P2P Secure Collaboration between Byzantine Processes in Heterogeneous distributed – Processing Systems

Ramesh Dharavath
Associate Professor, Dept of CSE
Pulipati Prasad Institute of Technology & Science
Khammam, Andhra Pradesh, India

Vallem Swetha Reddy
Assistant Professor, Dept of CSE
Pulipati Prasad Institute of Technology & Science
Khammam, Andhra Pradesh, India

ABSTRACT

The www and related technologies have made multi domain collaborations a reality. Collaborations enable processes to effectively share resources; it introduces several security and privacy challenges. Managing security and efficient exchange of information is even more challenging. In this paper, we propose a distributed secure frame work between Byzantine processes (nodes) in order to predict and resolve the functionalities of communication errors in collaboration environments. We introduce the idea of secure paths, which enables the front-end clients (e.g. Web browsers) that invoke application servers (e.g. web servers) to access the back-end databases when an end-user interacts. We present a cryptographic protocol for ensuring secure and timely availability of the data of a peer to other peers. Furthermore, we present an on-demand path discovery that enable peers to securely discover paths in the collaboration environment.

Keywords

Secure paths, Security, distributed systems, collaboration environment.

1. INTRODUCTION

The www has become integrated into practices of individuals, business, and governments. In such a combined world, there are immense possibilities of collaboration in distributed environments. Though interoperability has several advantages and is crucial in the context of new dynamic collaborative applications and adaptive enterprises, it introduces several security and privacy concerns. In particular, a domain represents a core element in a collaborating environment. A domain is a separate autonomous entity that manages a group of resources and has its own administration and access control policies. Collaboration could be viewed as an interoperation between the access control policies of the involved domains. It is more challenging to handle security in a fully distributed and dynamic interoperation environment where domains join and leave in an ad hoc manner and in the absence of a trusted mediator.

A client submits a request to some application Server on behalf of an end-user; the application server Processes the client's request, stores the resulting state in a back-end database, and returns a result to the client. This simple interaction scheme is at the centre of the called business mode today. If a failure occurs at the middle or back-end tier during request processing, or a timeout mode expires at the client side, the end-user typically receives an exception notification. This does not convey what actually happened, or whether a new state was actually stored in the database. In practice, end-users strongly resubmit the request with the risk of committing multiple server-side transactions. P2P networks are more vulnerable to dissemination of malicious or spurious content, malicious code, viruses, worms, and trojans than the traditional client-server networks, due to their unregulated and unmanaged aspect.

The traditional mechanisms for generating trust and protecting client-server networks cannot be used for pure P2P networks. This is because the trusted central authority used in the traditional client-server networks is absent in P2P networks. Introduction of a central trusted authority like a Certificate Authority (CA) can minimize the difficulty of securing P2P networks. The major disadvantage of the centralized approach is, if the central authority turns malicious, the network will become vulnerable. A two-party cryptographic protocol not only protects the reputation information from its owner, but also facilitates secure exchange of information between the two peers participating in a transaction.

1.1 Contributions

The contributions in this paper can be summarized as follows:

- We present a secure collaboration environment and discuss the security collaboration challenges in such an environment. We define access paths and present access path security requirements in a secure collaboration.
- We provide a framework for enabling secure collaboration in an agent-free environment, in which access control decisions are dependent on the user's access history in the collaboration environment.
- We discuss several security attacks that can be performed in a secure environment and provide mitigation techniques to such attacks.
- A self-certification-based identity system protected by cryptographically blind identity mechanisms.
- A light weight and simple reputation method.
- An attack resistant cryptographic protocol for generation of authentic global reputation information of a peer.

We do not make any operations on the failure task to detect the scheme used by the client-side software to detect the crash of application servers, and we tolerate failure detection mistakes among application servers. Network partitions might block the algorithm, but as long as we assume that partitions are eventually repaired, our algorithm ensures Transaction semantics.

