On the Impact of DoS Attacks on Secure Service Discovery

Slim Trabelsi[1], Guillaume Urvoy-Keller[2], and Yves Roudier[2]

[1]SAP Labs France - 805, avenue du Dr. Maurice Donat, 06250 Mougins, France
[2]Institut Eurecom – 2229, route des Crêtes, 06904 Sophia-Antipolis, France

***Abstract*— Service discovery, an essential building block of nomadic and ubiquitous computing applications, needs to be secured to be effectively deployed. Centralized and decentralized approaches have been proposed to this end. This paper analyzes the application layer secure matching function using a Markovian performance model in order to analyze the effects of DoS attacks in centralized and decentralized service discovery systems and to assess anti-clogging mechanisms as a protection against such attacks. This study highlights the determinant parameters that should be evaluated for ensuring a scalable and secure service discovery deployment.**

***Index Terms*— Service Discovery, Performance, Model, Attacks, Security**

I. Introduction

The deployment of network centric ubiquitous computing systems, as envisioned by [1] for instance and the trend towards Service Oriented Architectures will undoubtedly generalize the need for discovery mechanisms as essential components for locating ambient and location-based services. Service discovery in a network can be implemented in two manners, first using a decentralized architecture relying on point to point (broadcast) or point to multipoint (multicast) communication, and second using a centralized architecture based on an identified registry relied upon by users and servers to facilitate discovery request matching. The choice of the appropriate architecture to enable an efficient service discovery highly depends on the deployment environment (LAN, wireless or ad-hoc communications, Internet, VPN, etc.) and on parameters like the expected number of users and services, the type and amount of resources available (CPU, memory …), and the power consumption. The performance of discovery mechanisms has already been studied through simulation. [2] and [3] present an evaluation of the performance of post-query discovery strategies in ad-hoc networks in which the authors test five strategies with the DSR and DSDV routing protocols. [4] introduces a service discovery performance model that makes it possible to predict discovery service failure and overloading in real time. This work presents simulation results that suggest that a decentralized architecture yields better robustness than a centralized one. These studies aim at getting a better understanding of the phenomena observed during discovery like message loss, faults, delays, or saturation, so as to select the most efficient service discovery mechanism for a given application. Numerous service discovery standards like WS-Discovery, Jini, UPnP, SLP, or UDDI have also been proposed in recent years, even though their performance has not been assessed analytically to our knowledge. Security in these standards is usually limited to recommendations about classical message authentication and integrity protection, thereby implicitly restricting discovery to known services. This approach falls short for taking into account the increasing use of discovery in open ubiquitous computing scenarios with numerous new threats [5] [6]. We presented security mechanisms to deal with such issues in [7] and [8]. This paper introduces Markovian models that aim at assessing the impact at the application level of introducing security mechanisms, for both centralized and decentralized service discovery. Focusing on the application level, i.e., neglecting network artifacts such as delay or losses enables us to delineate network effects from the impact of security mechanisms in terms of processing overhead for the nodes in the system. We also present two performance models to analyze the effects of DoS attacks in centralized and decentralized service discovery systems and to assess anti-clogging mechanisms as a protection against such attacks.

II. Modeling Secure Service Discovery

We essentially focus on Service Oriented Architectures (SOA), which introduce a loosely coupled interaction model that serves as a basis to define protocols and procedures to interconnect different application systems or software components. SOA mainly consists of services, which are software wrapped components providing elaborate functions (e.g., database access, data processing, business logic…), and of clients, which are requesting such services through the exchange of messages. These two types of players rely on a standardized interface to communicate but do not necessarily

share the same implementation platforms (programming language or OS). With the emergence of new dynamic networks, discovery techniques are being adapted in order to cope with the pervasive deployment of services onto this new infrastructure. This in particular stresses the need to locate and combine services to achieve a given task in an unknown environment. In this respect, service discovery is evolving from a simple brokering mechanism to a central composition component, with increasingly demanding security requirements, and whose performance should be better understood.

