Network and Trust Model for Dynamic Federation

Yang Xiang, John Alan Kennedy, Matthias Egger
Rechenzentrum Garching
Max-Planck-Society
Garching, Germany
{yang.xiang, jkennedy, matthias.egger}@rzg.mpg.de

Harald Richter
Department of Informatics
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
hri@tu-clausthal.de

***Abstract*—Most existing approaches to identity federation are based on static relationships. This leads to problems with scalability and deployment in real-time environment such as mobile networks. This paper introduces an underlying network and trust model for dynamic federation. We present a modified Dijkstra algorithm to calculate the trust value and apply a distributed reputation calculated based on the PageRank algorithm from Google to each entity in order to increase the attack resistance of the system.**

***Keywords*-AAI; dynamic; federation; trust; reputation;**

I. INTRODUCTION

Existing solutions for authentication and authorization infrastructure (AAI) like Shibboleth [1] are generally based on static federation. In a static federation, relationships among identity providers (IdPs) and Service providers (SPs) are manually pre-configured in their meta data. The question of whether an entity can trust another depends on if they can find each other in the meta data, thus this question can not be answered in a dynamic manner due to the static nature of the meta data.

The static structure of AAI leads to problems with scalability and interoperability for the following reasons: every new relationship between any two entities must be added manually as such a static federation can not be quickly and easily expanded to an AAI with hundreds or even thousands of IdPs and SPs. Furthermore it is difficult to connect two or more independent federations beyond their borders to form a con-federation because an entity of a certain federation does not know and hence does not trust entities from other federations. Finally, a static AAI can not be deployed in a real-time environment like a mobile network where users access the services of any provider at any time.

Hence we introduce a concept of a dynamic federation, in which the IdPs and SPs will be regarded as peers of a trusted network that evolves over time. A trust relationship between two entities is regarded as a network connection. In such a dynamic federation, an SP does not need to know an IdP beforehand. A trust relationship will be created on demand and the trust value, namely how much an IdP can be trusted will be determined on the fly.

We consider the following use case as illustrated in Figure 1:

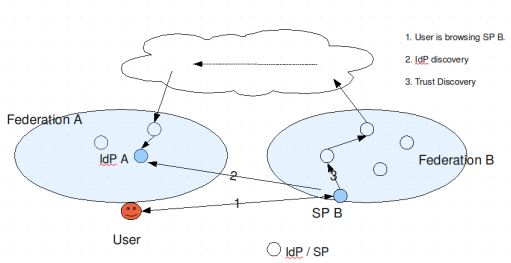

Figure 1. Use case of dynamic federation

1) A User is registered with an IdP A of federation A.
2) He is browsing SP B belonging to federation B.
3) SP B detects that IdP A is the preferred IdP for the browsing user by using a discovery service.
4) SP B gets the meta data of IdP A in order to determine if IdP A can be trusted and to what extent.
5) With a positive result SP B requests IdP A to authenticate the user.
6) IdP A authenticates the user and returns an assertion to SP B.
7) SP B authorizes the user to access the requested service.

In this case SP B does not need to know IdP A beforehand. The trust relationship between SP B and IdP A is created on the fly.

For the dynamic federation to be well defined, we give the following definitions:

Definition 1.1 (entity): An entity is an IdP or SP in a dynamic federation. When the dynamic federation is regarded as a network, we call an IdP or SP a peer of the network. Sometimes we also use the term *vertex* or *node* from graph theory.

Definition 1.2 (IdP discovery): IdP discovery is the process, by which an SP detects the preferred IdP of a user. The result of this process is the endpoint of the IdP.

Definition 1.3 (Trust discovery): Trust discovery is the process, by which an SP determines whether an unknown IdP can be trusted and vice versa. The result of this process is a binary decision or a trust value in the range of (0, 1).

Definition 1.4 (Trust value): The trust value is a subjec-

 ISBN: 978-1-61208-101-4

tive quantification of how much an entity can trust another. The trust value depends on the path from truster to trustee, thus it is not a global value.

Definition 1.5 (Reputation value): The reputation value of an entity is a quantity derived from the underlying federation, which is globally visible to all members of the federation. An entity has in general different reputation values as seen by others, we call this personalized reputation. The reputation of an entity depends on the number of its incoming edges in the connection graph and on the reputation value propagated from its neighbors along these edges.

