (BA) protocol. Before signing a certificate, the CA group constructs another CAN overlay network (multicast group) on top of itself to optimize the communication cost of a BA protocol. The Multicast members then cooperate to prove the possession of the private key associated with the proposed public key in a certificate request and to sign the requested certificate.

The rest of the paper is organized as follows. Section II describes system and adversary models for our framework and the notations in our protocol. Section III details a CA group structure, group management policies, a bootstrapping phase, and the protocols in a running phase. Section IV presents security analysis of our framework. Finally, Section V highlights our contribution.

II. System and Adversary Models

A. Structured P2P Network Model

We consider a P2P network as an application-level overlay network over the underlying Internet without IP multicast services. The inter-communication among nodes is asynchronous, which does not guarantee message delivery, reliability, and ordering. The P2P network is composed of dynamic nodes

that unpredictably join, leave, and fail. Each node has a unique non-zero network ID mapping tightly to the overlay-network topology. The multicast and broadcast communication controlled by a CAN-based flooding algorithm [12] is not reliable and totally ordered. We assume that timed services and access to hardware clocks allow time-out and retransmission to mask low level communication failures, and provide asynchronous communication [5], which is practically implementable for distributed services.

B. Trust Model

In our trust model, a node is trustworthy if it is trusted by a dynamic trust group, termed a **CA group**, which acts as an internal TTP for a P2P community and forms the security foundation. While none of the nodes trusts others, each node believes that the majority of nodes in the trusted group are honest. Since a BA protocol is used to detect malicious nodes in a CA group, the CA group of size n is expected to have an honest majority (i.e., at most $\lfloor \frac{n-1}{3} \rfloor$ malicious nodes exist).

C. Bootstrapping Model

Each bootstrapping (BT) node has already registered a public key with an external TTP. Thus, it can use cryptographic functions for confidentiality, integrity, and authentication. We assume that all BT nodes can protect their private keys well, execute algorithms correctly, and run cryptographic functions to establish secure and authentic communications among the BT nodes. Moreover, the number of malicious BT nodes is not large enough to reveal any secret information. For instance, the RSA-key generator, proposed by Boneh and Franklin [4], can prevent collusion attacks from $\lfloor \frac{n-1}{2} \rfloor$ out of n BT nodes to factor the modulus N or to compromise the private key.

D. Adversary Model

An adversary may either omit to do what it is supposed to do or behave arbitrarily. More specifically, an adversary can eavesdrop on, drop, forge or intercept messages. Here, interception means launching a man-in-the-middle (MITM) attack. Moreover, we assume that an adversary has limited computational power to break cryptographic functions, but he/she can send any messages at any time. We also assume that two kinds of adversaries exist in the system: *1) external:* adversaries who participate in a user overlay network but are not members of a CA group; and *2) internal:* adversaries who are members of both a user overlay network and a CA group.

E. Attack Model

Our intention is to devise a CA group for structured P2P networks. Thus, we focus on two main attacks to the CA group:

1) Node impersonation: An external adversary A' tries to convince a CA group that its public key $PK_{A'}$ belongs to an honest node A. We assume that each P2P node will be reached by different CA nodes through distinct communication paths. Therefore, if adversaries intercept traffic from the CA group on more than a fraction α of paths, then the adversaries can impersonate other nodes.

TABLE I
NOTATION FOR PROTOCOL DESCRIPTIONS

Notation	Description
r_X	A pseudo random number generated by entity X
SK_X	Private(Secret) key of entity X
PK_X	Public key of entity X
SK_X^i	The i^{th} share associated with X's private key
PK_X^i	The i^{th} share associated with X's public key
$U_1, U_2, ..., U_n$	A string generated from a concatenation of strings $U_1 \| \| U_2 \| \| ... \| \| U_n$
$E_{PK_X}(U)$	The cypher text of a string U encrypted with X's public key
$D_{SK_X}(U)$	The plain text of a string U decrypted with X's private key
$Sig_{SK_X}(U)$	A digital signature over a string U generated with X's private key
$[U]_{SK_X}$	A string U attached by digital signature $Sig_{SK_X}(U)$
$CertReq_X$	A certificate request is a binding between X's identifier and public-key, signed with its own private key, i.e., $CertReq_X = [X, PK_X]_{SK_X}$
$Cert_X$	A public-key certificate is X's CertReq, including CA's policies, signed with CA's private key, i.e., $Cert_X = [CretReq_X, TimeExp]_{SK_{CA}}$
$Cert_X^i$	A partial certificate is X's certificate request, signed with the i^{th} share of CA, in stead of CA's private key
$\rightarrowtail$	Unicast traffic flows over the underlying network
$\rightarrow$	Unicast traffic flows over a user overlay network
$\hookrightarrow$	Unicast traffic flows over a CA overlay network
$\looparrowright$	Unicast traffic flows over a decision overlay network
$\hookrightarrow *$	Broadcast traffic flows over a CA overlay network
$\looparrowright *$	Multicast traffic flows over a decision overlay network

2) CA functionality interference: An internal adversary, who has convinced a CA group of his honesty and joined the CA group, may prevent the CA group from issuing certificates to legitimate users or make false accusations to subvert the honest majority of the CA group.

F. Notation

The notations in Table I are used for the public-key cryptography, threshold scheme, and message formats in the following protocol descriptions.

