Global Journal of Computer Science and Technology
Network, Web & Security
Volume 12 Issue 15 Version 1.0 Year 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Authorised Secure Host Communication under Data Provenance Verification- A Signcryption Based Contract Signing Protocol

By Bolladi Swathi & P.Vasanth Sena

Sree Chaitanya College of Engineering, Karimnagar, AP,India

Abstract - The wide qualities of distributed (ex: P2P networks) network has given us many advantages and threats for enhancement of distributed computing. The best way to reduce threats is adding a reputation-based globally trusted model. Many present trust models are failing to restrain effectively some behaviors like collusive attacks, but pay no heed towards the security of this mechanism.

GJCST-E Classification : *E.3*

Authorised Secure Host Communication under Data Provenance Verification- A Signcryption Based Contract Signing Protocol

Bolladi Swathi [α] & P.Vasanth Sena [σ]

Abstract - The wide qualities of distributed (ex: P2P networks) network has given us many advantages and threats for enhancement of distributed computing. The best way to reduce threats is adding a reputation-based globally trusted model. Many present trust models are failing to restrain effectively some behaviors like collusive attacks, but pay no heed towards the security of this mechanism.

I. Introduction

Of late, distributed computing has become popular and well recognized in a wide range of applications, like file-sharing, digital content delivery, and distributed Grid computing [1]. But the fact remains that, peer anonymity and autonomy make distributed networks easy towards attacks by any peer who is not rust worthy. The recent works [2-5] are a benchmark to the fact that the trust theories in social networks construct well recognized trust models, to find a solution for these kinds of behaviors.

The present reputation-based trust model designs trusted rank of a peer based on its past transactions, and it's similar to the peer with full trust value is offered the role of the service provider. This method has some advantages on any malicious behaviors to a certain extent, but has a meager effect when it comes to complex attacks and when disturbances are created on these reputation systems, like collusions. The researches now a day focus on the design and working of the trust system in all sensible arenas, and barely care concerning the security difficulty it faces which can damage the tag "node consistency handling". The security of node reliability handling is the most important element which assures a safe working of the trust management system (TMS). Thus, it is vital to develop and discuss about the security mechanism of the TMS.

Dealing by means of these research issues, we project node reliability based distributed trust model with the security mechanism that we refer as the secure node reliability information management (SNRIM), for distributed networks, which would scales better over node reliability information management(NRIM).

Author α : M.Tech student at Dept of CSE, Sree Chaitanya College of Engineering, Karimnagar, AP,India. E-mail : swathi.bolladi@gmail.com
Author σ : Associate Professor in Dept of Cse, Sree Chaitanya College of Engineering, AP,India.

II. Related Work

This sector gives a wide review of some of the present distributed node reliability systems, concentrating on problems like storage and veracity. We would like to at first give an outline of the node reliability systems. Kevin A. Burton designed an open privacy distributed node reliability system [5] on p2p, which hails from the distributed trust model which brought to us the idea of node reliability network, which is made up of identities and certificates. Therefore, the certainty of the identities is appreciated from a visible sub-graph of the reputable network. P2PREP [6], which is a node reliability sharing protocol designed for Gnutella, where every peer keeps track and shares the node reliability of their peers. Reputation sharing is made by distributed polling protocol. Service requesters use this trust by polling peers. Karl Aberer et.al. [7] Made a trust managing system on the distributed system which combines the trust and data management to construct a complete distributed architecture for information systems. The node reliabilities here are expressed as complaints; higher the complaints, less trustworthy it is. After every transaction, if there is dissatisfaction, a peer files a complaint stating the problem. To examine the node reliability of a peer involves searches for complaints about the peer. Kamvar et.al [8] proposed a node reliability management system, for distributed file sharing systems such as Gnutella fighting against the spread of inauthentic file. Here, every peer has a global node reliability that shows experiences of every peer with it. Sit and Morris [9] gave an idea for security of p2p networks. Their model permits nodes to make packets with arbitrary material, but lets the nodes not to intercept arbitrary traffic. They gave taxonomy of all varied attacks and at the routing layer, they find a node lookup, routing table preservation, division the network and virtualization as threat to security. They deal also with multilevel protocols, like file storage, where nodes need not have the necessary invariants, like storage replication. They work also on denial-of-service attacks, and rapidly joining and leaving the network, or arranging for various nodes which sends bulk volumes of data to overload a victim's network connection (i.e., distributed denial of service attacks). Dingledine et al. [10] and Douceur [11] work on address spoofing assaults as well. Having several potentially hazardous nodes in the system and with no trusted central head which certifies node identities and become complex to know whether you can trust the claimed identity of somebody to an unknown. Bellovin [12] finds many problems with Napster and Gnutella. He discusses how complex it is to extent the use of Napster and Gnutella use via firewalls, and the ways they pass information that users feel is personal, like the search queries given. Bellovin researches also on Gnutella's "push" feature, which functions on firewalls, useful for denial of service attacks. He feels Napster's central architecture more safe against these kind attacks, even if it needs users to trust the central server. It is to be renowned that a substitute reply for secure routing table maintenance and forwarding that we denied. This answer exchanges every node by a bunch of replicas as told by Lynch et al. [13]. The replicas are run using a state machine replication algorithm like BFT [14] that can sustain faults like Byzantine. BFT can replicate arbitrary state machines and, therefore, it can look like Pastry's routing table maintenance and forwarding protocols. Here, we look into Reputation Systems for distributed networks—highly useful design which protects the distributed network without a central component, and amplifies all the advantages of the distributed network.