The topic *IdP discovery* is beyond the scope of this paper, in which we firstly focus on the issue *trust discovery*. If we treat a federation consisting of IdPs and SPs as a graph where the IdPs and SPs are vertices and the trust relationship are edges, we can reduce the problem of trust discovery to a question of pathfinding. In the following, we will compare different network models and analyze different pathfinding strategies.

II. Related work

OASIS defined two mechanisms for dynamic metadata publication and resolution [10]. However, it refers to the topic of IdP discovery rather than trust discovery. The trust model of the OASIS mechanisms still relies on X.509. The same drawback is encountered in approaches to dynamic SAML and dynamic metadata exchange [13]. To enable SAML for dynamic federations a generic solution with a SAML extension and a dynamic trust list was introduced [5]. However, there are no details regarding the SAML extension and dynamic trust list described in the paper. In addition, there was an approach to build dynamic federations in Grids [3]. This approach is based on WS-Trust 1.1 [4] and WS-Federation 1.0 [6] and hence, is not SAML-compatible.

III. Strategies of pathfinding

A. P2P networks

When we regard IdPs and SPs as peers of a network, they essentially form a type of peer-to-peer (P2P) trust network, because each entity provides and consumes trust information. In recent years, P2P networks were studied intensively. P2P networks have evolved in terms of their architecture from unstructured networks like Gnutella and Napster [17] to structured networks using distributed hash tables (DHT) like Chord [20]. Structured networks are more efficient and scalable, therefore almost all modern P2P networks belong to this type.

However, all of these systems mentioned above focus on the issue of how to find certain data in a P2P network within an acceptable bound such as in $O(logN)$ time and how to minimize the join / leave overhead of peers. Thus, they do not care whether the path to the targeted node is the shortest or the optimal one. In a network of trust an entity can not be trusted solely because of the fact that it is reachable in the network; Rather we need to calculate the trust value for a trust decision and / or for further authorization decisions. This means, we need to know not only whether a node can be found in a network but also the level of trust we associate with this node.

B. Classic routing protocols

Routing protocols in contrast to P2P networks focus on finding the optimal path between nodes. Distance vector routing protocols like RIP [14] are based on Bellman-Ford algorithms [7]. Within distance vector protocols, a route is defined as a vector with length and direction where the vector length is a generalization of the distance between source and destination. The advantage of distance vector routing protocols is their simplicity. They are easy to implement, configure and maintain. However, to avoid routing loops a maximum hop count and a hold-down-timer are introduced, this leads to major problems with scalability and convergence in the routing iterations (count-to-infinity problem).

Link-State Routing Protocols like OSPF [18] are based on the concept that routers flood information about the state of their adjacent neighbors, called link state to all nodes in a sub-network. Hence, each router will have a map of the entire sub-network after a certain time. Generally, a Dijkstra algorithm [12] is used to calculate the shortest path from the source node to a destination in the sub-network. Link-state routing protocols support complex topologies of large sub-networks and scale well. The Dijkstra algorithm used in link-state protocols can be implemented with less run time than the Bellman-Ford algorithm used in distance vector protocols. Because routers using the link-state protocol have a map of the entire sub-network, routing loops are rarely encountered. Generally, link-state protocols converge much more quickly than distance vector protocols and have no limitation on hop count. The trade-offs of link-state protocols are the expense of implementation and maintenance and the high requirement of CPU power and memory.

IV. The network model for dynamic federation

Following the above analysis we can draw the conclusion that routing protocols are better suited to forming a dynamic federation for AAI than P2P network models. This is because both routing protocols and dynamic federation address the same problem, i.e., how to find the best path from a given source node to a destination node in the network.

We decided to create the network model for dynamic federation by using a link-state model similar to OSPF, which consists of the following components:

- **Network Hierarchy**

Like OSPF the network of entities shall be divided into areas to reduce traffic overhead. For example, each federation of

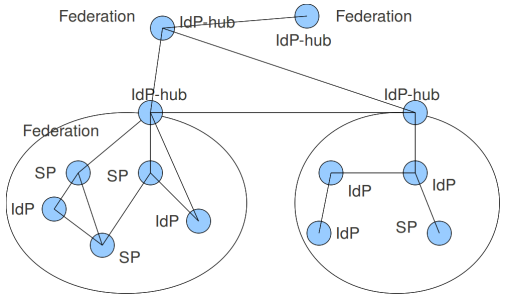

Figure 2. Example structure of dynamic federation

a con-federation can be treated as one area while the con-federation can be treated as the entire network. Each federation has an interface to the outside, which is responsible for inter-federation communication and only this interface is visible from outside. We call this interface the IdP-hub.