III. A FRAMEWORK OF A SELF-ORGANIZING AND SELF-HEALING CA GROUP (SOHCG)

A. A CA Group Structure

The structure of a CA group is a d-dimensional Cartesian space (CAN structure) with overloading zones [11], in which each zone of the CAN overlay network maintains a share corresponding to the CA's private key. Moreover, the CA group recruits certified nodes, termed **CA nodes**, from a user overlay network to form a CAN overlay network. To do so, we must modify the original CAN-based structure and algorithms as follows. Note that nodes sharing the same zone are termed **peers**.

- *A Combination of an (n, n) Threshold Scheme and a CAN with Overloading Zones:* Duplicating a share with multiple peers and matching a share to a zone can provide fault and key-compromise tolerances. That is, an adversary must disrupt all peers in one zone to disable a CA group or compromise every share (at least one node in each zone) to reveal the CA's private key.

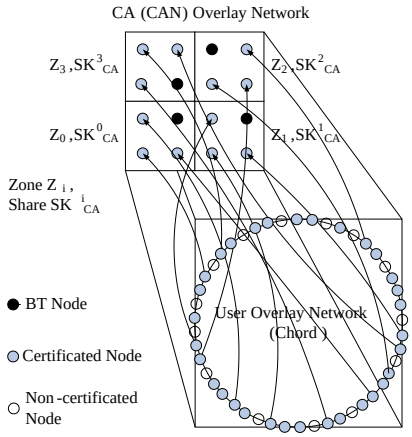

Fig. 1. Matching CA Group to 2-Dimensional CAN with Overloading

- *Flooding in a CAN with Overloading Zones (zone flooding):* Two modifications to the original flooding algorithm proposed in [12] are required. First, to provide multi-path routing, we allow CAN nodes to forward a message farther than half-way along the dimension but not to forward a message whose sequence number has already been received. Second, to prevent collusion attacks, all peers of the sender's zone contribute in generating a random CAN ID by using a distributed and synchronized scheme, such as the protocol proposed by Benaloh [2]. According to the distance in the CAN structure, the next zone's peer closest to the random ID is selected as a designated node for each flooding step. Therefore, in each step and so on, a sender cannot intentionally forwards messages to its conspirators. Here, for broadcasting and multicasting, the original flooding is used for non-overloading zones; meanwhile, the zone flooding is used for overloading zones.

We now involve two categories of P2P networks: one used for constructing a CA group is termed a **CA overlay network**, and one run by users can be any structured P2P overlay networks and is termed a **user overlay network**. Hence, a CA node participating in both overlay networks must run the protocol stacks of both CA and user overlay networks. For the sake of simplicity and without loss of generality, this paper assumes that the CA overlay network is constructed by a 2-dimensional CAN network, and the user overlay network is a Chord network. Figure 1 shows the related constructions of both CA and user overlay networks.

B. Group Management Policies

Since our proof of private-key possession is based on the assumption that paths from different CA nodes to a P2P node are distinct, CA nodes must be selected from a user overlay network uniformly. Moreover, to mitigate collusion and Sybil attacks [3], CA nodes' membership must be dynamic in order to limit the time that a malicious CA node can subvert the functionality and honest majority of a CA group, i.e., age rounds of certificate issuing. To achieve the above two properties, we define the following two policies.

1) Peer Recruitment: When the number of peers in one zone is less than p_{min}, all peers in that zone cooperate to generate a random ID with the same distributed scheme employed in zone flooding (See Subsection III-A). The peer closest to the random ID is selected as a coordinator. Thus, all peers are equally likely to be selected, instead of a fixed coordinator per zone. Based on a look-up service in the user overlay network, the coordinator selects the user node associated with the random ID. The coordinator then verifies whether the selected node is a certified node, not a current CA node or a revoked node, before inviting this node to be a new CA node. If the verification fails, another random ID will be generated until peers can find a satisfactory node. In the joining step, the coordinator hands over the share to the new CA node via a secure and authentic channel established by the public-key scheme. As a result, every peer has control of and participation in peer recruitment.

2) Peer Retirement: In a CAN with overloading zones, every CAN node (or CA node) maintains a peer list and a neighbor list (i.e., a list of peers in its adjacent zones). Usually an entry of these two lists is composed of a node ID and its IP address. We add a *Cert* field (i.e., the certificate of that node) and a *CertCount* field (i.e., a counter of certificates issued by that node) to the peer list. Additionally, we add only a *Cert* field to the neighbor list. When a CA node P is selected to be a member of a decision group, every peer of P queries P's entry in its peer list and decreases the value of P's *CertCount* field. If the *CertCount* field is greater than zero, P is selected. Otherwise P's peers consider P as an **expiry node**, remove P from their peer lists, and tell P's neighbors to remove P from their neighbor lists as well. Hence, P is retired from the CA group. Moreover, in a CAN overlay network, every node must periodically send refresh messages to its neighbors and peers. If P's peers or neighbors do not receive the refresh messages within a limited time, they consider P an **unliveness node** and remove P from their peer lists or neighbor lists, respectively.

After a user overlay network constructs a CA group for certificate provision, we can classify a node according to its functionalities (i.e., a user node, a CA node, or a decision node), behavior (i.e., malicious or non-malicious), and certification status (i.e., certified or non-certified), as shown in Figure 2. Nodes in each group may overlap those in other groups and migrate from one group to others according to their certification procedures (i.e., certificate issuing or revocation) and the group management policies (i.e., retirement or recruitment).