A. Service Discovery Basis

The main players of the discovery phase are: the service requester (client), which can be a human user or software and the service provider (server), which represents the entity providing one or multiple services that can be accessed by the clients.

1) Centralized Discovery

Centralized discovery approaches rely on a registry which plays the role of yellow pages, and which clients can refer to. The registry (or repository) is a database containing descriptions and references to some available services. Servers publish their services by contacting a registry, while clients discover published services by requesting a registry. A service advertises its capabilities (a set of attributes describing the service) to the registry, which will store them for a certain amount of time. A client contacts the registry to find a service by sending a request containing service preferences, which the registry tries to match with the most suitable provider found from the stored advertisements. In that approach, registries have to be considered by the services and the clients as a third trusted party.

2) Decentralized Discovery

Limiting service discovery to registry supported architecture that many standards SOA based services have adopted in their implementations is reductive in terms of network architecture and equipments (e.g., need to deploy specific equipments like registries). An alternative approach to service discovery exists that relies on peer to peer advertisements between services and clients (point to point and point to multipoint). In such an approach, clients discover services by broadcasting their requests to their neighborhood, and if one of the neighbors features the requested service, it will directly respond; the neighbor may otherwise forward a request to its own neighborhood. This mechanism is used for instance by the P2P-based Web Service Discovery system (PWSD) [9], which relies on the Chord P2P protocol to perform the service discovery over the internet.

3) Secure Service Discovery

The first approach to securing service discovery is to rely on an infrastructure for establishing the trustworthiness of clients and services. In the work by Zhu et al. [10], each participant to the discovery protocol is located behind a trusted proxy that sets up trust relationships through key exchanges with other proxies. [12] suggests instead the use of a central entity that combines the roles of a Certificate Authority and registry, and help clients and servers to set up a trust relationship and established secure channels between each another. Concerning privacy Zhu et al [11] proposed a Bloom filter based matching aims at hiding private information related to client and services that could be exchanged during a service discovery process. These solutions are not adapted to decentralized architectures and focus more on servers than client security, contrary to [7] and [8] on which we focus. We detail those solutions along with their model hereafter.

B. Centralized Discovery Model

1) Security Description

Our security solution designed for a centralized configuration relies on security policies provided by clients and services. Registries have to be considered by services and clients as a trusted third party whose role is no more limited to a basic matchmaker, but it evolves to a security guarantor. In this configuration [7], clients and services first establish a secure connection (e.g., SSL) with the registry to protect the confidentiality of the exchanged messages. Servers can restrict the discovery of their services to only certified users by specifying a security discovery policy to be enforced by the registry. Clients are also able to restrict the matching scope to some certified services by specifying a security discovery policy also enforced by the trusted registry. Both clients and servers have to provide credentials issued by a known authority that can be used by the registry to authenticate them during the policy verification phase. More details on this architecture can be found in [7].

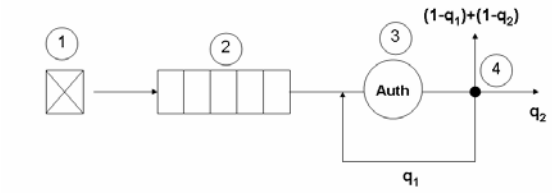

Fig. 1. Centralized Model

2) Model

Figure 1 presents the processing phase of a secure client service request at the registry for a centralized configuration in case of a single-threaded registry, i.e., a sole thread is in charge of all processing steps.

The discovery process consists of these steps:

1. Client service discovery requests arrival: requests are assumed to be generated according to an arrival process with a rate λ.

2. Buffering: The registry can temporarily store the requests to be processed by the central unit. Messages are served in a FIFO manner.

3. Request processing: the registry first matches a client request with the service profiles available locally. The matched service will be authenticated in order to verify its compliance with the security policy provided by the client. If the verification is successful, the registry also has to further

authenticate the client in order to verify its compliance with the security policy provided by the service. The corresponding service time is a random variable with a mean value $1/\mu$.