III. Node reliability systems

A vital corollary of a good node reliability management is the online auction system eBay [9]. Here, buyers and sellers rate each other post transaction, and the final node reliability of a contestant is the ratings he has over the last 6 months. This system depends on a central system to store and manage these ratings.

In varied areas nodes rate each other post transaction, like in eBay system. Like, every time peer I gets a file from peer j, it rates the transaction as positive (tr(i, j) = 1) or negative (tr(i, j) = −1). Peer i can rate a download as negative, if he finds the file inauthentic or tampered with, or if interrupted. Like in the eBay approach, we may possibly characterize a local faith value s_{ij} as the sum of the ratings of the individual transactions that node i has downloaded from node j: $s_{ij} = ptr_{ij}$.

Similarly, every peer i can store many transactions it has had with node j, $sat(i,j)$ and the number of intolerable transactions it has had with node j, $unsat(i,j)$. Then, s_{ij} is defined:

$$s_{ij} = sat(i,j) - unsat(i,j) \qquad (1)$$

Previous work in distributed node reliability systems [6, 1] are based on same notions of local trust values. The obstacle is in an environment is how to deal with the local trust values s_{ij} without a central storage and management. Every previous system named above finds this problem; every system proposed has a couple of negatives. It mostly averages the ratings of some nodes and has no wide view about a peer's node reliability, or it averages the ratings of the nodes and congests the network with system messages questioning for every peer's local trust values for each query.

a) *Threat Model*

A Gnutella-like network has a power-law topology and helps Insert and Search techniques. The nodes have a predefined Join & Leave protocols. The nodes are connected with a communication channel which is not secure. As the nodes have opposing interests, a motivation is required to decrease leechers. Leechers are the ones who gain benefit from the system without giving anything to the system. The rogue nodes send malware in the network. Finally, nodes judge the quality before making Go/No-Go in every transaction and develop trust relationships mutually.

A good node reliability system gives the way to achieve the target. Any node reliability system is open to ballot stuffing and bad mouthing as told in [18]. A poor node reliability system naturally gives problems that exploit the attackers. Peers should have unique way to handle to which their node reliabilities are tagged. If they are absent in trusted central agency, an attacker gathers infinite identities and gives recommendations to itself. A node can alter the reliability data in the network to uplift its node reliability and there are problems that are in the picture based on how a given node reliability system is made. We discuss those problems and their mitigation in the sections where the design decision is made.

b) *Self-Certification*

To participate in the node reliability system, a node should have handled. The reliability of a node is represented with handle. This handle is the "identity" of the node even if does not "discover" a node, i.e., it may not lead to the real-life identity of the peer. A node gets advices for every transaction, and all advices are stored together for calculation of the reliability of a node.

In a central system, the head gives these identities. In a distributed node reliability system, self-certification [33] divides the trusted entity among the nodes and gives their own identities. Every node has its own CA that gives the identity certificate(s) to the peer. All the certificates used here are same to SDSI certificates [6]. The name of a node is with its identity and the node reliability of a CA is the node reliability.

Self-certification obviates the central trusted entity for giving identities in a central system. Peers having self-certified identities are pseudononymous in the system as there isn't a way to map the identity of a node in the system to its real-life. Though anonymity or at least pseudonymity is required in distributed networks, in a node reliability system it is a double edge sword. If there is no mapping between multiple identities and the owner (peer), the system is open to Sybil attack or Liar farms.

A node uses self-certification generating many identities and raises the node reliability of identities doing false transactions. The malicious node need not collude with distinct nodes to build its node reliability, but should generate a set of identities. The set of identities managed by one node is called an identity farm. The identities issuing a false recommendation are called a liar farm. These attacks are of the class of attacks named Sybil attacks. A node having an identity farm is as powerful subverting a node reliability system as a node colluded with many of other peers.

An identity farm is countered if, a node is not allowed to one identity or all the identities of a node are sent back the peer. A node can be stopped to one identity by mapping its identity to its real-life identity and leaving anonymity, or by making the identity generation resource high that the node cannot generate more identities. Identity generated is made resource intensive by traditional micro-payment method, although the resource restrictions have a varied impact based on every peer's resourcefulness.