2. RELATED SCENARIO

We consider a distributed system with a finite set of Processes that communicate by message passing. Processes fail by crashing. At any point in time, a process is either up or down. A crash causes a transition from up to down, and a recovery causes the transition from down to up. The crash of a process has no impact on its stable storage. When it is up, a process behaves according to the algorithm that was assigned to it. Processes do not behave maliciously. We review current work done for protecting the users of distributed systems using distributed CAs. This section is focused on distributed systems with one or more central credentials.

Publius - is a monolithic system comprised of a set of independently managed servers. It is censorship resistant and allows a publisher to publish anonymously. It uses cryptographic secret splitting techniques and divides the secret among a set of servers.

SDSI - is a Simple Distributed Security Infrastructure, Simplifies the X.509 certificates design and provides the means for self-certification, local name spaces, secure formation of groups, and simple access control mechanisms.

RBAC- Role-Based Access Control was introduced in 1992 by Ferraiolo and Kuhn. RBAC associates permissions with roles and not with users.

Cryptographic blinding- enables an authority to digitally sign a document without seeing the content of the document.
COCA- uses a set of distributed CA servers and provides fault tolerance and mitigation against denial of service attacks. COCA puts lose constraints on the communication channels among the servers and between the client and the servers in order to improve deployability.

The protocols and algorithms presented in this paper can still be applied when other access control models are adopted. We have chosen RBAC because it is suitable for specifying the security requirements of a wide range of commercial, medical, government applications, and moreover, it is being standardized by the National Institute of Standards. A domain that does not use RBAC as its access control model can easily generate an export RBAC policy to join the collaboration.

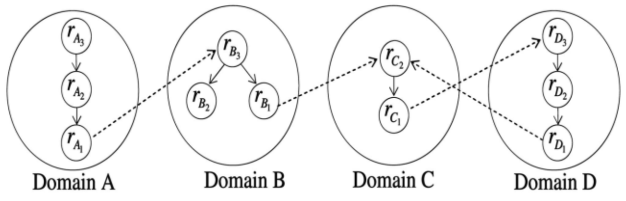

Fig. 1. Collaboration and dissimilarities.

2.1 Clients

Client processes are sketched by c1; c2; . . . ; ck (ci € Client).We assume a domain, "Request," of request values, and we facilitate how requests in this domain are submitted to application servers. Clients have an operation issue (), which is invoked with a request as parameter (e.g., on behalf of an end-user). We say that the client issues a request when the operation issue () is invoked. The issue () primitive is supposed to return a result value from the domain "Result." When it does so, we say that the client delivers the result (e.g., to the end-user).

2.2 Application Servers

Application server processes are denoted by a1; a2; am (ai € Appserver). We will discuss the situation of having a dynamic set of application servers. Application servers are stateless in the sense that they do not maintain states across request methodologies: Requests do not have side-effects on the state of application servers, only on the database state. Thus, a request cannot make any assumption about previous requests in terms of application-server state changes.

2.3 Database Servers

Database server processes are denoted by s1; s2; Sn (si € Server). We consider a fixed set of databases for simplicity of presentation. Since we want our methodology to apply to off-the-shelf database systems, we view a database server as an XA engine. In particular, a database server is a real server: It does not invoke other servers, it only responds to invocations. We do not represent full XA functionality, we only represent the transaction Commitment aspects of XA (prepare () and commit ()). We use two non-blocking primitives, vote () and decide (), to represent the transaction commitment functionality. The vote () primitive takes as a parameter a result identifier and returns a vote in the domain Vote = {y, n}. Surely speaking, a yes vote means that the database server is able to commit the result (i.e., the corresponding transaction). The decide () primitive takes two parameters: a result identifier and an outcome in the domain Outcome = {commit, abort}.

2.4 Secure Collaboration

In this section, we present the notion of agent-free secure collaboration environment. In an agent-free environment, there is no global entity ensuring secure interoperability among the collaborating domains. Fig. 2 shows both the mediated and the agent-free types of collaboration environments. In an agent-free environment, we have the following main assumptions:

- Processes have limited information about the collaboration environment. Each domain only has information about its own security policy, the cross-links and restricted links in which it is involved.
- Each domain is responsible for making its own access control decisions. The first priority of each domain is to ensure that its security policy is not violated.
- Domains are willing to collaborate in propagating access control messages and requests across domain boundaries.