4. Probabilistic decisions (acceptance or rejection): q1 is the probability that a service matches with a client request and also be compliant with its policy. q_2 is the probability that a client be compliant with this service policy.

C. Decentralized Discovery Model

1) Security Description

The security solution proposed in [8] for a decentralized configuration relies on a particular usage of the Identity Based Encryption mechanism [13]. The server advertises its service capabilities by multicasting its profile to the entire network. Clients can cache service information or ask for a specific service by multicasting their requests to all available servers and only concerned services will respond to him. With no possible reliance on any third party in ad-hoc configurations, clients and servers now must assure their own secure service discovery using a particular encryption scheme. In [8], Attribute Based Encryption (ABE) [14] was adopted to make it possible for a server to encrypt its service description according to the restrictions imposed to users (i.e., only a class of users holding corresponding private keys will be able to access to services information). Clients also can use the same encryption mechanism in order to protect their request messages from unauthorized servers (i.e., only a class of servers is able to decrypt the request).

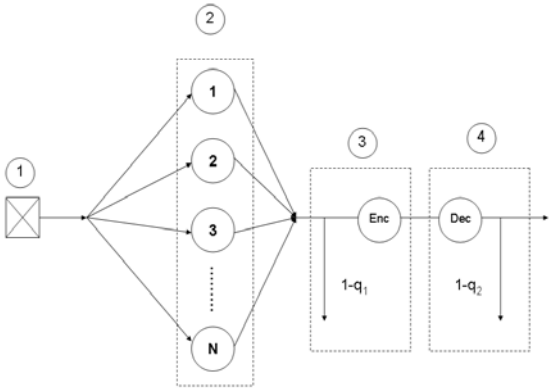

Fig. 2. Decentralized Model

2) Model

Authentication and policy verification computing time are now replaced with encryption and decryption time. The fact that the request is routed using multicast adds complexity to the event handling. In a decentralized architecture, nodes usually have limited capacities as compared to a registry: For this reason, we considered in our model that servers do not buffer new requests when they are busy. In Figure 2, the execution takes this order:

1. Client service discovery request arrival: requests are generated according to an arrival process with rate λ.

2. Servers message processing: all the available servers are contacted by the client via multicast. Each of these servers has to decrypt the messages in order to authenticate and access to client's request. The time to decrypt is assumed to be a random variable with a mean value $1/\mu_1$.

3. Service authentication: q_1 is the probability to successfully decrypt a client request. In case of success the server has to encrypt the response message to the client.

4. Client authentication: q_2 is the success decryption probability of a client, i.e., the probability that a client be compliant with this service policy.

D. Mathematical Model

Markov chains corresponding to the two queuing models were detailed and validated with a continuous time simulator described in [15]. The centralized discovery model is represented by a bi-dimensional Markov chain (see Figure3); the first dimension of the Markov chain (A) represents the number of requests stored in the cache plus the number of requests currently processed (0 or 1). The second dimension of the Markov chain (B) is a Boolean representing the request in the second processing cycle. If B = 1, the parameter A represents the number of requests in the cache. If B = 0, A represents the number of requests in the cache plus one request in the first cycle processing state. For instance, the left upper state in Figure 3 corresponds to A = 0 and B = 0. Client requests are entering the system according to a Poisson process with rate λ. The parameter A is the first to be incremented. After an authentication and verification first cycle (exponential with rate μ), the system moves to the second authentication and verification cycle with a probability q_1 (B = 1) or the client is rejected with a probability $(1-q_1)$. If B = 1 and a new request reaches the registry, only A will be incremented.