As Figure 2 shows, there are two levels in the hierarchy: IdP/SP and IdP-hub where the IdP-hub is considered to be pre-configured.

- **The link-state database**

We borrow the notion of LSA(Link State Advertisement) from OSPF. Each entity describes the link-state of its neighbors with LSAs and sends them to other entities in the network. Actually, an LSA is a list of IDs of all manually linked neighbors. When entity A has B's metadata, then A trusts B and B is an adjacent neighbor of A.

On the con-federation level, the IdP-hub provides federation information by sending federation-LSAs to other IdP-hubs in the con-federation to describe its federation to the outside. The federation-LSA contains at least the Id of its federation and the end location of the IdP-hub who is sending this LSA.

- **Trust table**

The trust table contains the metadata of all entities, it is effectively a metadata repository. In the trust table, all entities of a federation are stored with an entity-ID, a calculated trust value and the type of the entity, which is an IdP or SP, as Table I depicts.

Entity-ID	Value	Type
A	0.5	idp
B	0.8	sp
C	0.7	idp

Table I
EXAMPLE TRUST TABLE OF AN IDP OR SP

The trust table of an IdP-hub has an additional part, which contains entries for inter-federation relations. Here, the IdP-hubs of all other federations are stored. Each IdP-hub has an ID, which is its federation ID, and a calculated trust value. This value reflects the trust values between IdP-hubs

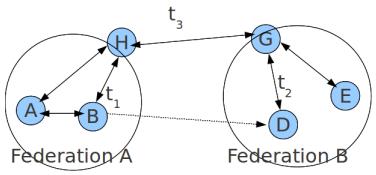

Figure 3. Calculation of inter-federation trust value

as shown with an example in Table II, where F1 and F2 are idp-hubs of other federations.

Entity-ID	Value	Type
A	0.5	idp
B	0.8	sp
C	0.7	idp
F1	0.3	idp-hub
F2	0.5	idp-hub

Table II
EXAMPLE TRUST TABLE OF AN IDP-HUB, NEEDED FOR INTER-FEDERATION RELATIONS

The trust table of all entities except IdP-hubs can be kept very small because the scope is limited to a single federation. According to a survey of the identity federations listed at the web site of Shibboleth [2] the size of the federations varies between 6 and 992 while the average number of entities is 205. When an SP receives a request from a user, it will first check his IdP in its own trust table. If there is no entry found, it will ask the IdP-hub responsible for its federation. The IdP-hub will check the Id of the destination federation in the part of its trust table dedicated for inter-federation relations. If there is an entry for the IdP-hub of the remote federation it will request the trust value of the respective IdP and compose the final trust value t as follows:

$$t = t_1 * t_2 * t_3$$

where t_1 is the trust value of the link between the IdP-hub and the SP, t_2 is the trust value of the link between the two IdP-hubs and t_3 is the trust value of the link between the remote IdP-hub and the respective IdP. Finally, the IdP-hub will send this composed trust value t back to the requesting SP. Figure 3 illustrate the calculation of interfederation trust value where the entity B is the SP mentioned above, the entity D is the respective IdP and the entity H and G are IdP-hubs of each respective federation.

- **Network protocol**

The SAML Assertion Query and Request Protocol [9] is used to exchange messages between entities. In doing this, HTTP POST Binding [8] is used to encapsulate SAML messages into a HTTP envelope. For the synchronization of the linkstate database and neighbor discovery an appropriate protocol named DYNFED similar to the OSPF protocol is under our development. Since the DYNFED protocol runs

 ISBN: 978-1-61208-101-4

on top of the SAML Assertion Query and Request Protocol, we get a 3-layer protocol structure as shown below:

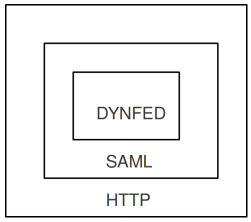

Figure 4. 3-layer protocol structure

V. THE TRUST MODEL

A. Trust value calculation

We treat a federation as a directed graph $G(V, E)$ and define a weight function t: E $\rightarrow$ R by mapping edges to real-valued weights called trust values. We denote this as $t(v_i, v_j)$, where v_i and v_j are adjacent nodes. We follow the steps below to calculate the trust value between any two entities v_0 and v_n:

1) First we start with pre-defined values for the trust relation between any two neighboring entities v_i and v_j:

$$t(v_i, v_j) = \begin{cases} 1 & \text{if } v_i \text{ can verify } v_j\text{'s certificate with the certificate of a common certification authority (CA);} \\ 0.5 & \text{if } v_j\text{'s certifcate can be verified with the certificate of a trustworthy CA;} \\ 0.1 & \text{if } v_j\text{'s certificate is self-signed;} \end{cases} \quad (1)$$

The procedure is as follows: if a certificate can be verified by a trustworthy CA, then the authenticity of the certificate can be generally trusted. If the certificate can be verified by a common CA, then both entities normally belong to the same organization, and not only the authenticity of the certificate but also the certificate owner has a higher trust value.

2) The trust value of a path $p = (v_0, v_1, ..., v_n)$ between non-adjacent v_0 and v_n is the product of the trust values of its constituent edges:

$$t(p) = \prod_{i=1}^{n} t(v_{i-1}, v_i) * d \quad \text{where } 0 < d \leq 1 \quad (2)$$

Here, we use a constant d to dampen the trust value. The reason for this damping factor is as follows: second-hand evidence can not be considered as reliable as first-hand evidence. Every time an entity passes its trust value to its neighbor, i.e., the first-hand evidence, to the next entity along the path, the degree of this trust value shall be reduced. Obviously, we have $\lim_{n\to\infty} t(p) = 0$.

3) For the effective trust value in case of multiple paths between two entities v_0 and v_n we have:

$$\hat{t}(p) = \begin{cases} max(t(p)), & \text{if there is at least one path from } v_0 \text{ to } v_n, \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

Finally, based on the calculated trust value $\hat{t}(p)$ the node v_0 can make a binary decision with respect to a configurable threshold if node v_n can be trusted or not.

In order to calculate $max(t(p))$ we modify the Dijkstra algorithm [11] to find the path p from source v_0 to any vertex v_n with the largest trust value as shown in algorithm 1:

Algorithm 1 Modified Dijkstra's algorithm

function modified_Dijkstra (G, v_0)

//$G(V, E)$ is a directed graph and v_0 is the source vertex

for each vertex $(v_i, t(v_i)) \in V$ //V is the vertex set of the input graph G

$t(v_i) := 0$; //set the trust value of vertex v_i to 0

end

$t(v_0) := 1$; //set the trust value of the source vertex to 1

$S := \emptyset$; //S is an empty set;

while V is not empty

take vertex $(v_i, t(v_i))$ from V, whereas its trust value $t(v_i)$ is the largest among all vertices in V;

if $t(v_i) = 0$

break; //all remaining vertices are inaccessible from v_0

$S := S \cup \{(v_i, t(v_i))\}$; //add v_i to S

$V := V \setminus \{(v_i, t(v_i))\}$; //remove v_i from V

for each vertex $v_j \in Adj(v_i)$ //v_j is neighbor of v_i

if $t(v_j) < t(v_i) * t(v_i, v_j) * d$ //$t(v_i, v_j)$ is the trust value of edge $e : v_i \rightarrow v_j$

$t(v_j) := t(v_i) * t(v_i, v_j) * d$;

else

do nothing;

end

end

return S;

With this algorithm we firstly assign a trust value of zero to each vertex v_n of the directed graph G except of the source vertex v_0. The algorithm returns a set S, which contains all vertices,

which can be reached from v_0, together with their corresponding trust values. After termination of the algorithm, the trust value $t(v_n)$ is equal to the maximum trust value $\hat{t}(p)$ from v_0 to v_n.