In self-certification, we have a combination of approaches. Every node CA gives many identities. The advices received for a peer's identity from identities of peers, signed by the other peer's CA(s), are recognized as signed by the CA, and are made to counter the liar farms. In every transaction, the requester averages all the advices of the provider by CAs of the provider's last advisors. Hence, all the past advices owned by the provider are but they get averaged. Finally, it sums up the averages of each CA calculating the node reliability of the provider identity.

Hence, a peer should not use its own identities (all generated by the same CA) to advice its other identities.

A determined peer can begin many CAs and give groups of identities. In order to oppose a rogue node with multiple CAs, the nodes are made to batches on various grounds like a node can't be a part of many groups. For example, a distributed network in a city ensures the nodes by their zip codes. Every node gets its group certificate from the required head and attaches it to its CA. The certificate of a group head is publicly used by any node inside or outside the group. The node sends its credentials to the group and the head checks and signs the group certificate.

Unlike the traditional CA or distributed CA ways, grouping of nodes has the anonymity of the peers; when grouped with self-certification it curtails the happening of a Sybil attack. In opposition to the traditional CA, neither the group head nor the transacting nodes establish the identity of the peer. The certificate revocations are not necessary in the group-based way as the group head vouches for the real-life of the peer, unlike the traditional certificate-based approaches where many certificate attributes are attested by the head and need revocation. If a good identity is adjusted, its misuses are self destructive as its node reliability will go down if misused.

The node is named P while the head is denoted by A. Here P→A: X represents that the node (P) sends a message X to the head (A), here P_{k2} stands for the private key of P and P_{k1} represents the public key of the node $P.E_k(\tau)$ represents encryption of the phrase (τ) with K, while EB_k (X) represents blinding phrase X with key K.

1. P→A:B1={EB_{ka} (I_{Alice_r})}, I_{Alice} The peer Alice gives a BLINDING KEY, K and identity for herself (I_{Alice_r}) Alice cannot be recognized from her identity (I_{Alice_r}). She also blinds her identity (I_{Alice_r}) with the blind key Ka. B1 stands for the blind identity. Alice passes B1 to the head with her real identity that approves her membership to a group.
2. A→P:B2= $E_{P_{AuthorityK2}}$ {B1=EB_{ka} (I_{Alice_r})}, I_{Alice} The head attests the blinded identity, B1 and sends it (B2) back to the peer.
3. $E_{P_{AuthorityK2}}$ {I_{Alice_r}} = EB_{ka} {B2}}The peer un blinds the signed identity and extracts the identity authorized by the head $E_{P_{AuthorityK2}}$ {I_{Alice_r}}.

The logic in the group-based way is that in a distributed network, nodes are interested in the ranks of providers than only the value of the node reliabilities. The simulations tell that this way varies the name of nodes but it having least effect on the relative ranks of the peers. This approach is from the Google page rank idea in which the pages in proximity of other don't give the page rank of the target page in the pages at a distance [34]. The relative ranks don't object the nodes from adjusting thresholds. The thresholds depend on ranks. Adjusting the thresholds for absolute values are have a limited utility. Google has ranks instead of links pointing to/from pages. It is clear from the Google corollary that rank-based mechanisms can be measured. Debates between there might be some systems needing absolute values still take place. This paper is not into that, as use of absolute values is more complex and is specific information that is not a part of our discussion.

It is opposed and supported that this way is unjustified to nodes whose authentic advice are from nodes that are a part of a large group. We support the argument and our implementations display that the relative ranks of the providers change the least. Hence, the providers are least influenced (Δ Mean Rank Difference≈14 for varied sizes groups) by the batches of advices. The requesters who give the advice to the providers can't be influenced by the batching of advices.

c) Node Reliability Model

The standard Join methodology is made use of by peer to connect itself to a specific distributed network. The search appeal entails the peer supplicant to produce a list of nodes who have the demanded file(s) with them. RANGE indicates the count of nodes who tender a mentioned meticulous file. The peer supplicant chooses the provider with the peak status by instigating the cryptographic procedure which involves the peer supplicant making use of the Download methodology of the network for downloading the relevant file mentioned by the client, which again assists in validating the reliability, dependability and the value of the file. A proposal is then sent to the peer client between min - recommendation and max - recommendation, which are limited to the restrictions ensuring that a single implication doesn't utterly annul or radically improve the meticulousness of a supplicant. On receiving the suggestions from the client, it averages the prior received implications and incorporates the recently received ones to estimate its repute.

The factors mentioned above can be assigned values by the means of Decision Theory, Game Theory, and Probability and function F() is identified on the basis of intensity levels of menace faced by nodes in the distributed network. The function F() in this paper is described as the arithmetic average of the suggestions that are collected by the peer supplicant. The recommended node reliability copy is self governing as compared to topology of the distributed network, nodal addressing formats, bootstrap procedures, joining and leaving protocols of the nodes present and the name service.