Although our peer retirement and peer recruitment policies cause nodes to continuously join and leave the CA group (or the CAN overlay network) at a high churning rate, a CAN overlay network can still operate well because only $2d$ zones are affected by the joining or leaving of one node (where d is the dimension of a CAN overlay network). Before a CA group is formed and maintained by a coalition of certified nodes, the CA group must be initialized by BT nodes in the bootstrapping phase, as described in the next subsection.

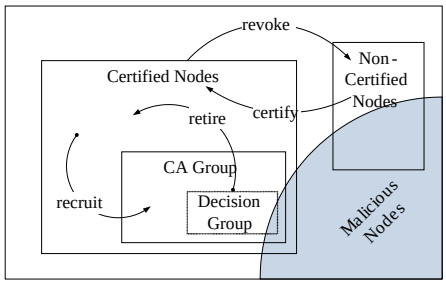

Fig. 2. Groups Relation and Management

C. Bootstrapping Phase

At the beginning of this phase, the authority creates BT nodes to form a CA group and to function as shareholders in a threshold signature scheme. Moreover, although a large number of peers and zones in a CA group can increase fault tolerance and attack resistance, increasing CA group's size achieves trades efficiency for security. In other words, if the number of peers in one zone increases, not only does fault tolerance increase but also the number of states (neighbors and peers lists). Meanwhile, if the number of zones (shares) increases, not only does the key-compromised tolerance increase but also the computing and communication cost. To craft our scheme for a tradeoff between efficiency and security, we define the following parameters to control CA group size.

- p_{min}: the minimum number of peers before recruitment.
- p_{max}: the maximum number of peers in the mature state, where $p_{max} \geq 2p_{min} + 2$.
- z_{min}: the minimum number of zones at the initial state.
- z_{max}: the maximum number of zones at the mature state, where $z_{max} \geq 2z_{min}$.
- age: the maximum number of certificates that each node can issue before retirement.

To initialize an (n, n) threshold signature scheme without a dealer and knowledge about the corresponding private key and to establish a CA group according to the predefined parameters, the bootstrapping phase consists of the following two steps.

1) Zone and Share Initialization: The authority constructs a 2-dimensional CAN with z_{min} zones and at least z_{max} BT nodes. Hence, an initial CA group, which starts with z_{min} zones (at least two BT nodes per zone) and a (z_{min}, z_{min}) threshold scheme, has a suitable and practical size for a start-up network. Then z_{min} BT nodes from different zones use Boneh and Franklin's algorithm [4] to select a public key PK_{CA} and to generate a modular N and shares SK^i for each zone. Both the modular N and the public key PK_{CA} are published on public servers or embedded in software distributions, as happens in many operating systems and web browsers. Additionally, the IP addresses of the BT nodes are published under the CA domain name or in the extension field of the CA certificate [6]. The BT nodes then start issuing certificates for P2P users and managing the CA group.

2) CA Group Initialization: Meanwhile, BT nodes start recruiting certified nodes to the CA group and hand over the shares and parameters through secure and authentic channels established by a public-key scheme. When a zone consists of p_{min} peers, the BT nodes in that zone delete their shares and fade out from the threshold signature scheme forever. However, BT nodes are still permanent members of the CA group. Note that BT nodes maintain CA policies and current peer-lists for certificate requests.

Now the CA zone itself can grow up without BT nodes. Each zone continues recruiting until the number of peers amounts to the upper bound p_{max}. Then the zone is split into two halves, each of which contains half of its peers and BT nodes if applicable. To limit the number of zones to z_{max}, the coordinator of a split zone broadcasts a *Zone Count* message to the CA group to get zone information from every zone before splitting. The coordinator then checks the completeness of the zone combinations in the Cartesian space before counting the number of zones. If the current number of zones is less than z_{max}, the recruiting and splitting are repeated until the CA group consists of z_{max} zones and p_{max} peers in every zone. For example, if a zone Z_0 has to be split into two new zones Z_{00} and Z_{01}, all peers in zone Z_0 cooperate to select a nonce r in the manner used in zone flooding (See Subsection III-A). Next, zone Z_0 is split into two new zones Z_{00}, Z_{01} with the new shares $SK_{CA}^{Z_{00}} \equiv r \pmod N$ and $SK_{CA}^{Z_{01}} \equiv SK_{CA}^{Z_0} - r \pmod N$, respectively. Finally, zones Z_{00} and Z_{01} render their new shares with their neighbor zones. The details of the *Share Rendering Protocol* is presented in Subsection III-G.

Eventually, the CA group grows to a mature state with a proper size for achieving a suitable efficiency/security tradeoff. The following subsections explain how the mature CA group can achieve two main functionalities of a traditional CA, i.e., registration and certificate issuing.

D. Key-Registration Protocol

When a new node joins a user overlay network, it must request its certificate from a CA group to upgrade from an uncertified node to a certified one. To this end, a CA group first prepares its delegation, called a **decision group**, when receiving a certificate request. The decision group performs the later protocols and multicast efficiently and employs a challenge-response protocol to prove the private-key possession before accepting the corresponding public key. Table II shows the *Key-Registration Protocol*, which consists of the following four steps.