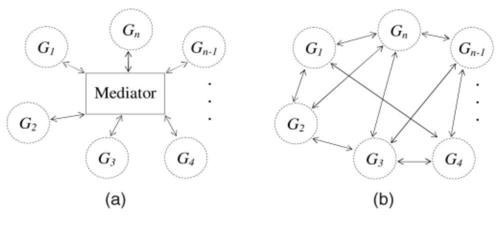

Fig. 2 Collaboration environment with and without an agent. (a) Mediated. (b) Agent-Free.

We included this section in order to summarize the identity management methods used by the current P2P networks, reputation systems, and client-server networks. In P2P networks, there is no way to ascertain the distinctness of a peer in the absence of a central agency or without using external means. This thesis has been named as the Sybil attack and proved in and has been reiterated in.

3. METHODS FOR COLLABORATION

Our framework enables domains to make localized access control decisions based on the user's access history in the collaboration environment. It is composed by the following. Major modules (see Fig. 3):

- Request processing module. Enables domains to generate and evaluate user access requests across domains. Request processing enables domains to accumulate secure access paths and use these access paths to evaluate cross-domain access requests.
- Path authentication module. The user path migrates with the user requests; path authentication ensures is required to check the authenticity of the received paths. In addition, path authentication generates path signatures for generated requests.
- Path discovery module. Enables users residing in their home domain to discover secure access paths to roles accessible in target domains. Path discovery could be on-demand or proactive depending on the collaboration environment.

In order to participate in the reputation system, a peer needs to have a handle. The reputation of a peer is associated with its handle. This handle is commonly termed as the "identity" of the peer even though it may not "identify" a peer, i.e., it may not lead to the real-life identity of the peer. A peer receives a recommendation for each transaction performed by it, and all of its recommendations are accumulated together for calculation of the reputation of a given peer.

3.1 The Transaction Phenomenon

The Transaction phenomenon is defined with three categories of properties: termination, agreement, and validity. Termination captures liveness guarantees by preventing blocking modularity. Agreement captures safety guarantees by ensuring the consistency of the client and the databases. Validity restricts the space of possible results to exclude meaningless ones.

- **Termination.**

(T.1) If the client issues a request, then, unless it crashes, the client eventually delivers a result.

(T.2) If any database server si votes for a result, then si eventually commits or aborts the result.

- **Agreement.**

(A.1) No result is delivered by the client unless the result is committed by all database servers.

(A.2) No database server commits two different results.

(A.3) No two database servers decide differently on the same result.

- **Validity.**

(V.1) If the client delivers a result, then the result must have been computed by an application server with, as a parameter, a request issued by the client.

(V.2) No database server commits a result unless all database servers have voted {yes} for that result.

3.2 Reputation Exchange Method

Once the requester has selected the provider with the highest reputation, it initiates the reputation exchange method with the provider. In the reputation exchange Protocol, the requester is denoted by R while the provider is denoted by P. Here R→P: X denotes that the requester (R) sends a message X to the provider (P). The symbol P_{K2} represents the private key of the peer P and P_{K1} represents the public key of the peer P. $E_K(\urcorner)$ represents encryption of the phrase $(\urcorner)$ with key K, while $EB_K(X)$ represents blinding phrase X with key K. $H(\partial)$ denotes a one way hash of the value ∂. This protocol only assumes that insert & search functions are available and are not resilient to peers
that may not follow the recommended join & leave protocol of the network. The steps in the reputation exchange protocol are as follows:

Step 1: R→P: RTS & IDR

Step 2: P→R: IDP & TID & E_{PK2} (H (TID || RTS))

Step 3: R: LTID = Max (Search PK1 || TID))

Step 4: R: IF (LTID >=TID)

Step 5: R→P: Past Recommendation Request & r

4. PATH DISCOVERIES (On – Demand)

Domains are able to collaborate with neighboring domains through the established collaboration cross-links. Neighboring domains are single hop collaborations as they only involve two domains. Single hop collaborations are easy to achieve and initiate as domains already have full knowledge of their established cross-links. On the other hand, to collaborate through multihop collaborations domains need to build one or more single access paths to target domains. To enable domains to discover available multihop collaborations a distributed path discovery algorithm is required. The discovery algorithm enables domains in an interoperation environment to discover paths to roles in other domains, whether reachable through one or more intermediate nodes.