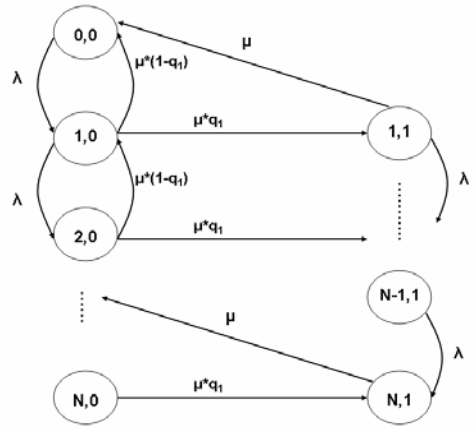

Fig. 3. Markov Chain for Centralized Model

The Decentralized model is represented by three linear Markov chains; the first one shown in Figure 4 describes the number of servers trying to decrypt protected requests sent by the users according to a rate λ. The second and third Markov chains (Figure 5 (a) and (b)) are describing successively the response encryption of a server and the response decryption of a client.

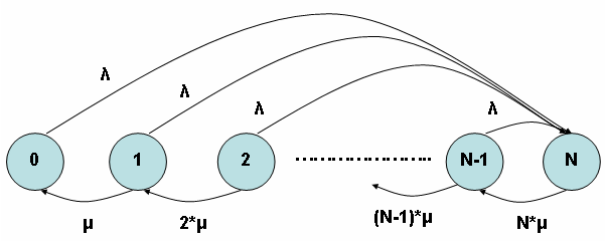

Fig. 4. Decentralized Markov Chain Model

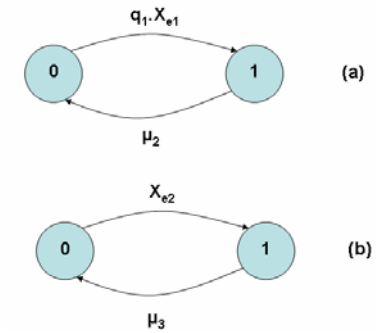

Fig. 5. a – Encryption Markov chain; b - Decryption Markov chain

These mathematical models make it possible to study the performance parameters of service discovery and to determine whether a centralized or a decentralized strategy should be adopted. In [16] we detail and analyze some performance parameters obtained using this model, especially the request rejection rate, the mean number of requests processed in the systems, and the mean service time of the centralized and distributed service discovery system.

III. Performance Evaluation of DoS Attacks Against Service Discovery Systems

As pointed out in [6], it is not possible for a malicious user observing the system and analyzing encrypted service discovery messages to build a software attack that would subvert the service discovery system. Such an attack requires a total or partial knowledge of the message parameters such as the service profile, the WSDL file, or the requested attributes. The only possible attack is a brute force denial of service (DoS) that consists in sending huge amounts of bogus messages to the servers in order to slow them down. This is especially true for the decentralized secure solution where security is based on computationally intensive cryptographic algorithms.

A DoS attacks may be carried out either with a small number of hosts using spoofed addresses or alternatively using a botnet consisting of a large number of peers under the control of the attacker. An efficient approach against DoS attacks is to use anti-clogging mechanisms [17], which force the requester to perform a costly computation upfront before his request be analyzed by the server. At the same time, this technique enables to verify the validity of the IP address of the requester.

This section introduces two performance models for assessing the impact of DoS attacks in centralized and decentralized service discovery architectures and the efficiency of anti-clogging mechanisms for the two types of DoS attacks, the ones based on spoofed addresses and the ones making use of a botnet.

A. Attack Model

We consider brute force DoS attacks that consist in injecting bogus messages into the system. The bogus traffic is modeled as an additional traffic class with arrival rate λ_{attack} melt to the class of normal traffic. If no anti-clogging mechanism is used, malicious traffic is processed until either the matching phase in centralized model (step 3 in Fig. 3) or the authentication phase in the decentralized model (step 2 in Fig. 4), where all corrupted messages are dropped.