1) Certificate Request: When node A needs a new certificate, A first generates a private/public key pair by itself and constructs a certificate request $CertReq_A$ under the PKCS 10 [9] format. A then submits the $CertReq_A$ to one of the BT nodes, which does not issue the certificate by itself but forms a *Forward Request (FR)* message by attaching CA policies (e.g., expiry time, etc.) to the $CertReq_A$ before randomly forwarding the message to one of the CA nodes.

2) Decision Group Establishment: The first selected node P_0 (assume that P_0 is in zone 0) combines its node ID and certificate with the *FR* message to form a *Group Request*

TABLE II
KEY REGISTRATION PROTOCOL (P_i WHERE $i = 0, ..., n$)

Step 1. Certificate Request

A:	Generates $CertReq_A = [A, PK_A]_{SK_A}$
$A \rightarrowtail BT$:	$[A, BT, Cert\ Request, CertReq_A]_{SK_A}$
$BT \hookrightarrow P_0$:	$FR = [BT, P_0, Cert\ Request, CertReq_A, ExpTime]_{SK_{BT}}$

Step 2. Decision Group Establishment

$P_0 \hookrightarrow *$:	$[P_0, *, Group\ Request, FR, P_0, Cert_{P_0}]_{SK_{P_0}}$
P_i:	Records FR
$P_i \hookrightarrow P_0$:	$[P_i, P_0, Group\ Join, P_i, ZoneInf_i, Cert_{P_i}]_{SK_{P_i}}$
P_0:	Checks the completeness of zone area
P_i:	Form a mini-CAN
$P_0 \hookrightarrow *$:	$[P_0, *, Group\ Confirmation, P_0, Cert_{P_0}, ..., P_n, Cert_{P_n}]_{SK_{P_0}}$

Step 3. Challenge-Response

P_i:	Generates a nonce r_{P_i} and a challenge $C_{P_i} = E_{PK_A}(r_{P_i})$
$P_i \rightarrow A$:	$C_{P_i,A} = [P_i, A, Challenge, C_{P_i}, Cert_{P_i}]_{SK_{P_i}}$
A:	Generates a response $R_{P_i} = D_{SK_A}(C_{P_i})$
$A \rightarrow P_i$:	$R_{P_i,A} = [A, P_i, Response, R_{P_i}]_{SK_A}$
A:	$V_A[i] = (C_{P_i,A}, R_{P_i,A})$

Step 4. (C, R) **Agreement**

$P_i \hookrightarrow *$:	$[P_i, *, (C, R)\ Distribute, (C_{P_i,A}, R_{P_i,A})]_{SK_{P_i}}$
P_i:	$V_{P_i}[j] = (C_{P_j,A}, R_{P_j,A})$
	For j from 0 to n
	$\overrightarrow{V}[j] = val(\overrightarrow{V_{P_i}}[j])$
P_i:	If $threshold(\overrightarrow{V}) = true$
	Runs *Certification Issue Protocol*
	Else
	$P_i \hookrightarrow P_0$: $[P_i, P_0, Authen\ Fault]_{SK_{P_i}}$
	Runs *Malicious Node Detection Protocol*

message and then broadcasts the *Group Request* message to a CA group. Hence, all nodes selected through the zone flooding algorithm are decision nodes. Each responds to P_0 with a *Group Join* message consisting of its zone information and certificate, and stores the *FR* message for generating a partial certificate. Before forming a decision group, which is a CAN-based multicast group called a **mini-CAN**, P_0 must check the completeness of the Cartesian space by combining all zones' information from every *Group Join* message. Therefore, the number of nodes (or zones) in a decision group is equal to the number of zones in a CA group. In other words, the decision group has all shares for reconstructing a complete certificate, as shown in Figure 3. After P_0 learns the IDs and certificates of all decision nodes from the *Group Join* messages, it multicasts all the information to the decision group with a *Group Confirmation* message. Thus, each decision node knows the others and is ready to prove A's private-key possession.

3) Challenge-Response: Each decision node P_i generates a nonce r_{P_i} and a challenge C_{P_i}, which is the nonce encrypted with A's public key, and then sends C_{P_i} to A. A decrypts the challenge using its private key and sends the response R_{P_i} to P_i. A also stores these challenge-response (C, R) pairs in a vector $\overrightarrow{V_A}$ for the *Malicious Node Detection Protocol*, which is introduced in Subsection III-F.

4) (C, R) *Agreement:* The proof of private-key possession is based on an agreement on the validity of (C, R) pairs from every decision node. This agreement can be reached in three

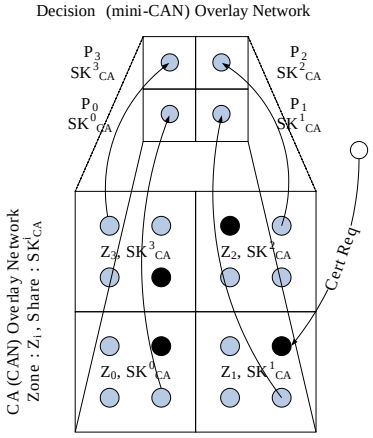

Fig. 3. Forming a Decision Group over a CA Group

levels: **consensus**, **threshold**, and **none**. The values of the validity and the agreement are defined below.

