Computing of Trust in Distributed Networks

Huafei Zhu, Bao Feng, Robert H. Deng

InfoComm Security Department, Institute for InfoComm Research.
21 Heng Mui Keng Terrace, Singapore 119613.
{huafei, baofeng, deng}@i2r.a-star.edu.sg

Abstract. In distributed networks, a target party T could be a person never meet with a source party S, therefore S may not hold any prior evaluation of trustworthiness of T. To get permit to access S, T should be somewhat trusted by S. Consequently, we should study the approach to evaluate trustworthiness of T. To attack the problem, we view individual participant in distributed networks as a node of a delegation graph G and map a delegation path from target party T to source party S in networks into an edge in the correspondent transitive closure of graph G. Based on the transitive closure property of the graph G, we decompose the problem to three related questions below:
-how to evaluate trustworthiness of participants in an edge?
-how to compute trustworthiness of participants in a path?
-how to evaluate the trustworthiness of a target participant in a transitive closure graph?
We attack the above three questions by first computing trustworthiness of participants in distributed and authenticated channel. Then we present a practical approach to evaluate trustworthiness by removing the assumption of the authenticated channel in distributed networks.

Keywords: Direct trust, recommendation trust, transitive closure, trust value

1 Introduction

THE PROBLEMS. We start our works by considering two realistic examples:

Example 1: Suppose a target party T (say, an employee of a department) is permitted to access a source participant S in distributed networks only if there is a path from $T's$ recommender N_0 (say, the director of the department) to S (access policy of S). To access the source S, T applies for N_0 as $T's$ recommender at first. If the application is approved, T then employ a special program (say PolicyMaker [7], [8] or PathServer [19]), to search a path from $N_0 \rightarrow N_1 \rightarrow \cdots \rightarrow N_k \rightarrow S$. Finally T provides the path information to S so that S is able to verify this path originating from the target participant T and the node N_0 acting as a recommender of target participant T.

Example 2: In distributed computing environments, a party T needs to verify the correctness of certificate data issued by some party S without having $S's$

public date at hand. Such kinds of events often occur in the large scale distributed networks. A prospective solution to this problem is that T can employ a special program N_0 (say PolicyMaker or PathServer), to search a path $N_0 \rightarrow N_1 \rightarrow \cdots \rightarrow N_k \rightarrow S$ that enables T to confirm the validity of this certificate.

Three interesting problems can be abstracted from the above examples:

-Problem 1: In example 1, The policy of S is that "to access the source S, T has to apply for N_0 as $T's$ recommender". The question is that why should N_0 trust T and act as a recommender of T? Equivalently, how does N_0 evaluate trustworthiness of T?

We emphasize the importance of the problem. Depending on the task which T wants party N_0 to perform, N_0 has to decide whether T is sufficiently trustworthy. Usually there is a maximum value one is willing to risk within a certain trust relationship. To determine such a maximum value, one has to estimate the degree of trust, that is a party should estimate the valuation of trustworthiness at first.

-Problem 2: Generally a path search engine (say, PathServer) needs not to be trusted. A standard approach is to employ a path finder program in nullifying Byzantine faulty participants of trust worthiness and accepting as legal query by majority. Consequently, we should provide a means of estimating the trustworthiness of paths generated by a path finder which is run by T (or by N_0, a recommender of T).

We emphasize the importance of the second problem. To verify the correctness of $S's$ certificate data, a party T without verification data at hand, should search a path containing a set of nodes $(N_1, \cdots, N_k)$. We aware that if any N_i in the path provides a false statement regarding N_{i-1}, then there is no reason to believe that a proper semantics for a source channel is reached. One way to bolster assurance is to limiting intermediate participant N_i to be trusted by N_{i-1} with some degree. Therefore a method to evaluate the trust degree of the paths is definitely needed.

-Problem 3: In example 1, a recommender N_0 should provide paths to S so that S is able to verify that any path is originating from the target participant T and the node N_0 is acting as a recommender of target participant T. In essence, this is an evaluation of $T's$ trustworthiness by S. Consequently we need a method to evaluate the trustworthiness of the target participant.

We emphasize the importance of the third problem. In distributed networks, a party T could be a person never meet with S, therefore S may not hold any prior evaluation of trustworthiness of T. To get permit to access S, T should be somewhat trusted by S. Consequently, we need a method to evaluate trustworthiness of T.