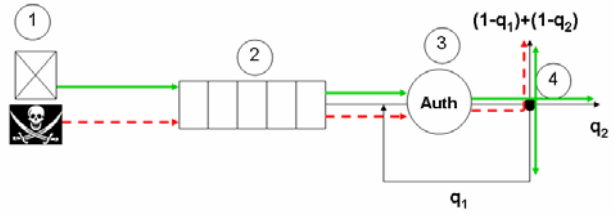

Fig. 3. Attack model for a Centralized Architecture

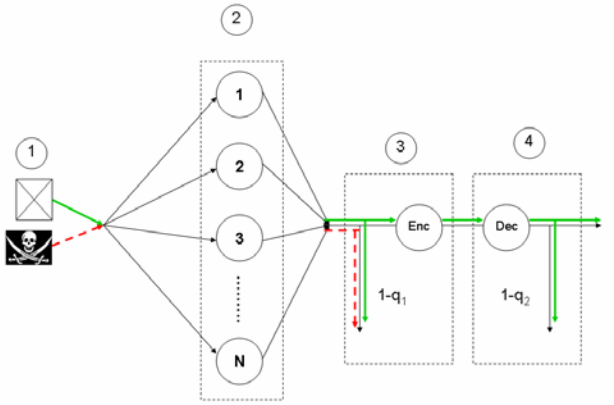

Fig. 4. Attack Model for a Decentralized Architecture

Anti-clogging mechanisms like puzzle auctions [17] introduce little overhead on the server side as compared to the computational effort required on the client side. In [17], Wang et al. estimated to 4.6 μs the puzzle construction time on a Pentium III, which is clearly negligible compared with the processing time of a request. In our model, we model the anti-clogging function as a first stage in both the centralized and decentralized models with an exponential service time with an average value equal to a small fraction of the other service times of the models.

B. DoS Attacks using Spoofing

In this section, we focus on the impact of anti-clogging on the performance of each architecture (centralized or distributed) when the DoS attack is carried out using spoofing. In such a case, the attack is defeated by the sending of the puzzle, as the spoofed machine, if any, is unable to answer.

We consider, for each architecture, the case where anti-clogging is used (protected systems) or not (unprotected system).

We consider the following scenario: normal requests arrive at a rate λ=8 requests/time_unit. Processing time values are

the ones used in [16]: the number of servers and the buffer size (in the centralized model) are set to 5 and q_1 equals to 0.5. Qualitatively speaking, those values correspond to a scenario where the registry and the distributed servers have limited capacities, as can be the case in a wireless scenario with off-the-shelf machines with power constraints. In such an environment, machines from normal users and also from attackers have limited capacity. Even if malicious machines are more powerful than normal machines, one should keep in mind that the objective of the DoS attack is not necessarily to tear down the network infrastructure, but only the service discovery system, which imposes a constraint on the maximum value of λ_{attack}.

We vary the attack rate λ_{attack} from 0.5 to 40 requests/time_unit and observe its impact on the normal traffic in terms of rejection rate and response time. Note that when λ_{attack}=40 requests/time_unit, malicious (resp. normal) traffic represents 83.3% (resp. 16.7%) of the total traffic. Given the discussion above, such a value for λ_{attack} might be considered as very high in environments like an ad-hoc wifi network for instance.

Figure 5 presents the acceptance ratio of normal traffic (i.e. one minus the rejection rate). Anti-clogging has a significant impact on both architectures as long as malicious traffic does not overwhelm normal traffic (here, when λ_{attack}<8 requests/time_unit). We also note that anti-clogging is more efficient for a centralized architecture than for a distributed one, because distributed servers have no buffers. Again, the limited capacity of the distributed servers might be related to the network environments. If the servers were in a LAN environment, they might be much more powerful. Note that our analytical model enables us to quickly assess the impact of such a DoS attack for other environments or machine specifications.