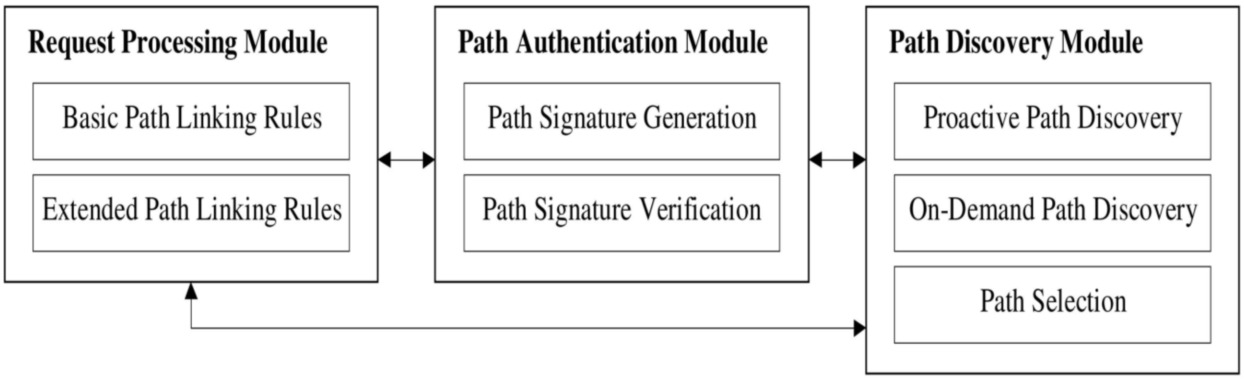
Request Processing Module
Basic Path Linking Rules
Extended Path Linking Rules
Path Authentication Module
Path Signature Generation
Path Signature Verification
Path Discovery Module
Proactive Path Discovery
On-Demand Path Discovery
Path Selection

Fig. 3. Modules of the agent-free secure interoperability framework.

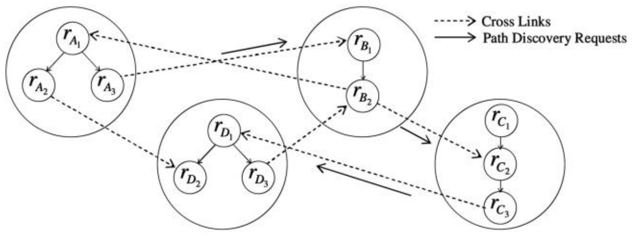

Fig. 4. Example of On-demand Path Discovery.

It works depending upon the following agreement properties:

(Agreement A.1). No result is delivered by the client unless it is committed by all database servers.

(Agreement A.2). No database server commits two different results.

(Agreement A.3). No two database servers disagree on the outcome of a result.

The effect of the Network Size (N), the Number of Transactions (T), and the Group Size (d) on the Mean Rank Difference (M) over all the peers in the network. The Mean Rank Difference (M) is calculated by averaging the rank difference of all the peers in the network for one instance of the simulation. The rank difference for a peer is the difference in the rank of the peer when the proposed identity mechanisms are used, and when it is not used. Specifically, we tried to answer the following two questions:

1. Is the mean rank difference a good predictor of the rank difference of individual nodes? What is the variation in the rank difference of individual peers in the simulation for a given value of d? Mathematically, what percentage of nodes has their rank difference equal to the mean rank difference?
2. Does the network size (N), group size (d), or the number of transactions (T) in the network impact the mean rank difference in the network? In other words, what is the expected mean rank difference for other network configurations which are different (in terms of size, group size d, or number of transactions) than the networks simulated by us?