POSSIBLE SOLUTIONS: We remark that authentication in centralized computer system is simplified by the presence of a central authority that controls all participants and knows what principle can initiate requests on what participants. In distributed systems, there typically is no such a central authority for this information. As the distributed system gets larger and more diverse (e.g., peer to peer networks), the difficulty of reliability authenticating a channel can increase substantially (see [20], [21], [22], [12] for more details). To solve the problems mentioned above in the large scale distributed networks, we should make assumptions listed below:

-Assumption 1: Since there are many different certificates and all these assertions means different things, we should make a certificate in clear meanings. We therefore assume that each party in the networks create signed certificates that bind semantics(e.g., a name and an e-mail address) to a public key as that in PGP environment [24].

-Assumptions 2: Each party may hold several certificates $C_1, \cdots, C_t$, and we assume that a party A trusts $B's$ certificate C_1, does not implies A also should trust $B's$ another different certificate C_2 and so on.

-Assumption 3: The delegation in our model is one-way in the sense that a party A trusts party B does not implies that B trusts A.

Based on the above assumptions, we are able to provide possible solutions to the questions. We study models of computing of trust and then we study the mechanisms to support trust relationships in distributed networks. To support computing of trust, a new participant T decides to join into distributed networks should specify one or more parties as direct recommender(s) and the specified direct trusted recommender DT can run a path finder program to search a path from DT to the source party S so that S evaluates $T's$ trustworthiness and provide service if the trust value is lager than a predetermined threshold value (otherwise, it refuse to provide service). Since there is a path from T to its direct recommender (say N_0), and there is also a path from this direct recommender N_0 to the source participant S, we can define an edge from S to T (this is the transitive closure property of a graph G).

RELATED WORKS: The public key infrastructure, including X.509 [13], SPKI/SDSI [14] and PGP [24], is a basic yet powerful tool to manage trust relationships among networks. To apply these generic tools to evaluate trustworthiness, one should consider the subtle issues among them, more details:

-The X.509 trust model [1]: The X.509 trust model is strictly centralized and hierarchical certificate management model. Each participant must have a certificate signed by a central certification authority. Therefore, X.509 trust model does not fit for our purpose for computing of trust value in distributed environments.

-SPKI trust model [14]: The simple public-key infrastructure (SPKI) is flexible for trust management. There is one bit defining the ability to delegate in the certificate structure. We aware that there is no restriction to control the delegated certificate chains in SPKI model, therefore those problems: how to control these delegated certificates, and how can this issuer update the trust value of each delegated certificate, should be studied further. We remark that these problems are closely related to the second question mentioned above and we remark that this notice is also true for SDSI trust management model [18].

-PGP trust model [24]: PGP is remarkable model for trust management in distributed networks. The underlying assumptions are that a trust may trust other entities, may validate certificate form certificate from other entities or may trust a third party to validate certificates. However, the entities in PGP environment should be trusted by a trustor completely, therefore no mechanism supports the computing of trustworthiness in PGP setting.

Following the above foundational works, lots of beautiful works have been developed for various purposes:

-Blaze et.al ([7], [8], [9], [10]), have implemented several automated trust management system: PolicyMaker, KeyNote for specifying what a public key is authorized to do. The PolicyMaker system is an unified approach to specifying and interpreting security policies, credentials, and relationships that allows direct authorization of security-critical actions. We remark that PolicyMaker system is essentially a search engine. The KeyNote system, the successor of PolicyMaker, has the same design principle, but more is done in the trust management engine, rather than in the calling applications. It supports signature verification and special language assertion, and KeyNote assertions always returns a Boolean (authorized or not) answer.

-Reiter and Stubblebine [19], have already developed an interesting authentication model, different from PolicyMaker and KeyNote, using a path of trusted intermediaries, each enables to authenticate the next in the path. They also developed an efficient path finder program PathServer as an interesting application for searching paths to authentication in PGP environment.

-Aura's model ([3], [4]), is another interesting model to delegate certificates in distributed networks. Each certificate in the path is signed by the subject of the previous certificate and its task is to find a path of trusted intermediaries, each enables to authenticate the next in the path.

We remark that each certificate in a delegate path is completely trusted by source participant in above three models if the proposed actions are consistence with the local policy by applying the assertion predicts to the action environment. Those models do NOT provide a means of computing of trustworthiness of participants in the network dynamically.

OUR WORKS: We view individual participant in distributed networks as node of a delegation graph G and map a delegation path from target party T to source party S in networks into an edge in the correspondent transitive closure of the graph G. Based on the transitive closure property of the graph G, we study three issues related to computing of trust below:

-how to evaluate trustworthiness of participants in an edge?

