Secure Service Discovery with Distributed Registries

Slim Trabelsi[1], Yves Roudier[2]

[1]SAP Labs France, 805, Avenue du Docteur Maurice Donat 06254 Mougins Cedex, France
[2]EURECOM, 2229 route des Crêtes, BP 193, 06904 Sophia-Antipolis, France
Slim.Trabelsi@sap.com, Yves.Roudier@eurecom.fr

***Abstract*—This paper discusses how to extend service discovery mechanisms to support a scalable querying and indexing system that addresses security requirements related to existing service discovery standards. Through the use of an onion routing protocol, anonymizing the publication and the access to service profiles, combined with a P2P registry infrastructure we propose a scalable and secure service discovery architecture the can be deployed over non trusted network domains.**

***Index Terms*— security, service discovery, distributed hash tables, onion routing, attribute based encryption**

I. Introduction

Service discovery is a basic component of Service Oriented Architectures (SOA) that enables the dynamic detection of services available in the network, yet previously unknown. Such a flexible mechanism unsurprisingly comes with new security challenges regarding trust and privacy. Private data exchanged during the discovery process can be captured and reused for unwanted purposes. The openness and criticality of the discovery protocol makes it a very likely target of denial of service attacks. Most existing solutions to secure discovery rely on trusted third parties such as security modules, secure proxies, or trusted registries to encrypt the data exchanged or to establish trust between service discovery participants. This requirement for trusted third parties makes it much harder for service discovery protocols to be deployed on a large scale; even though services seem to be the essence of ambient environment capabilities in pervasive computing, such a requirement also makes it less likely for SOA to finally make its way into such applications if service discovery cannot be secured in a scalable fashion. Contrary to centralized discovery architectures, in which the absence of a service is detected at the registry, architectures with distributed registries can propagate queries unresolved locally to registries belonging to other domains and networks in order to extend the search scope.

This paper discusses how to extend service discovery to support the querying of service profiles stored in a Distributed Hash Tables (DHT) of registries for P2P indexing. The solution we propose provides a scalable discovery system that addresses the security requirements defined in [1] and [2] through the use of an onion routing protocol anonymizing the publication and the access to service profiles.

II. Related Work

A distributed architecture for service discovery was proposed by Chakraborty et al. [3] that aims to provide an insecure local service discovery in a P2P manner, using advertisements between local peers and groups for scalable service discovery. Services are categorized into hierarchical groups according to their capabilities, each node of the system being in charge of a group. A client request not matching locally is forwarded to the node in charge of the group to which the requested service belongs. Despite the authors' claim concerning scalability, each node must know the whole group hierarchy to route the request to a correct registry, an unrealistic assumption for a system involving millions of nodes throughout the world.

To our knowledge, the first study dealing with security and particularly privacy for distributed and scalable discovery architectures was proposed by Cardoso et al. [4]. It extends the MUSDAC [5] middleware enabling interaction between various service discovery protocols. In each local network, a MUSDAC manager is deployed on top of the existing discovery protocols as an interface handling inter-domain and inter-network discovery and access requests. Extending the discovery scope to other domains raises new privacy issues: discovery information with private data should be protected, which is addressed using a gradual trust model regulating the execution of the discovery process. The scalability of this approach is however strongly limited by the need to know the public keys of all bridges and managers in particular in a ubiquitous computing setting.

We proposed in [6] a first solution that relies on attribute based encryption applied to messages exchanged between registries, while matchmaking on a small part of the message in clear. This solution is scalable enough but insufficiently secure as it is possible for an attacker that analyses the traffic to get information about the client's intentions and to trace the addresses of the node publishing services.

III. Requirements

Scaling the discovery system does not mean sacrificing its security. Most of the existing secure service discovery solutions are relying on a local component in charge of establishing a secure channel between clients and servers by authenticating them and encrypting the messages exchanged. As soon as the discovery system deals with multiple distributed registries, it becomes a hard task to establish a trust relationship with all these registries. The distributed discovery system must therefore fulfill the following requirements:

- Scalability: no limitation related to the network size and the number of clients or services.
- Efficient intra domain lookup: no restriction for the request scope.
- Efficient matching and indexing: avoiding collisions and bad matching performance values.
- Privacy protection: attackers must not know which clients sent a request nor where the critical services are located.
- Authentication: implicit or explicit authentication for trust establishment
- Access Control: only authorized entities should discover restricted resources (requests or profiles).