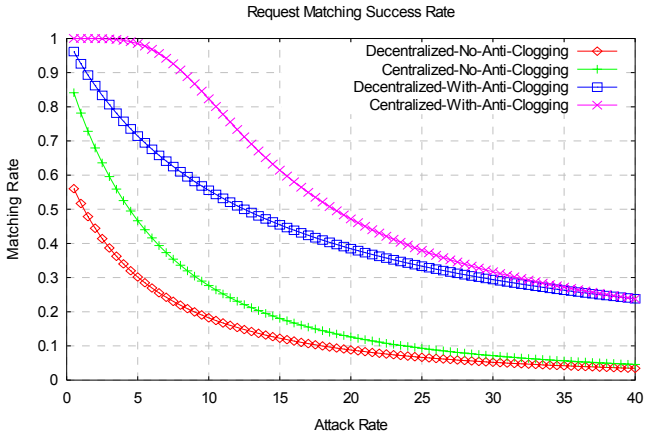

Fig. 5. Acceptance ratio for normal traffic (with and without anti-clogging mechanisms)

Figure 6 depicts the impact of a DoS attack on the total sojourn time of a request in the system, a.k.a response time. In the decentralized model, the sojourn time is the same irrespectively of the use of anti-clogging because no buffering is performed at the distributed servers. Anti-clogging thus only impacts the acceptance ratio of the non corrupted traffic for this architecture.

For the case of a centralized architecture, we notice that corrupted requests may use some slots in the registry's buffer, even if anti-clogging is used (since the buffer is a shared resource between bogus and normal traffic anyway). Without anti-clogging, corrupted requests are dropped just after the first authentication cycle. With anti-clogging, they are dropped earlier as the time to generate a cryptographic puzzle is small compared to the matching time. For this reason, we notice in Figure 6 that for a centralized model using anti-clogging, the average sojourn time of a non corrupted request takes more time than without anti-clogging; and the discrepancy between the two curves increases with an increasing attack rate.

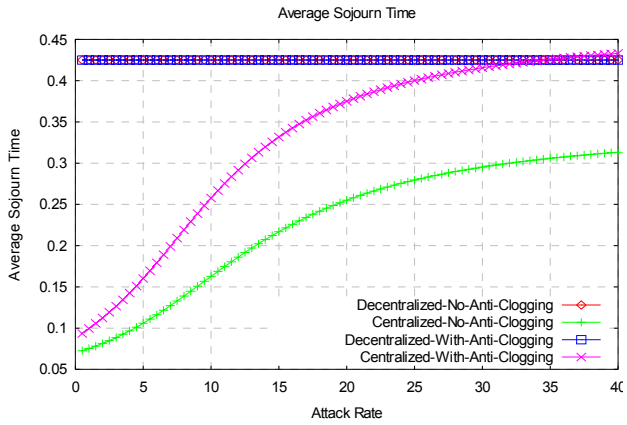

Fig. 6. Total sojourn time of a request in the system

C. *DoS Attacks using Botnet*

In this section, we focus on the impact of anti-clogging on the performance of each architecture (centralized or distributed) when the DoS attack is carried out using a botnet. In such a case, anti-clogging will slow down the attacker but, assuming they can resolve the cryptographic puzzle, the attacking machine will be able to proceed one step further in the system, like in an unprotected system.

It is out of the scope of this paper to formally evaluate the impact of a DoS attack using botnet, as it requires to assess by how much the maximum attack rate of the botnet is decreased when anti-clogging is used. The latter depends on the environment in which the service is deployed (wireless, wired, ad-hoc or connected to the Internet). We can however speculate on the efficiency of anti-clogging by looking at Fig. 5. Indeed, we observe from Fig. 5 that as soon as $\lambda_{attack} > \lambda$, a significant amount of normal traffic is rejected. The complexity of the anti-clogging algorithm must be such that it caps the attack rate to values close to the rate of legacy traffic. This can be a complex task depending on the environment. In addition, one should not forget that increasing the complexity of the cryptographic puzzle also impacts machines from normal users.

Overall, we can conclude that a DoS attack is apparently

the only feasible attack against our secure service discovery system. Anti-clogging can mitigate the impact of such an attack. However, its efficiency vary depending on the resources available at the network and at the application level, and also on whether the attackers are willing to attack only the service or are ready to tear down the whole network.