-how to compute trustworthiness of participants in a path?

-how to evaluate the trustworthiness of a target participant in a transitive closure graph?

We attack the above three questions by first computing trustworthiness of participants in distributed and authenticated channel. Then we present a practical approach to evaluate trustworthiness by removing the assumption of the authenticated channel in distributed networks.

2 Mechanisms to computing of trust

MODELS OF TRUST DELEGATION STRUCTURES: To compute trust values in distributed networks, one should study the model of trust delegation structure at first. The model should capture the characteristics of propagation. We also know that if any N_i in the path provides a false statement regarding N_{i-1}, then there is no reason to believe that a proper semantics for a source channel is reached. One way to bolster assurance is to limiting intermediate participant N_i to be trusted by N_{i-1} with some degree (say, the minimum trust value should be threshold trust value accepted by S). Therefore we can view each participant N_{i-1} is a recommender of previous certificate N_i. Based on the those observations, we consider four types of trust delegation structures:

$$f_1 : RT \times DT \to DT;$$
$$f_2 : RT \times RT \to DT;$$
$$f_3 : DT \times RT \to DT;$$
$$f_4 : DT \times DT \to DT;$$

Where RT is a path of trusted intermediaries $\{N_1, N_2, \cdots, N_k\}$, each enables to authenticate the next in a path. DT is a set of direct recommenders of target parties in the network. Among them, the structure of function f_1 is only suitable for our purpose since we insist on a path originating form target T to a source S and viewing N_0 as a direct recommender of a target participant T. The function f_1 can be extended to large scope of members:

$$F(N_1, N_2, \cdots, N_k) = (RT)^k \times DT$$

We call $S \leftarrow N_k \leftarrow \cdots \leftarrow N_0 \leftarrow T$ a path from target T to the source S. T is called a target participant, S is called a source participant, N_0 is called a direct recommender of T, each N_i is called an intermediary direct recommender of N_0.

DEFINITION: A graph $G = (V, E)$ has a finite set V of vertices and a finite set $E \subseteq V \times V$ of edges. The transitive closure $G^* = (V^*, E^*)$ of a graph $G = (V, E)$ is defined to have $V^* = V$ and to have an edge (u, v) in E^* if and only if there is a path from u to v in G.

We view individual participant in distributed networks as a node of a graph G. Since there is a path from T to its direct recommender N_0, and there is also a path from this direct recommender N_0 to the source participant S, it follows that we can define a new path from S to T in the graph G. We map this newly generated path into an edge of the transitive closure of graph G, and computes the trustworthiness based on transitive closure of graph G.

We remark that several notable path search engines fit our purpose well. Balze et.al, has developed two elegant search engines: PolicyMaker and KeyNote which can be tailored for our purpose. Reiter and Stubblebine [19] have already presented an elegant enquiry protocol for making authorization decisions from large databases of certificates. Another interesting path search engine presented by Aura can also be integrated with our model (see [3], [4], [2] for more details, the basic idea in Aura's search engine is to start searches from a delegation path from both the server and the client certificates and to meet in the middle of the path.)

DIRECT TRUST DEGREE EVALUATION ALGORITHM: The concept of computing of trust is not new, it has been studied by Beth, Borcherding and Klein already [6]. We remark that our trust computing model is different from Beth, Borcherding and Klein and the mechanisms to compute trust value is solely based on the formulation $F(N_1, N_2, \cdots, N_k) = (RT)^k \times DT$.

As mentioned in the above section, when a new participant decides to join the network, it should apply for a direct trust value from an already existing participant. Consequently, we should define the initialization of direct trust degree computing algorithm. We remark that trust is a predicate however we can assign a value v to trust under condition of the output of predicate. This value is called a trust degree or a trust value.

Notions:

-$dtv_T^{N_0}$ denotes direct trust value assigned to T by N_0;

-$Cert_X$: participant $X's$ certificate that binds semantics (e.g., a name and an e-mail address) to its public key;

-$Histrory_T^{N_0}$: The record of history as N_0 is a direct recommender of T; If T is a new participant in the networks, then $Histrory_T^{N_0} \leftarrow Null$;

-$\mathrm{Pred}_X(\cdot,\cdot)$, a predicate defined over the set of certificates according to the strategy defined by X;

-$dtv_T^{N_0} \in [0,1]$ and the fact that $tr_T^{N_0} = 0$ implies N_0 distrusts T completely while the fact that $tr_T^{N_0} = 1$ implies N_0 trusts T completely.