A negative suggestion may be issued by an applicant to the peer supplicant which may turn out to be hazardous concerning its node reliability even though the supplicant actually is worthy of a positive recommendation for a specified transaction. If in a way, only positive recommendations are accepted, then it would be tougher to distinguish between new and bad peers. Hence an assumption is made here that both positive and negative proposals are permitted and a given peer would no longer cooperate with those nodes who frequently deliver negative proposals.

d) Contract signing between peers: a signcryption approach

The entire process starts here with the employment of RSA signature algorithm [42] otherwise known as Signcryption. At this point, the 1st user divides his private key d into d1 and d2 such that $d = d1 + d2$ by following park[40]. The signature of this user has to be exchanged with the other and this signature is $\sigma_A = h(m)^{d1} \bmod n$. The partial signature generated by the 1st user is to assure that he has zero-knowledge base and this is done by Gennaro topology[27].The relations we have are defective owed to network failure or router's attacks [36],[46]. But, TTP is reliable since the messages inserted reach the destination for sure but with some delay.

i. *Registration Protocol*

The receiver of the information has only to record i.e. merely the recording process of the initiator with TTP is enough. He then gets a long-term voucher along with CA. After this, the following processes are done: (for our convenience, let the sender be BOB and receiver as ALICE.)

a. Alice first sets an RSA modulus $n = pq$, where p and q are two -bit safe primes, i.e., there exist two primes p' and q' such that $p = 2p' + 1$, $q = 2q' + 1$. After, Alice selects her random public key $e \in_R \mathbb{Z}^*_{\phi(n)}$, and calculates her private key $d = e^{-1} \bmod \varnothing(n)$, where $\phi(n) = (p-1)*(q-1)$. At last, Alice registers her public key with a CA to get her certificate C_A, which binds her identity and the corresponding pubic key (n, e) together.

b. Alice randomly splits d into $d1$ and $d2$ such that $d = d1 + d2 \bmod \phi(n)$ by choosing $d1 \in_R \mathbb{Z}^*_{\phi(n)}$, and computes $e_1 = d_1^{-1} \bmod \phi(n)$. She also generates a sample message-signature pair (ω, σ_ω), where $\omega \in \mathbb{Z}^*_n \setminus \{1, -1\}, ord(\omega) \geq p'q'$ and $\sigma_\omega = \omega^{d1} mod\, n$. Then, Alice sends $(C_A, \omega, \sigma_\omega, d2)$ to the TTP but keeps (d, d_1, d_2, e_1) secret.

c. The TTP first checks for the validation of Alice's certificate C_A. After that, the TTP checks that the triple $(\omega, \sigma_\omega, d2)$ s arranged correctly. If the whole thing is in exact order as per its rules, TTP saves d2 and generates a voucher V_A by computing $V_A = Sign_{TTP}(C_A, \omega, \sigma_\omega)$. This proves the TTP's signature on message $(C_A, \omega, \sigma_\omega)$, which guarantees that the TTP can issue a valid partial signature on behalf of Alice by using the secret $d2$.

ii. *Signature Exchange Protocol*

Before all this, a contract has to be agreed between bob and Alice and they should sign it. It should also has a deadline, and identify the Alice, Bob, and TTP.

a) Initially, the initiator Alice has to compute her partial signature $\sigma_1 = h(m)^{d1} \bmod n$, and then sends the triple $(C_A, \omega, \sigma_\omega)$ to the responder Bob. Here, $h(.)$ is a cryptographically secure hash function.

b) After receiving (C_A, V_A, σ_1), Bob first verifies that C_A is whether issued by CA, and V_A is Alice's voucher created by the TTP. Then, Bob checks if the identities of Alice, Bob, and the TTP are correctly mentioned as part of the contract '*m*'. If all these checking are ok, Bob initiates the below interactive zero-knowledge protocol with Alice to check whether σ_1 is Alice's valid partial signature on contact.

   i. Then Bob selects two numbers $i, j \in_R [1, n]$ at random, and a challenge c to Alice is sent by computing $c = \sigma_1^{2i} \sigma_w^{\ j} \bmod n$.

   ii. Receiving the challenge c, Alice calculates the response $r = c^e \bmod n$ She then returns her commitment $\bar{r} = TCcom(r, t)$ to Bob using a random number t, where $TCcom$ is the commitment algorithm.

   iii. After receiving the commitment $\bar{r}$, Bob sends Alice the pair (i, j) to acknowledge that he is done with the challenge c properly.

   iv. Alice verifies for correct preparation of c, that is $c \equiv \sigma_1^{2i} \sigma_\omega^{j} \bmod n$. If ok, Alice withdraws his commitment $\bar{r}$ by knowing the responses (r, t) to Bob. With this (r, t) , Bob knows σ_1 as valid if and only if $r \equiv h(m)^{2i} \omega^j \bmod n$ and $\bar{r} \equiv TCcom(r, t)$.