B. Attack resistance

An attack on a public key-based system of an identity federation means that some arbitrary faked target is accepted by the entire system. Two different types of attacks, namely node attack and edge attack are defined by Levien et al. in [16]. A node attack corresponds to stealing the secret keys of the victim and gaining total control of it. An edge attack corresponds to cheating the administrator of another entity to accept the attacker's certificate so that a trust relationship between the attacker's entity and the target entity can be established. An edge attack is easier to achieve than a node attack. If successful, the attacker can create many "bad" entities and deceives administrators of other IdPs or SPs into accepting his certificate and classifying it as trustworthy. Unfortunately, no trust metrics exists that can protect against node attacks efficiently as illustrated by Levien et al. [16]. Furthermore, the trust metric "shortest path" is also unable to protect against edge attacks. Thus we need a new measure to increase the attack resistance of our trust model. Levien discovered that the PageRank algorithm [19] is attack-resistant and can be used to protect against edge attacks [15]. While Google uses the PageRank algorithm to compute ranking of web pages we can use it to determine the reputation of entities by defining the computed PageRank as the reputation of each entity. Both the world wide web and an identity federation can be treated as a directed graph, hence, the PageRank algorithm can be applied to both systems. If the reputation of bad entities created by the attacker is considerably lower than the reputation of good entities, then we can easily detect those bad entities and exclude them from the system. This is why we use PageRank for our dynamic federation.

We give a recap of the PageRank algorithm, which is defined in [19] as follows:

$$R(u) = c * \sum_{v \in B_u} \frac{R(v)}{N_v} + c * E(u) \tag{4}$$

where $R(u)$ denotes the PageRank of the web page u, B_u is the set of all web pages that point to u and N_v is the number of all outgoing links from the web page v. The factor c is used for normalization, so that the total rank of all web pages is constant. Furthermore, the PageRank of a web page can be treated as the probability that a random surfer lands on that page after lots of clicks. Because of the normalization it holds $\sum_u R(u) = 1$. The second part of equation (4), E is an initial vector over all web pages and represents the distribution of the probability that the surfer gets bored after several clicks and switches to a random page. One of the choices of E presented in [19] is a uniform distribution with $\sum_u E(u) = 0.15$, however, the authors noticed that a uniform distribution of E leads to a problem: some web pages with many related links like copyright warning, disclaimers and mailing list archives receive an overly high ranking. Thus, they introduced the "personalized" PageRank by selecting a distribution that E consists of only one single web page or several trusted web pages. By this means, the respective web pages get the highest PageRank value followed by their immediately linked pages. It means that the "personalized" PageRank prefers the chosen web pages considerably.

Now, we replace the web pages with IdPs and SPs as well as the personalized PageRank with the reputation value. We can then calculate a personalized reputation value for each IdP or SP. Like web pages, the more incoming edges an IdP or SP has, the higher its reputation value is. However, with a uniform distribution of E the reputation calculation according to the PageRank algorithm is not yet attack-resistant. An attacker can just generate many faked entities and interlink them with each other. Because of the uniform distribution of E, each entity regardless of whether the entity is trustworthy or not has the same contribution to increase the reputation of the faked entity. In contrast, by choosing the distribution vector E consisting of only a single entity, i.e., our own entity, we make the reputation calculation resistant against edge-attack for the respective entity. In this way, the entities generated by the attacker do not contribute to increase the reputation value because their probabilities in E are always equal to zero and hence are useless. Now, we can calculate the personalized reputation of arbitrary entity u regarding entity v_i (i.e., the reputation of u as seen by v_i) by defining the initial vector $E(u)$ from equation (4) as follows:

$$c * E(u) \quad = \quad \begin{cases} 1-c, & \text{if } u = v_i \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

It means the vector $E(u)$ contains only the entity v_i. The exact value of c must be determined through simulations and experiments. In the original PageRank paper [19] c was set to 0.85.

Therefore each IdP or SP can use equation (4) and (5) to compute the personalized reputation for each other entity in the network. The fewer inbound edges an entity has, the lower its reputation is. Furthermore, the further an entity is located away from v_0, the lower its reputation is. With the personalized reputation it makes no sense for an attacker to create many entities pointing to his entity because its reputation will not get increased by this. Instead, he must create links to trusted entities, i.e., he must fake edges, the more the better. However, because the number of faked edges is proportional to the effort of an attacker, the introduction

 ISBN: 978-1-61208-101-4

of reputation can significantly increase the system resistance against edge attacks.

Thus, we can combine the concept of reputation with the concept of trust values and extend equation (3) as follows:

$$\hat{t}(p) = \begin{cases} max(t(p)) * R_{v_0}(v_n), & \text{if there is at least one path from } v_0 \text{ to } v_n, \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

where $R_{v_0}(v_n)$ is the reputation value of node v_n as seen by v_0.