Computing of direct trust worthiness can be performed as follows:

Input: $(Cert_{N_0}, Cert_T, Histrory_T^{N_0})$;
Computing:
  $u \leftarrow \mathrm{Pred}_{N_0}(Cert_{N_0}, Cert_T)$
  If $u = 0$, then $v \leftarrow 0$
  Else, $v \leftarrow \mathrm{Strategy}_{\mathrm{N_0}}(Cert_{N_0}, Cert_T, Histrory_T^{N_0} | u = 1)$;
Output: a value $dtv_T^{N_0} \leftarrow v$.

COMPUTING OF RECOMMENDATION TRUST VALUE. The source participant S in the distributed networks, will check the paths that was provided by a path finder, say $PathServer$ since the task to search a path from N_0 to S is very difficulty (see [19] and [7], [10] for more details). Upon receiving path information, S will compute a trust value (recommendation trust value) assigned to N_0 (or simply assign the trust value to the path finder which is run by N_0, a direct recommender of the target participant). To formulate computing of recommendation trust value, we first define two notions below:

Suppose $P_1, \cdots, P_k$ are k pathes to be provided to S. These pathes are referred to as delegation pathes. Let $N(P_i)$ be a set of recommenders $\{N_{i_1}, N_{i_2}, \cdots, N_{i_k}\}$ in the i-th path.

- $P_1, \cdots, P_k$ are called independent, denoted by $DP(P_1, \cdots, P_k)$ if

$$\forall P_i, P_j, \exists E(P_i) \cap E(P_j) = \emptyset, 1 \leq i, j \leq k$$

-$P_1, \cdots, P_k$ are called relevant, denoted by $RP(P_1, \cdots, P_k)$ if there exists i, j such that

$$N(P_i) \cap N(P_j) \neq \emptyset$$

Based on the above notions, we can define a recommendation trust value $rtv_{N_0}^S$ by computing

$$rtv_{N_0}^S = F(History_{N_0}^S, rtv_{P_1}, rtv_{P_2}, \cdots, rtv_{P_k}).$$

Where $rtv_{P_j} = \min\ \{dtv_{N_0}{}^{N_{j_1}}, \cdots, dtv_{N_{j_k}}{}^{S}\}$

Case 1: $(P_1, P_2, \cdots, P_k)$ are independent paths, the combination of these direct trust value is defined by

$$rtv_{comb} = \frac{1}{k}\sum_{i=1}^{k} rtv_{P_i}$$

The recommendation trust value is defined as:

$$rtv_{N_0}^S \leftarrow t \times rtv_{N_0}^S + (1 - t) \times rtv_{comb}$$

where t is referred to as trust factor, which is determined by S completely and $t = 0$ if $rtv_{N_0}^S$ is not in the recorded in the history.

Case 2: Suppose $(P_1, P_2, \cdots, P_k)$ are k pathes relevant paths. These paths are divided into m sets so that paths in each set are independent. These sets are denoted by $SP_1, SP_2, \cdots, SP_m$. The recommendation trust value is defined as:

$$rtv_{comb} = \frac{1}{m}\sum_{i=1}^{m} rtv_{P_{i_j}}$$

Where P_{i_j} is a path chosen at random from the set SP_i.

$$rtv_{N_0}^S \leftarrow t \times rtv_{N_0}^S + (1 - t) \times rtv_{comb}$$

where t is referred to as trust factor, which is determined by A completely and $t = 0$ if $rtv_{N_0}^S$ is not in the recorded in the history.

Note that we choose one path at random in a relevant set SP_j each as the input to compute the recommendation trust value to reduce computational complexity at $S's$ side.

RECOMMENDATION VALUE EVALUATION ALGORITHM: Finally a source participant S should compute the recommendation trustworthiness of the target participant T. Since there is a path from T to its direct recommender, and there is also a path from this direct recommender to the source participant S, it is natural if we define a path from S to T. Consequently the delegation graph is a transitive closure of a graph.

The graph $G = (V, E)$ can be maintained by a recommender or a path finder. Suppose a path-finder is a direct recommender of a target participant T, then the path-finder associates with T by an edge. Notice that a transitive closure of a graph G^* is uniquely determined by the graph G, therefore S can simply view the recommendation trust value $rtv_{N_0}^S$ as $S's$ trustworthiness of T.