c) Bob checks the σ_1 Alice's valid partial signature and the deadline t mentioned in contract m is whether enough for resolving the dispute resolution from the TTP. Then only he sends his signature σ_B to Alice.

d) After receiving σ_B, Alice has to check whether it is Bob's valid signature. If it is, she sends Bob the partial signature σ_2 by computing $\sigma_2 = h(m)^{d2} \bmod n$. As Bob receives σ_2, he sets $\overline{\sigma_A} = \sigma_1 \sigma_2 \bmod n$, and accepts σ_2 as valid if and only if $h(m)^2 = \overline{\sigma_A}^{2e} \bmod n$. Here, Bob can receive Alice's standard RSA signature σ_A on message m from $\overline{\sigma}_A$. If all this do not happen, Bob seeks the help of TTP for connection before the expiry of the date.

IV. Node reliability exchange protocol

The status swapping procedure is commenced with the node supplicant when the node applicant chooses the supplicant with the highest status. This procedure requires the applicant to be represented as R and the node supplicant is represented as P. As in R→P: X represents that the node sends a message X to the supplicant (P).denotes private key of node P while P_{k1} denotes public key of the peer $P.E_k(\tau)$ denotes encryption of the phase (τ) with key K and E B_K (X) symbolizes blinding phrase with a key K. H(λ) denotes a one way hash of the value λ. This procedure supposes that obtainable functions are inserting and search, but are not flexible enough for nodes which may not be proposed tag along the join and leave procedures of the network. The status swapping procedure contains the following phases:

Step 1: R→P: RTS & IDR a REQUEST FOR TRANSACTION (RTS) is sent by the node applicant along with its own IDENTITY CERTIFICATE (IDR) to the node supplicant as it is required for authentication purposes in Step 7.

Step 2: P→ R: IDP & TID & $E_{p_{k2}}$ (H(TID)‖RTS . The peculiar IDENTITY CERTIFICATE (IDP), the CURRENT TRANSACTION ID (TID) and the signed TID, $E_{p_{k2}}$ (H(TID)‖RTS is sent by the node supplicant wherein signed TID is essential for the supplicant to avoid duplication of the usage of the same transaction id again. The applicant also applies for this signed TID and piles it up in the network at the end of the procedure for admission to other peers.

Step 3: R : LTID (Max (Search(PK1‖ TID)). The value of the LAST TRANSACTION ID (LTID) that was used by the supplicant is gathered by the node applicant who then combines the public ke P of the node supplicant along with the string TID and a search operation is carried out. Any node present in the network responds only when it has the relevant TID that is specified by the applicant and the node applicant chooses the highest TID out of all the TIDs received. The highest TID value becomes the LTID. It is certainly possible that the node supplicant may conspire with the node who piled up its last LTID and may modify it, but this is impossible as the applicant registers relevant information.

Step 4: R : IF(LTID≥ TID)GO TO Step 12 Foul play is presumed if the value of LTID initiated by the node applicant is originally from some other random transaction and applicant jumps to Step12.

Step 5: R→P: Past Recommendation Request & r. If the step 4 check gives successful results, then applicant requests the supplicant for the earlier received proposals. If the current transaction being performed is, say Nth transaction, the applicant makes a head-on request for N-1th,N-2th,….,N-nth proposals where r<N. The node applicant is solely responsible for deciding the value of r and is considered to be directly proportional to the applicant's venture in the transaction.

Step 6: P→R: CHAIN, $E_{p_{K2}}$ (CHAIN)

CHAIN=({RE C_{N-1} ‖ EZ_{N-1K2} (H(RE C_{N-1}))}‖

{RE C_{N-2} ‖ $E_{Z_{N-2K2}}$ (H(RE C_{N-2} ,RE C_{N-1}))} ‖

{RE C_{N-3} ‖ $E_{Z_{N-3K2}}$ (H(RE C_{N-3} ,RE C_{N-2}))}‖

{RE C_{N-4} ‖ $E_{Z_{N-4K2}}$ (H(RE C_{N-r} ,RE C_{N-r-1}))})

The earlier received proposals RE C_{N-1} , RE C_{N-2},……, RE C_{N-3} which were provided by nodes Z_{N-1}, Z_{N-2},….., $Z_{N\ 3}$ .is sent by the supplicant. The CHAIN is singed so as to enable the applicant to hold supplicant responsible for the chain. The supplicant can, in no way, change the proposals that have been assessed by the earlier applicants. Consider an applicant (say Z_1) has signed both the (ι th) and the previous (ι -1th) recommendation using its private key Z_{K2}, as $E_{Zn_{K2}}$ (H(RE C_{N-3} ‖ RE $C_{N-(\iota-1)}$)), in no way can a supplicant alter the CHAIN.