- *The Validity of a* (C, R) *Pair:* Denoted by $val(C, R)$, if $C_{P_i} = E_{PK_A}(R_{P_i})$, then $val(C, R) = ture$, otherwise $val(C, R) = false$.
- *The Agreement on a Decision Vector* $\overrightarrow{V}$*:* The consensus function of $\overrightarrow{V}$, denoted by $consensus(\overrightarrow{V})$, and the threshold function of $\overrightarrow{V}$, denoted by $threshold(\overrightarrow{V})$, are defined as follows.
  - If all $V[i]$ have the same value v, then $consensus(\overrightarrow{V}) = v$, otherwise $\perp$.
  - If at least $n - (\lfloor \frac{n-1}{3} \rfloor + \alpha n)$ of $V[i]$ have the same value v, then $threshold(\overrightarrow{V}) = v$, otherwise $\perp$, where n is a number of nodes in the decision group and α is a fraction of paths intercepted by a MITM attack.

We denote a decision vector $\overrightarrow{V} = (v_1, v_2, ..., v_n)$, where $v_i = val(C_i, R_i) \in \{true, false\}$. Hence three levels of agreement on $\overrightarrow{V}$ are defined as follows.

Consensus:	$consensus(\overrightarrow{V}) = v\ threshold(\overrightarrow{V}) = v$
Threshold:	$consensus(\overrightarrow{V}) = \perp\ threshold(\overrightarrow{V}) = v$
None:	$consensus(\overrightarrow{V}) = \perp\ threshold(\overrightarrow{V}) = \perp$

Each decision node P_i must multicast its (C, R) pair to others by a *(C,R) Distribute* message. Each also saves all (C, R) pairs as a vector $\overrightarrow{V_{P_i}}$. If there is either a true consensus or a true threshold on $\overrightarrow{V_{P_i}}$, then P_i runs *Certification Issue Protocol*, which will be presented in Subsection III-E. Otherwise, P_i sends an *Authen Fault* message to P_0 and runs *Malicious Node Detection Protocol* (see Subsection III-F).

E. Certification Issue Protocol

Based on the threshold signature and the homomorphic property of a public-key cryptographic function, each decision node computes a partial certificate in a distributed fashion and sends it to a coordinator who combines all partial certificates and reconstructs a complete certificate. In the following, we use the RSA scheme as an example to explain the approach. Let public key PK be (e, N) and private (secret) key SK be d. Note that for a message m,

if $d \equiv d_0 + d_1 + ... + d_k \pmod N$, then we obtain $m^d \equiv m^{\sum_{i=0}^{k} d_i} \pmod N \equiv \prod_{i=0}^{k} m^{d_i} \pmod N$. The *Certification Issue Protocol* is shown in Table III and consists of the following two steps.

1) Partial Certificate Generation: Once a decision group has a true agreement on (C, R), i.e., consensus or threshold level, it signs the $CertReq_A||ExpTime$ with its share SK^i_{CA} and sends the partial certificate $Cert^i_A$ to the coordinator. In contrast, if there is no agreement on (C, R) pairs, a decision node runs *Malicious Node Detection Protocol* (described in Subsection III-F).

2) Complete Certificate Construction: After a coordinator P_0 gets all partial certificates with no *Authen Fault* message, P_0 combines all of them to construct A's complete certificate $Cert_A$ and then verifies the $Cert_A$ with CA's public key. If the verification is successful, P_0 sends the $Cert_A$ to A. Otherwise, some $Cert^i_A$ might be corrupted if some decision nodes are malicious or have compromised shares. Hence, P_0 repeats *Decision Group Establishment Protocol* (see Subsection III-D) to select a new decision group until the verification is successful or the number of runs exceeds a pre-defined limit. If the limit is exceeded, P_0 informs A by sending a *Cert Fault* message. Although P_0 is malicious and sends A a faulty certificate, A can detect the faulty certificate by using CA'S public key. Eventually, A can resubmit a certificate request to change the coordinator P_0 who may be malicious.

TABLE III
CERTIFICATE ISSUE PROTOCOL (P_i WHERE $i = 0, ..., n$)

Step 1. Partial Certificate

P_i: Generates $Cert^i_A = [CertReq_A, ExpTime]_{SK^i_{CA}}$
$P_i \nrightarrow P_0$: $[P_i, P_0, Partial\ Cert, Cert^i_A]_{SK_{P_i}}$
If $consensus(\overrightarrow{V}) = \perp$
Runs *Malicious Node Detection Protocol*

Step 2. Complete Certificate

P_0: If $no\ Authen\ Fault$
Reconstructs $Cert_A = \prod_{i=0}^{k} Cert^i_A \pmod N$
Verifies $Cert_A$ with PK_{CA}
If verification is successful
$P_0 \rightarrow A$: $[P_0, A, Complete\ Cert, Cert_A]_{SK_{P_0}}$
Else-If limit is not exceeded
Runs *Decision Group Establishment Protocol*
Else
$P_0 \rightarrow A$: $[P_0, A, Cert\ Fault]_{SK_{P_0}}$
Else
$P_0 \rightarrow A$: $[P_0, A, Authen\ Fault]_{SK_{P_0}}$

Since a CA group selected from a P2P community may consist of malicious nodes, we need a detection protocol to detect malicious nodes in the CA group.