EXTENSIONS: The mechanisms of computing of trust above is defined in the distributed and authenticated environments. That is the trust value computed by a recommender (either direct recommender or intermediary recommenders of a path) is authenticated by recommender itself. To compute of trust in distributed networks and public channel environment, one can apply Aura's model ([3], [4]). In Aura's model each certificate in the chain is signed by the subject of the previous certificate, and that each certificate in the chain authorizes the request and one can view the intermediate participant N_i to be trusted by N_{i-1} completely. However, in our model, we assume that each certificate in the chain authorizes the request and one can view the intermediate participant N_i to be trusted by N_{i-1} associated with trust degree. Therefore we can view each participant N_{i-1} is a recommender of previous certificate N_i with some trust degree. Since delegation graph G is authenticated and the transitive closure of a graph G^* is uniquely determined by the graph G, S can simply view the recommendation trust value $rtv_{N_0}^S$ as $S's$ trustworthiness of T and compute the correspondent trust value by the approach studied above. This is an interesting application of transitive signature scheme.

3 Conclusions

We have developed a novel model to compute trustworthiness in distributed networks based on a transitive closure of a graph and we present solutions to three questions as well.

References

1. Alfred W. Arsenault, Sean Turner. Public key infrastructure. http:// www.ietf.org/ html.charters/ pkix-charter.html.
2. Tuomas Aura: Distributed Access-Rights Managements with Delegations Certificates. Secure Internet Programming 1999: 211-235.
3. Tuomas Aura: Fast Access Control Decisions from Delegation Certificate Databases. ACISP 1998: 284-295.
4. Tuomas Aura: On the Structure of Delegation Networks. CSFW 1998: 14-26.
5. Tuomas Aura, Silja Maki: Towards a Survivable Security Architecture for Ad-Hoc Networks. Security Protocols Workshop 2001: 63-73.
6. Thomas Beth, Malte Borcherding, Birgit Klein: Valuation of Trust in Open Networks. ESORICS 1994: 3-18 40.
7. Matt Blaze, Joan Feigenbaum, Angelos D. Keromytis: KeyNote: Trust Management for Public-Key Infrastructures. Security Protocols Workshop 1998: 59-63.
8. Matt Blaze, Joan Feigenbaum, Angelos D. Keromytis: The Role of Trust Management in Distributed Systems Security. Secure Internet Programming 1999: 185-210.
9. Matt Blaze, Joan Feigenbaum, Martin Strauss: Compliance Checking in the PolicyMaker Trust Management System. Financial Cryptography 1998: 254-274.
10. Matt Blaze, John Ioannidis, Angelos D. Keromytis: Trust management for IPsec. TISSEC 5(2): 95-118 (2002).
11. Naranker Dulay, Nicodemos Damianou, Emil Lupu, Morris Sloman: A Policy Language for the Management of Distributed Agents. AOSE 2001: 84-100
12. USDoD, Trusted Computer Systems Evaluation criteria. U.S Department of Defense, 1985.
13. Carl M. Ellison: The nature of a useable PKI. Computer Networks 31(8): 823-830 (1999).
14. C. EllisonB, Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. SPKI: http: //www.ietf.org/ rfc/ rfc2693.txt.
15. Carl M. Ellison, Bruce Schneir: Risks of PKI: Secure Email. CACM 43(1): 160 (2000).
16. Carl M. Ellison, Bruce Schneier: Risks of PKI: E-Commerce. CACM 43(2): 152 (2000).
17. Silvio Micali, Ronald L. Rivest: Transitive Signature Schemes. CT-RSA 2002: 236-243
18. Ron Rivest and Butler Lampson. SDSI: A Simple Distributed Security Infrastructure, http: // theory.lcs.mit.edu / cis/sdsi.html.
19. M. Reiter and S. Stubblebine. Resilient authentication using path independence. IEEE Transactions on computers, Vol.47, No.12, December 1998.
20. Morris Sloman, Jorge Lobo, Emil Lupu: Policies for Distributed Systems and Networks, International Workshop, POLICY 2001 Bristol, UK, January 29-31, 2001, Proceedings. Springer 2001.
21. Tyrone Grandison, Morris Sloman. As survey of Trust in Internet Applications. IEEE Communication Survey and Tutorials, Fourth Quarter 2000, http://www.comsoc.org/ pubs/ surveys

22. Tyrone Grandison, Morris Sloman: Specifying and Analyzing Trust for Internet Applications. IEEE 2002: 145-157
23. Raphael Yahalom, Birgit Klein, Thomas Beth: Trust-Based Navigation in Distribution Systems. Computing Systems 7(1): 45-73.
24. Phil Zimmermann. Pretty Good Privacy (PGP), PGP User's Guide, MIT, October, 1994.