Step 7:R : Result=Verify(RE C_{N-1} ;RE C_{N-2} RE C_{N-r})

If Result != Verified GO TO STEP 12

A simple public key cryptography protocol is employed by an applicant to authenticate the CHAIN. The authentication process is easier when a supplicant possesses certificates of all the nodes with whom it had connections earlier. In case it doesn't have one, it accumulates it from the supplicant itself. The provider had obtained its requester's certificate in Step 1. Liar farms (specified in Section 3.2, paragraph 2) are checked for by the applicant. The applicant jumps to Step 12 in case the authentication process fails.

Step 8: Contract signing between node selected under node reliability check and node that requesting the service

Signature exchange protocol will get into action between Peer "SRP" that requesting the service and Peer "SPP" that selected as service provider by node reliability check.

Initially, the initiator SRP has to compute her partial signature $\sigma_1 = h(m)^{d1} \bmod n$, and then sends the triple $(C_A, \omega, \sigma_\omega)$ to the responder SPP. Here, $h(.)$ is a cryptographically secure hash function. After receiving (C_A, V_A, σ_1), SPP first verifies that C_A is whether issued by CA, and V_A is SRP's voucher created by the TTP. Then, SPP checks if the identities of SRP, SPP, and the TTP are correctly mentioned as part of the contract 'm'. If all these checking are ok, SPP initiates the below interactive zero-knowledge protocol with SRP to check whether σ_1 is SRP's valid partial signature on contact. Then SPP selects two numbers $i, j \in_R [1,n]$ at random, and a challenge c to SRP is sent by computing $c = \sigma_1^{2i} \sigma_w^{j} \bmod n$. Receiving the challenge c , SRP calculates the response $r = c^e \bmod n$ She then returns her commitment $\bar{r} = TCcom(r,t)$ to SPP using a random number t , where $TCcom$ is the commitment algorithm. After receiving the commitment $\bar{r}$, SPP sends SRP the pair (i, j) to acknowledge that he is done with the challenge c properly. SRP verifies for correct preparation of c, that is $c \equiv \sigma_1^{2i} \sigma_\omega^{j} \bmod n$. If ok, SRP withdraws his commitment $\bar{r}$ by knowing the responses (r,t) to SPP. With this (r,t) , SPP knows σ_1 as valid if and only if $r \equiv h(m)^{2i} \omega^j \bmod n$ and $\bar{r} \equiv TCcom(r,t)$. c). SPP checks the σ_1 SRP's valid partial signature and the deadline t mentioned in contract m is whether enough for resolving the dispute resolution from the TTP. Then only he sends his signature σ_B to SRP. After receiving σ_B , SRP has to check whether it is SPP's valid signature. If it is, she sends SPP the partial signature σ_2 by computing $\sigma_2 = h(m)^{d2} \bmod n$. As SPP receives σ_2 , he sets $\overline{\sigma_A} = \sigma_1 \sigma_2 \bmod n$, and accepts σ_2 as valid if and only if $h(m)^2 = \overline{\sigma_A}^{2e} \bmod n$. Here, SPP can receive SRP's standard RSA signature σ_A on message m from $\overline{\sigma_A}$. If all this do not happen, SPP seeks the help of TTP for connection before the expiry of the date.

Step 9: P→R : File or Service

The file or service is afforded as per the obligation specified concerning search operation performed for the supplicants.

Step 10: R →P : B1 =E B_{Ka} (REC‖ TID‖ $E_{R_{K2}}$ {H(REC, ‖ TID)})

A BLINDING KEY (Ka) is produced by an applicant on receiving the service, who then combines the RECOMMENDATION (REC) and the TRANSACTION ID (TID) it had received in Step 2 and signs it. Consequently, the signed proposal is blinded along with the blinding key, Ka. This is done in order to entrust the supplicant to the proposal received before it actually knows the value, lest it disowns it on recognizing that it is low. It is also involves the fact that the supplicant made use of TID in a blinded suggestion from the node applicant, which is also authenticated by the applicant itself. The blinded proposal includes the Chain that is consequently used by the supplicant to certify its status to some other applicant.

Step 11:

a. P →R : B1|| $E_{P_{K2}}$ (H(B1),nonce),nonce

b. R→P : Ka

A NONCE is sent by the supplicant after signing the proposal even though it is unable to see the proposal and acknowledges it back to the applicant, who then authorizes the signature and sends blinding key Ka to the supplicant to unblind the received string in Step10a and confirms the received proposal.

Step 121: Insert

$$(IDR;\{REC \| TID \| E_{R_{K2}} \{H(REC) \| H(TID)\}\})$$

The proposal assigned to the supplicant (REC), the transaction id (TID), and its own identity certificate is verified by the applicant and is then accumulated in the network using Insert methodology of the distributed network which marks the end of the transaction.

Step 13: Step 12 is concerning the methodology executed by an applicant when foul play is anticipated.