F. Malicious Node Detection Protocol

Generally, if an agreement on a decision vector does not have true consensus, either A or some decision nodes P_is are malicious. Note that in our protocol, decision nodes count a node that lied or its messages have been intercepted by MITM attacks as a malicious node. If A lied or its messages have been intercepted on more than a threshold number of paths, the decision group will not trust A and not issue a certificate for A. Meanwhile, if some decision nodes send conflicting messages (*challenge* message), drop messages or omit proper procedures, our protocol must eliminate such malicious nodes from the CA group. However, this detection protocol should be executed only when there is suspicious behavior because running a BA protocol is too costly. Consequently, we divide this protocol into the following two steps (see Table IV) to optimize computation and communication cost.

1) Proof request: To avoid running a BA protocol frequently, each decision node P_i requests the vector $\overrightarrow{V_A}$ from A by sending a *Proof Request* message. Then P_i compares the $\overrightarrow{V_A}$ with its own vector $\overrightarrow{V_{P_i}}$ in two cases:

- *A's Responses Investigation:* P_i investigates the consistency of responses in $V_A[j]$ and $V_{P_i}[j]$. If there are any inconsistencies, then A lied or its messages have been intercepted because no one can counterfeit the signed responses from A. Thus, it is not necessary to continue with the second case.
- *Decision Nodes' Challenges Investigation:* P_i investigates the consistency of challenges in $V_A[j]$ and $V_{P_i}[j]$: If there are inconsistent challenges from the decision node P_j, then P_j is a malicious node in P_i's view. However, other decision nodes cannot rely on P_i's accusation because they have no knowledge about what P_j said to P_i. Otherwise P_i could make an accusation causing an honest node to be eliminated. Hence, P_i must multicast a *Byzantine Request* message to activate a BA protocol, which creates useful information so that every honest decision node can reach the same agreement on an accusation of the bad behavior (under the assumption that the upper bound of the number of malicious nodes $f \leq \lfloor \frac{n-1}{3} \rfloor$, where n is the number of decision nodes).

2) Byzantine Agreement: When a decision node receives the *Byzantine Request* message, it multicasts its own vector $\overrightarrow{V_{P_i}}$ with a *Byzantine Agree* message. After receiving all *Byzantine Agree* messages, each decision node runs the BA algorithm to detect the decision node M providing no-agreement messages. The adjacent decision nodes of M cooperate to eliminate the malicious node M and to select a new CA node to replace it. Then the decision group repeats the *Key Registration Protocol* from step 3: Challenge-Response.

In addition to the previous protocols, our framework also provides a protocol to replace the previous share of an eliminated node.

G. Share Rendering Protocol

Although the *peer retirement* (see Subsection III-B) and the *Malicious Node Detection protocol* (see Subsection III-F) provide the replacements for compromised or malicious CA-nodes, doing so leaks the shares of eliminated nodes. Therefore, our scheme uses the additive property of an (n, n) threshold signature scheme and a shuffling scheme to implement a *Share Rendering Protocol*, which gradually and randomly renders shares according to the retirement, unliveness and

TABLE IV
MALICIOUS NODE DETECTION PROTOCOL (P_i WHERE $i = 0, ..., n$)

Step 1: Proof Request

$P_i \rightarrow A$:	$[P_i, A, Proof\ Request]_{SK_{P_i}}$
$A \rightarrow P_i$:	$[A, P_i, Proof\ Response, \overrightarrow{V_A}]_{SK_A}$
P_i:	For j from 0 to n
	If $R_{P_j,A}$ in $V_A[j]$ and $V_{P_i}[j]$ is not consistent
	A lied or A's message have been intercepted
	Else-If $C_{P_j,A}$ in $V_A[j]$ and $V_{P_i}[j]$
	is not consistent
	$P_i \nrightarrow *$: $[P_i, *, Byzantine\ Request]_{SK_{P_i}}$

Step 2: Byzantine Agree

$P_i \nrightarrow *$:	$[P_i, *, Byzantine\ Agree, \overrightarrow{V_{P_i}}]_{SK_{P_i}}$
P_i:	Run BA protocol on Challenge messages
$M's\ peers$:	Eliminate M from the CA group
	Reselect a new decision node to replace M
P_i:	Go to Step 3 in Key Registration Protocol

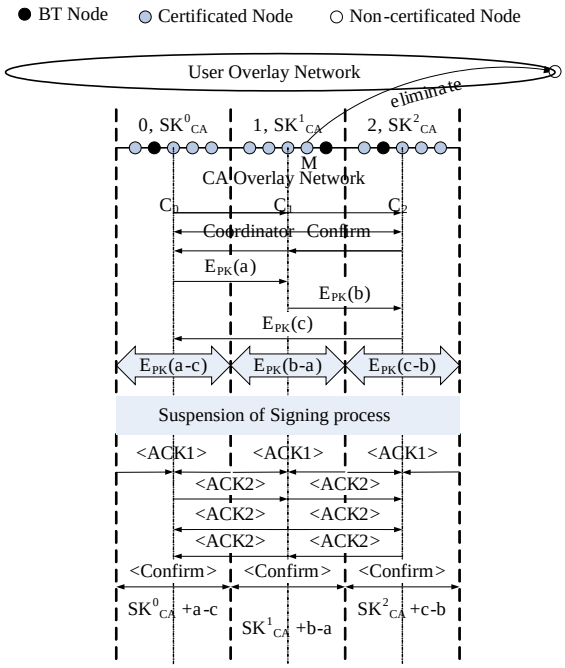

Fig. 4. Rendering the share: after a malicious node P is rejected

maliciousness of CA nodes. This protocol is activated by eliminated CA nodes' peers, who cooperate with their neighbor nodes in each dimension, e.g., x- and y-axis for 2-dimensional CAN. Without loss of generality, we explain this protocol in only one dimension and assume that a CA node M in zone 1 is eliminated while holding a share SK^1_{CA}. Therefore, the CA group must replace the share SK^1_{CA} by rendering it with its neighbor's shares (i.e., SK^0_{CA}, SK^2_{CA}). The rendering process is accomplished in the following four steps and graphically shown in Figure 4.