4.1 Overall Evaluation of the System

In order to evaluate the integrated benefit of self-certification, the cryptographic protocol, we performed the experiments done for the evaluation of the cryptographic protocol, and added an availability factor, AF, to each node. The availability factor accounts for the erratic availability of the past recommenders of a given peer. AF values from 50 percent to 90 percent were randomly allocated to peers.

5. SECURITY ANALYSIS

In this section, we state some security attacks that could be performed in an agent-free collaboration environment. Moreover, we show that our secure collaboration framework is resilient to these attacks. We assume all access messages exchanged between domains are sent over secure reliable nodes.

- **Path corruption.** The access path is one of the main elements required when making access control decisions. A malicious domain may attempt to alter the access path by removing or adding entries to the current access path. The path corruption could be divided into two types of attacks, namely, path insertion and deletion.
- **Path replay attacks.** An attacker could capture a request submitted during a valid session and try to replay such a request. This attack is not possible, as we assume that for each session, a new nonce is used to authenticate the path.
- **Denial of service.** An attacker would request a role via a path that contains a loop P = {r1, r2,., rn, r1, r2, . . .} and repeat such requests infinitely to increase the path size infinitely. Such an attack can be easily dealt with by introducing a bound on the permissible path size, which is basically the path cardinality constraint mentioned in Section 4.1, and the permissible path size can be set to double the number of domains present in the collaboration. Another form of denial of service could be

performed when malicious domain floods neighbouring domains with path requests.

Violations of the restricted relation R. In this case, a malicious domain involved in a restricted access relation does not honour such relations. In such a case, this domain gives access to a user that violates the restricted access relation R. This attack is easily detected by the neighbouring domain as such role access will be recorded in the user’s access path and delivered to the mitigated space.

6. PERFORMANCE ANALYSIS

We state here simply on nice runs where no process crashes or is suspected to have crashed. In particular, our performance measure takes into account only the number of messages exchanged in a nice run of a wo-register implementation. In terms of latency, we show that our algorithm introduces an overhead of about 16 percent over the baseline unreliable algorithm (that does not offer any guarantee). This overhead is actually lower than the overhead of a 2PC algorithm, which we show is around 23 percent in our environment. This might look surprising at first glance, because our algorithm also ensures a nonblocking property of databases besides the exactly-once guarantee (2PC is blocking [4] and ensures only at-most-once request delivery). However, in contrast to 2PC, our algorithm does not induce any forced disk IO. We use the same replication scheme to ensure the client's outcome determination as we use to guarantee nonblocking. Nevertheless, we assume that application servers cannot all crash at the same time, whereas a 2PC tolerates a total crash of these servers.

Here, we consider the case where a single application server crash is tolerated. Since our algorithm requires a majority of correct application processes, three application servers are required. In our primary-backup scheme, a single backup is enough. We assume an implementation of a wo-register using an optimized consensus algorithm along the lines. Basically, in a nice run, it takes only a round trip message for the first primary to write into the register (the first consensus coordinator is the default primary application server). In terms of the latency, as viewed by the client, our algorithm introduces the same number of communication steps than a primary-backup scheme, but more than a 2PC algorithm or an unreliable baseline algorithm. The 2PC, however, introduces eager disk accesses.

7. CONCLUDING REMARKS

In this paper, we have presented an agent-free collaboration environment in which domains collaborate in making localized access control decisions. We presented a framework to enable collaboration in such an environment, where domains collaborate securely without needing a trusted mediator and without needing a global view of the collaboration environment. In our framework, the user’s access path is used to provide domains with enough information to make secure access control decisions using both basic and extended path linking rules. We also provided a path authentication scheme that ensures that the path is not tampered with, as it propagates between domains. Furthermore, we have provided an on-demand path discovery algorithm that enable domains to discover available multihop collaborations. We analysed several security attacks that could be performed and showed how our framework can easily handle such attacks.