ABORT PROTOCOL

R: Insert (IDR; {CHAIN ||TID|| $E_{R_{K2}}$ {H(CHAIN) || H(TID)}})

If the authentication process in Step7 fails, the applicant takes the CHAIN that was verified b the supplicant and also the TID is taken into consideration after which, it is signed and the Insert methodology is preferred to be made use of to insert the chain and also its own identity certificate into the network. Subsequently, any suitable applicant wil be able to confirm with the statistics of the failed authentication efforts and a MIN RECOMMENDATION for that TID is presumed for the supplicant. Fafe proposals cannot be encouraged to be inserted into the network as TID is to initiated that is verified by the supplicant. If an applicant reaches Step 12 from Step 4 without any possible hindrances, it will then apply for the Chain form the supplicant and will then afterward execute

$$R: \text{Insert}(IDR, \{CHAIN \square TID \square \{H(TID \square RTS))\}\}) .$$

V. Analysis of the protocol

Only a single search request is supposed to be commenced in the network so as to gather the already received proposals that were previously received by the supplicant. Also able to prevent the tampering node reliability provided by SRP to SPP by nodes that in path. This process is required the accountability of tackling the issue of unbalanced nature of availability of nodes in the network, which is measured to be a main subject concerning distributed networks.

1. The supplicant unintentionally forwards the wrong TID in Step 2. Consider that id which the supplicant forwards as TID and the LTID be the last Transaction ID for the supplicant. The value of TID is always supposed to be equivalent to LTID + 1. If in case of TID' > LTID+1, there arises a situation wherein there will be inexplicable misplaced proposals. If again in case of TID' < LTID+1, then the supplicant will be caught up with in the Step 4 of the procedure, as the last id issued and used by the supplicant was made public and accessible to all the peers. The value of TID is considered as 0 if a node is for the first time donning the role of a supplicant.
2. The transaction in Step 8 will not be terminated by the supplicant. A supplicant is allowed to abandon the transaction after providing the applicant with the requested requisite information in Step 8 and also can abandon the transaction after Step 9. In both the cases, there is an absence of a proposal by the supplicant for the transaction id TID. The proposal in Step 11 can be liberated by the applicant provided the supplicant fails to verify and sign the blinded proposal, without acquiring the supplicant's signature. In the next transaction, precisely TID+1, the supplicant again fails to illustrate the proposal for that relevant transaction, TID to the transaction's applicant, TID+1 and hence the new applicant entrusts itself with the job of scanning the network making use of Search methodology for TID. In case TID is found, the suggested proposals are also found out pertaining to the suppliant in the transaction. The applicant will then be responsible as the TID would by then have been signed b the supplicant, who will have to acknowledge the proposal as it comprises the signature of the supplicant, TID & $E_{P_{K2}}$ (H (TID)). A minimal suggestion TID is presented to the supplicant by the node applicant in the absence of the availability of the required proposal. If in Step 10, the supplicant acknowledges the signed blinded proposal B1 & $E_{P_{K2}}$ (H (B1)), the applicant refuses to send the key, Ka and directs itself to Step 10, missing all the requisite steps, and then the supplicant scans the entire network and acquires the verified proposal of

Global Journal of Computer Science and Technology (E) Volume XII Issue XV Version I Year 2012

the applicant. If an applicant skips or fails to execute Step 10, then in the upcoming transaction TID+1, LTID is looked for by the new applicant and fails in his endeavor. Hence, TID can be considered as terminated and the next transaction can be continued with the transaction id provided, TID.

3. Collusion by rogues or liar farms. All status systems are prone to complicity on account of its nature. It is possible for two or more liar farms to combine and conspire in order to augment each other's status. The influence of the conspiracy can be alleviated by classifying proposals on the basis of personage *identities*, substantiating agencies etc. The list of conspirators can be circulated, thereby, guarding the remaining nodes from an possible attack. Peers when recognized as conspirators will not be permitted to get back into the stream of network and hence they have an impetus next to conspiracy. The series of proposals of the plotters will aid in offering support that few nodes are conspiring, thereby, protecting good nodes and from the intrusion of bad nodes into the network.
4. Multiple requesters and concurrency. A supplicant in the presently used procedure will not be provided with the facility of making use of the same identity in the synchronized communication. The first option for process intensification is that the supplier identifies and familiarizes all its applicants with each other. As a result, the verification process performed in Step 4 is performed amidst a group of applicants and results are arranged in accordance with the fact that TID dissimilarity needs to be initiated due to more number of applicants. After integrating the augments, there would be a bi party procedure that would still be prevalent where the cluster of applicants is considered to the second party while the supplicant is supposed to be the first party. The figure 1 explores the ability of the proposed model to prevent the false node reliability submitted by unauthorized nodes that acts as a service request node SRP.