1) Coordinator Selection: All peers in zone 0 generate a random number a and selects a coordinator C_0, which is the node closest to the random number a, based on the local information in their peer lists. Respectively, zone 1 and 2 generate random numbers b and c, and select coordinators C_1 and C_2. Then each coordinator C_i learns the others C_j and their certificates by exchanging *Coordinator Confirmation* messages $[C_i, C_j, Coordinator\ Confirmation, Cert_{C_i}]_{SK_{C_i}}$.

2) Nonce Exchange: According to the shuffling Scheme, each coordinator C_0, C_1, C_2 sends its nonce $n = a, b, c$ to the pairwise coordinator C_1, C_2, C_0, respectively, in such a way that no coordinator knows all nonces. For confidentiality, such nonce is encrypted by the receiver's public key before attaching to a *Nonce Exchange* message $[C_i, C_j, Nonce\ Exchang, E_{PK_{C_j}}(n)]_{SK_{C_i}}$ and being sent to the pairwise coordinator.

3) Nonce Distribution: Each coordinator C_i computes the difference $diff_i$ between its nonce and the received nonce and sends the $diff_i$ encrypted by the receiver's public key to all its peers $Peer_i$ with *Nonce Distribution* messages $[C_i, Peer_i, Nonce\ Distribution, E_{PK_{Peer_i}}(diff_i)]_{SK_{C_i}}$. To prevent the *Share Rendering Protocol* causing the share inconsistency, every peer suspends new certificate requests until all holding requests are finished and then acknowledges its coordinator with *ACK1*.

4) Nonce Confirmation: When the coordinator receives *ACK1* from all its peers, it acknowledges itself and the other two coordinators with *ACK2*. Each coordinator will broadcast a *Nonce Confirmation* message $[C_i, Peer_i, Nonce\ Confirmation]_{SK_{C_i}}$ to all its peer after receiving *ACK2* from all three coordinators. Then every peer renders its share (i.e., $\hat{SK}^0_{CA} = SK^0_{CA} + a - c, \hat{SK}^1_{CA} = SK^1_{CA} + b - a, \hat{SK}^2_{CA} = SK^2_{CA} + c - b$) and continues the suspended processes.

In our scheme, a CA group gradually and randomly renders its shares in each particular part of the CA group (i.e., that zone and its neighbor zones) so that the CA group can continue working and rendering simultaneously. Moreover, to recover a CA private-key, adversaries holding some shares must obtain all n shares before any adjacent remainder-shares are rendered.

IV. SECURITY ANALYSIS

This section first analyzes two main attacks to the proposed framework (i.e., node impersonation and CA functionality interference) and then others possible attacks.

a) Node Impersonation Attacks: In this attack, a malicious user-node A' or a coalition of malicious user-nodes $\{A'_i\}_{i=1,n}$ tries to impersonate a legitimate node A in two cases: 1) A' may propose a certificate request $[A, PK_{A'}]_{SK_{A'}}$ to a CA group; and 2) $\{A'_i\}_{i=1,n}$ may launch MITM attacks to compromise the *Key Registration Protocol*. In the former case, the majority of challenge messages from a CA group will reach A instead of A'. As a result, most of the proofs of private-key possession will fail because the public/private key pairs of A and A' are different. In the latter case, $\{A'_i\}_{i=1,n}$ collude to launch MITM attacks in order to intercept and counterfeit A's certificate request and challenge-response messages. These messages are forged to convince the CA group that one of the adversaries is A and possesses A's private key. However, the collusion of $\{A'_i\}_{i=1,n}$ cannot succeed in impersonating A if they cannot intercept more than the threshold number of challenge-response messages.

b) CA Functionality Interference: We consider the case of an internal adversary (a malicious decision-node) trying to interfere with two CA functionalities: key registration and certificate issue. In key registration, when a user node sends a certificate request to a trusted gateway (a BT node) known by all decision nodes, the BT node signs both the user's public key and CA policies before both are forwarded to all decision nodes. Hence, our framework can guarantee that every decision node receives the same proposed public-key and policies because no one can modify or fabricate the registration information. For certificate issue, first, a malicious decision-node may distribute conflicting challenges to interfere with the challenge-response scheme and cause a legitimate user to fail in its key-possession proof. Our protocol leverages the BA algorithm with signed messages to detect that malicious node, eliminate it from a CA group and then replace it with a new CA node. Second, a malicious decision-node may omit a partial signature or create a wrong one to interrupt certificate issuing. Although our framework cannot verify a partial signature to detect the malicious decision node, re-establishing a new decision group or re-submitting a certificate request can mitigate this attack.