8. ACKNOWLEDGMENTS

Our thanks to the experts who have contributed towards development of the template. This paper is an extended and revised version of a paper, with the title, *secure collaboration in a Mediator-Free Distributed Environment* appears in the Proceedings of the IEEE International Conference on Parallel and Distributed Systems 2008. The authors would like to thank Mohamed Shehab, Arif Ghafoor, and Elisa Bertino for their contributions about the structure of our protocol and to Jim Gray for sharing with us his views about the meaning of reliability in three-tier architectures. We are also very grateful to the reviewers for their helpful presentation suggestions on the earlier version of this paper.

9. REFERENCES

[1] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin, “A Calculus for Access Control in Distributed Systems, ACM Trans. Programming Languages and Systems, vol.15, no.4, pp. 706-734, Sept. 1993.

[2] M. Abadi and C. Fournet, “Access Control Based on Execution History,” Proc. 10th Ann. Network and Distributed System Symp. (NDSS), 2003.

[3] D. Bell and L. LaPadula, “Secure Computer Systems: Mathematical Foundations,” Technical Report MTR-2547, vol. 1, Mar. 1973.

[4] E. Bertino and R. Sandhu, “Database Security-Concepts, Approaches, and Challenges,” IEEE Trans. Dependable Secure Computing, vol. 2, no. 1, pp. 2-19, 2005.

[5] P. Bonatti, M. Sapino, and V. Subrahmanian, “Merging Heterogenous Security Orderings,” J. Computer Security, vol. 5,no. 1, pp. 3-29, 1997.

[6] S. Dawson, S. Qian, and P. Samarati, “Providing Security and Interoperation of Heterogeneous Systems,” Distributed Parallel Databases”, vol. 8, no. 1, pp. 119-145, 2000.

[7] A. Rowstron and P. Druschel, “Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems,” Proc. IFIP/ACM Int’l Conf. Distributed Systems Platforms (Middleware), pp. 329- 350, Nov. 2001.

[8] B.C. Ooi, C.Y. Kiau, and K. Tan, “Managing Trust in Peer-to-Peer Systems Using Reputation-Based Techniques,” Proc. Fourth Int’l Conf. Web Age Information Management, Aug. 2003.

[9] L. Liu, S. Zhang, K.D. Ryu, and P. Dasgupta, “R-Chain: A Self-Maintained Reputation Management System in p2p Networks,” Proc. 17th Int’l Conf. Parallel and Distributed Computing Systems (PDCS), Nov. 2004.

[10] R. Zhou, K. Hwang, and M. Cai, “Gossiptrust for Fast Reputation Aggregation in Peer-to-Peer Networks,” IEEE Trans. Knowledge and Data Eng., vol. 20, no. 9, pp. 1282-1295, Aug. 2008.

[11] Z. Xu, Y. He, and L. Deng, “A Multilevel Reputation System for Peer-to-Peer Networks,” Proc. Sixth Int’l Conf. Grid and Cooperative Computing (GCC ’07), pp. 67-74, 2007.

[12] M. Gupta, P. Judge, and M. Ammar, “A Reputation System for Peer-to-Peer Networks,” Proc. 13th Int’l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), 2003.

10. AUTHORS PROFILE

Ramesh Dharavath received the B.Tech degree in computer science and Engineering from KITS – Warangal (Kakatiya University) in Warangal, in 2004, and the M.Tech degree in Software Engineering from the University of Jawaharlal Nehru Technological University at Hyderabad, in 2009. He is a senior Faculty member with the designation of Associate Professor and also having affiliation with professional bodies like Indian Society of Technical Education (ISTE) and International Association of Engineers (IAENG). His research interests include Distributed systems, Database security, SQL injections, and Neural Network. He is currently focused on research in Heterogeneous Distributed Databases reliability and availability for critical systems.

Swetha Reddy Vallem received the B.Tech degree in computer science from MITS – Peddapalli (JNTU University) in 2005, and the M.Tech degree in Computer Science and Engineering from Anurag Engineering College – kodad in 2011. She is a senior Faculty member with the designation of Assistant Professor. Her research interests include Distributed systems, She is currently focused on research in Distributed Databases for critical systems.