IV. Conclusion

In this paper, we introduced analytical models providing a performance analysis of centralized and decentralized service discovery systems that take into account scenarios where DoS attacks are performed against the system. This is the first such analytical study of this problem to best of our knowledge. This study permits to assess the impact of such a DoS attack depending on the resources available at the attacker side (single machine or full botnet). We further estimate how anti-clogging can mitigate DoS attacks depending on the intensity of the attack. Our model is flexible enough to allow a precise assessment of the level of threat due to a DoS attack depending on the deployment scenario and the resources of the attacker, which could constitute a valuable tool for system administrators.

References

[1] M. Weiser, , "The Computer of the 21st Century". Scientific American, vol. 265, no. 3, 1991, pp. 66–75

[2] H. L. Luo , M. Barbeau, "Performance Evaluation of Service Discovery Strategies in Ad Hoc Networks". Second Annual Conference on Communication Networks and Services Research, 2004, pp. 61-68

[3] M. Barbeau, E. Kranakis, "Modeling and Performance Analysis of Service Discovery Strategies in Ad Hoc Networks". International Conference on Wireless Networks, 2003, pp. 44-50.

[4] C. Dabrowski, K.L. Mills, A.L. Rukhin, "Performance of Service-Discovery Architectures in Response to Node Failures". Software Engineering Research and Practice, 2003, pp. 95-104

[5] S. Trabelsi, Y. Roudier, J.C Pazzaglia, "Discovery: Threats and solutions". 2nd Conference on Security in Network Architectures and Information Systems, Annecy, France, 2007

[6] S. Trabelsi, Y. Roudier, J.C Pazzaglia, "Service discovery: Reviewing Threats and Security Architectures". Research Report RR-07/197, Institut Eurecom, 2007

[7] S. Trabelsi, L. Gomez, Y. Roudier, "Context-Aware Security Policy for the Service Discovery". Symposium on Security in Networks and Distributed Systems (SSNDS) Niagara Falls, Canada, 2007.

[8] S. Trabelsi, J.C Pazzaglia, Y. Roudier, "Secure Web service discovery: overcoming challenges of ubiquitous computing". 4th IEEE European Conference on Web Services, Zurich – Switzerland, 2006

[9] O. Garofalakis, et al, "Web Service Discovery Mechanisms: Looking for a Needle in a Haystack?". 15th ACM Conference on Hypertext and Hypermedia, Santa Cruz, USA, 2004.

[10] F. Zhu, M. Mutka, and L. Ni, "Splendor: A secure, private, and location-aware service discovery protocol supporting mobile services". First IEEE International Conference on Pervasive Computing and Communications, pp. 235–242, 2003.

[11] F. Zhu, M. Mutka, and L. Ni, "Prudent exposure: A private and user centric service discovery protocol". 2nd IEEE International Conference on Pervasive Computing and Communications, Orlando, USA, 2004.

[12] S.E. Czerwinski, et al, "An Architecture for a Secure Service Discovery Service". MobiCom, Seattle, WA, 1999.

[13] A. Shamir, "Identity-based cryptosystems and signature schemes". Advances in Cryptology, Lecture Notes in Computer Science, Vol. 196, pp. 47-53,Springer-Verlag, 1984.

[14] V. Goyal, et al, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data". 13th ACM Conference on Computer and Communications Security, Alexandria, USA, 2006.

[15] S. Trabelsi, G. Urvoy-Keller, Y. Roudier, "A Markovian performance model for secure service discovery systems". Rapport de recherche RR-08-214, Institut Eurecom, 2008.

[16] S. Trabelsi, G. Urvoy-Keller, Y. Roudier, "A Performance Based Approach To Selecting A Secure Service Discovery Architecture" in Proceedings of the Ifip Mobile and Wireless Networks Security workshop - MWNS 2008, Singapore, 2008.

[17] X. F. Wang, M.K Reiter, "Defending against denial-of-service attacks with puzzle auctions", in proceeding Symposium on of Security and Privacy, USA, 2003.