VI. Conclusions

Approaches for AAI like Shibboleth bring many benefits for the solution of authentication and authorization and are widely-used today. Nevertheless, Shibboleth federations are based on static relationships, which do not scale well and can not be deployed in dynamic situations like mobile networks. With this issue in mind, we have developed a network model to build identity federations and con-federations in a dynamic manner. Similar to a routing protocol, the trust value of the shortest path between two entities is calculated and stored in a trust table. Furthermore, a reputation value is also calculated for each entity for the sake of attack resistance. Based on the trust value an entity is able to decide in real-time if another querying entity is trustworthy without having to know it a priori as such static meta data are not required.

In order to realize and test our network model, we aim to implement our design by extending SAML to bear and send trust values as well as reputation values and define protocols to synchronize the link-state database between entities. In the future, the design and realization of the protocol DYNFED, which was mentioned in Section IV will be completed. An implementation of the network model will be based on the framework of Shibboleth and a trust module for Shibboleth IdPs and SPs will be developed.

References

[1] Shibboleth System. URL `http://shibboleth.internet2. edu/ [2010.06.16]`.

[2] Shibboleth federations. URL `https://spaces.internet2.edu/display/SHIB/ShibbolethFederations [2010.07.05]`.

[3] M. Ahsant, M. Surridge, T. Leonard, A. Krishna, and O. Mulmo. Dynamic trust federation in grids. *Trust Management*, pages 3–18, 2006.

[4] S. Anderson, J. Bohren, T. Boubez, M. Chanliau, G. Della-Libera, B. Dixon, P. Garg, M. Gudgin, P. Hallam-Baker, M. Hondo, et al. Web services trust language (ws-trust). *Public draft release, Actional Corporation, BEA Systems, Computer Associates International, International Business Machines Corporation, Layer*, 7.

[5] P. Arias Cabarcos, F. Almenárez Mendoza, A. Marín-López, and D. Díaz-Sánchez. Enabling SAML for Dynamic Identity Federation Management. *Wireless and Mobile Networking*, pages 173–184, 2009.

[6] S. Bajaj, G. Della-Libera, B. Dixon, M. Dusche, M. Hondo, M. Hur, C. Kaler, H. Lockhart, H. Maruyama, A. Nadalin, et al. Web Services Federation Language (WS-Federation). *BEA, IBM, Microsoft, RSA Security, Verisign*, 2003.

[7] D. P. Bertsekas and R. G. Gallaher. *Data Networks*. Prentice Hall, 1992.

[8] S. Cantor, F. Hirsch, J. Kemp, R. Philpott, and E. Maler. Bindings for the OASIS Security Assertion Markup Language (SAML) V2. 0, 2005.

[9] S. Cantor, J. Kemp, R. Philpott, and E. Maler. Assertions and protocols for the oasis security assertion markup language (saml) v2. 0, 2005.

[10] S. Cantor, I.J. Moreh, S.R. Philpott, and E. Maler. Metadata for the OASIS Security Assertion Markup Language (SAML) V2. 0, 2005.

[11] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to algorithms*. The MIT press, 2001.

[12] EW Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[13] P. Harding, L. Johansson, and N. Klingenstein. Dynamic Security Assertion Markup Language: Simplifying Single Sign-On. *IEEE Security and Privacy*, 6 (2):83–85, 2008.

[14] C. Hendrik. *Routing Information Protocol, RFC1058*. The Internet Society, 1988. URL `http://tools.ietf.org/html/rfc1058 [2010.02.08]`.

[15] R. Levien. Attack resistant trust metrics. 2003. URL `http://www.levien.com/thesis/compact.pdf [2010.02.09]`.

[16] R. Levien and A. Aiken. Attack-resistant trust metrics for public key certification. In *Proceedings of the 7th USENIX Security Symposium*, pages 229–242, 1998.

[17] D.S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-peer computing. Technical report, HP Laboratories Palo Alto , HPL-2002-57 (R.1), 2003.

[18] J. Moy. RFC2328: OSPF Version 2. *RFC Editor United States*, 1998.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford InfoLab, 1998.

[20] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM'01*, 2001. URL `http://pdos.lcs.mit.edu/chord/ [2010.02.08]`.

 ISBN: 978-1-61208-101-4