c) CA Group Subversion: A malicious user-node may intentionally send conflicting responses to make a false accusation causing a CA group to eliminate an honest decision-node. The *Malicious Detection Protocol* deliberately use the signature-based message authentication and the Byzantine algorithm to discover which decision nodes cause challenge-response invalidity. Therefore, the adversary cannot convince a decision group of false accusations against honest decision-nodes.

d) DoS Attacks: Similarly, a malicious user-node A may send a decision group conflicting responses to activate the BA protocol. To thwart the attacks, each decision node P_i compares the responses in its own vector $\overrightarrow{V_{P_i}}$ with the ones in A's vector $\overrightarrow{V_A}$ to confirm that A is not a liar before activating the BA protocol. Consequently, the adversary cannot exploit the BA protocol to disable the services of the CA group but may launch DoS attacks in other scenarios.

e) Sybil Attacks: Although our framework cannot prohibit an adversary from acquiring multiple IDs, it can limit the effect of Sybil attacks on the CA group. On the basis of a *CertCount* field in a peer list, a *Retirement* policy and an age parameter, the framework can limit the lifetime of a Sybil ID in the CA group. In addition, decision nodes are selected randomly from the CA nodes, which are selected uniformly from a P2P community in a distributed manner. Hence, the Sybil nodes cannot use their multiple IDs to subverting the CA group's honest majority.

V. Conclusion

In this paper, we proposed SOHCG, which is a fully self-organizing and self-healing system based on a CAN overlay network. In our scheme, a CA group grows up under a security/efficiency trade-off, maintains its membership in a dynamic fashion and employs a challenge-response scheme to provide an automatic key registration and a certificate issue. Meanwhile, a CAN with overloading zones compensates for the lack of fault tolerance in an (n, n) threshold signature scheme, and a CAN-based multicast is used to optimize the computation and communication cost of a BA protocol. In addition, a BA protocol with sign messages is leveraged to maintain an honest majority with the avoidance of DoS attacks in a CA group. Finally, all CA's shares are refreshed in a gradually random fashion. The security analysis shows that our scheme can prevent impersonation, collusion and MITM attacks as well as mitigate Dos and Sybil attacks. The time slot for an adversary to reveal the CA private-key is equal to the period of a refreshment, according to the frequency of certificate issues.

References

[1] A. Avramidis, P. Kotzanikolaou and C. Douligeris: Chord-PKI: Embedding a Public Key Infrastructure into the Chord Overlay Network. *EuroPKI 2007: Public Key Infrastructure, Springer-Verlag*, J. Lopez, P. Samarati, and J.L. Ferrer (Eds.), LNCS 4582, pp. 354-361, 2007.

[2] J. C. Benaloh: Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret. *Advances in Cryptology CRYPTO 86, Springer-Verlag*, A.M. Odlyzko (Eds.), LNCS 263, pp. 251-260, 1987.

[3] J. R. Douceur: The Sybil Attack. *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems, Springer-Verlag*, P. Druschel, F. Kaashoek, and A. Rowstron (Eds.), LNCS 2429, pp. 251-260, 2002.

[4] D. Boneh and M. Franklin: Efficient generation of shared RSA keys. *J.ACM, New York, NY, USA*, Vol. 48, Iss. 4, pp. 702-722, 2001.

[5] F. Cristain and C. Fetzer: The timed asynchronous distributed system model. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 10, No. 6, pp. 642-657, 1999.

[6] R. Housley, W. Ford, and D. Solo: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. *http://www.ietf.org/rfc/rfc2459.txt*, RFC 2459, 1999.

[7] F. Lesueur, L. Ḿe and V. V. T. Tong, A Distributed Certification System for Structured P2P Networks, *IFIP International Federation for Information Processing 2008, Resilient Networks and Services, Springer-Verlag*, D. Hausheer and J. Schonwalder (Eds.), LNCS 5127, pp. 40-52, 2008.

[8] F. Lesueur, L. Ḿe and V. V. T. Tong, An efficient distributed PKI for structured P2P networks, *P2P'09: IEEE Ninth International Conference on Peer-to-Peer Computing 2009*, pp. 1-10 , 2009.

[9] M. Nystrom and B. Kaliski: PKCS 10: Certification Request Syntax Specification Version 1.7. *http://www.ietf.org/rfc/rfc2986.txt*,RFC 2986, 2000.

[10] V. Pathak and L. Iftode: Byzantine fault tolerant public key authentication in peer-to-peer systems. *Computer Networks*, vol. 50, no. 4, pp. 579-596, 2006.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Schenker: A scalable content-addressable network. *SIGCOMM'01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, ACM*, pp. 161-172, 2001.

[12] S. Ratnasamy, M. Handley, R. Karp and S. Shenker: Application-Level Multicast Using Content-Addressable Networks. *NGC 2001: Networked Group Communication, Springer-Verlag*, J. Crowcroft and M. Hofmann (Eds.), LNCS 2233, pp. 14-29, 2001.

[13] A. Takeda, K. Hashimoto, G. Kitagata, S.M.S. Zabir, T. Kinoshita and N. Shiratori: A New Authentication Method with Distributed Hash Table for P2P Network. *AINAW 2008: 22nd International Conference on Advanced Information Networking and Applications - Workshops*, pp. 483-488, 2008.