We can observe that contract signing by signcryption approach is most effective to prevent the node reliability tampering attacks. Even node communication with contract signing also victimized few times but victimization occurred due to contract signing breakage. Hence if contract sign is alive then attacked to tamper the node reliability is almost null. The figure 2 confirms the stable growth in execution time when considers this contract signing process, which was compared with node communication process without contract signing.

Hike at node communication execution time that is negligible when consider the improvement in prevention of node reliability tampering attack attempts.

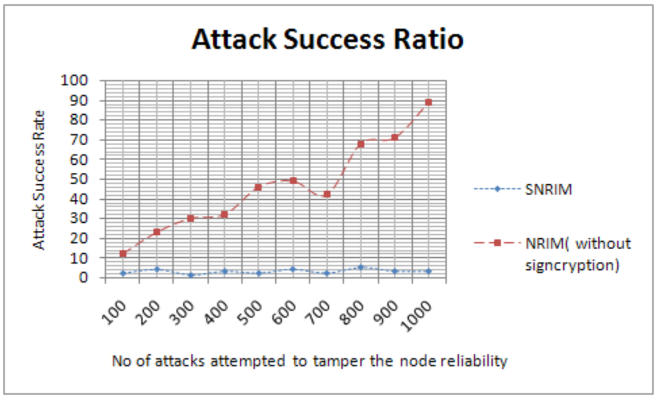

Fig. 1 : Attack success rate on NRIM and SNRIM

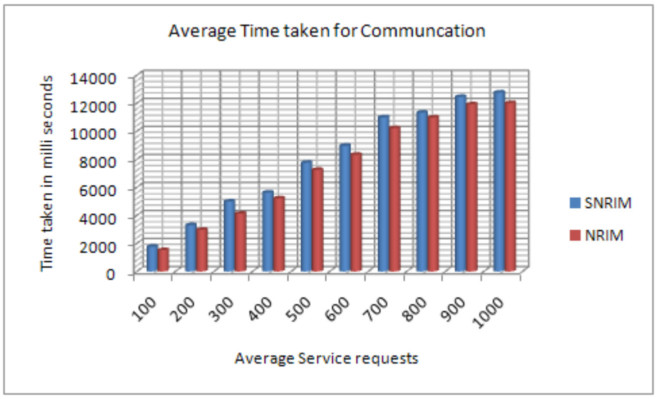

Fig. 2 : Average time taken to finish service request in SNRIM and NRIM

VI. Conclusion

Here in this paper we proposed a signcryption based contract signing for node communication based on node reliability check. The results are evident that proposed two way node reliability checking model is effective to avoid the node reliability tampering attack efforts. The planned model is screening a little hike in average process time of node communication, which can be negligible in the context of node reliability tampering attack avoidance. In future we plan to find a solution to avoid the contract sign breaching.

References Références Referencias

1. M. Ripeanu, I. Foster, and A. Iamnitchi, \Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design," IEEE Internet Computing Journal, vol. 6, no. 1, 2002.
2. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, \A scalable content addressable network," in ACM SIGCOMM, Aug. 2001.
3. I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, \Chord: A scalable Peer-To-Peer lookup service for internet applications," in ACM SIGCOMM, Aug. 2001, pp. 149{160.

4. E. Adar and B. A. Huberman, \Free riding on Gnutella," Tech. Rep., Xerox PARC, 2000.
5. S. Saroiu, P. K. Gummadi, and S. D. Gribble, \A measurement study of peer-to-peer sharing systems," in SPIE Conference on Multimedia Computing and Networking (MMCN), Jan. 2002.
6. K. Aberer and Z. Despotovic, \Managing trust in a peer-2-peer information system," in Ninth International Conference on Information and Knowledge Management (CIKM), Nov. 2001.
7. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, \A reputation-based approach for choosing reliable resources in peer-to-peer networks," in 9th ACM Conference on Computer and Communications Security, Nov. 2002.
8. S. D. Kamvar, M. Schlosser, and H. Garcia-Molina, \Eigenrep: Reputation management in p2p networks," Unpublished work, 2003.
9. S. Lee, R. Sherwood, and B. Bhattacharjee, \Cooperative peer groups in nice," in IEEE INFOCOM, Apr. 2003.
10. L. Xiong and L. Liu, \Building trust in decentralized peer-to-peer communities," in International Conference on Electronic Commerce Research (ICECR-5), Oct. 2002.
11. \Gnucleus home page," http://www.gnucleus.com/.
12. K. Sripanidkulchai, \The popularity of gnutella queries and its implications on scalability," White Paper Featured on O'Reilly's website http://www.openp2p.com/, Feb. 2001.
13. J. Chu, K. Labonte, and B. N. Levine, \Availability and locality measurements of peer-to-peer le systems," in ITCom: Scalability and Trac Control in IP Networks. July 2002, vol. 4868 of Proceedings of SPIE, Proceedings of SPIE.
14. \Kazaa participation level," http://www.kazaa.com/.

This page is intentionally left blank