IV. A Scalable Distributed Registry Model

A. Distributed Architectures for Service Discovery

The main motivation of this paper is to achieve scalable and distributed service discovery. Most scalability studies in the domain suggested the use of distributed registries. The indexing and routing operations essentially depend on the architecture adopted to deploy registries (see Figure 1):

- Flat: all registries are interconnected and can communicate through broadcast or multicast. There is no specific indexing/retrieval strategy; in case of new service request addressed to one of the registries registry that does not store the relevant information about the requested service, this one will forward the request to the other registries in order to find a matching to the query. Such an architecture might work for a small number of registries (less than 100) but becomes inefficient for a huge number of registries where the anarchic indexing/retrieval strategy generates increasingly important messaging overhead and message delivery latencies.
- Hierarchical: the registries are deployed as a tree in which data is indexed according to a structure distributed hierarchically through tree nodes. Indexing and retrieval operations will represent in the worst case *log(n)* operations. This architecture overcomes the scalability and indexing limitations of the flat architecture. However, in case of a registry failure or shutdown, the recovery process can be very costly and requires a replication system and an important signaling procedure. This could affect the availability of service discovery.
- DHT-based P2P: an alternative architecture currently used in file sharing applications relies on Distributed Hash Tables (DHTs) for indexing and retrieval. Each peer (a registry in our case) is in charge of indexing and maintaining a mapping between names and values. Indexing is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows DHTs to scale to an extremely large number of nodes while handling continuous node arrivals, departures, and failures. This architecture can handle thousands of registries with millions of data entries.

DHT based systems are clearly the most appropriate solution for deploying a scalable and robust service discovery system for most ubiquitous computing systems deployed over the Internet. The rest of this paper discusses which security measures can be adapted to their use.

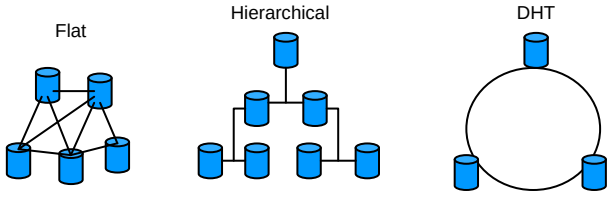

Fig. 1. Alternative architectures for distributed registries

B. Distributed Hash Tables

A distributed hash table (DHT) is an indexing and location system dedicated to peer-to-peer information storage. This distributed system enables a user (node) to efficiently retrieve the value associated with a given name. DHT-based indexing systems work efficiently for millions of users and data. The behavior of the system is strongly dependent on the collaboration of nodes. Each node with a fixed identifier is responsible for a range of key values representing a pointer to a stored element. Each stored element has an index value represented by a hash key. The key space is distributed among participant nodes, each node being in charge of a partition, in a circular fashion. If a new element is added to the system, the name of this data is hashed. Depending on the hash key value, the pointer to the data will then be assigned to the nodes in charge of the correspondent key range. In order to retrieve this data, a hash key is generated from the requested name: the value of the obtained key gives an indication about the nodes in charge of maintaining the information related to the stored data. The request is then routed to the node that will return the pointer for reaching the requested element.

DHT based systems provide interesting properties that can be exploited in service discovery mechanisms:

- Decentralization : the system consists of many autonomous clients without any central control
- Scalability : the system can adapt to a large number of peers
- Fault tolerance : the network is resilient, especially

with respect to stale peers

- Load balancing : messages are routed in a balanced fashion, which reduces the network overhead

C. Indexing and Data Retrieval

We selected the most scalable and reliable technology currently deployed for distributed indexing, the DHT called Kademlia [8], which provides a P2P storage and lookup protocol. This protocol is installed as an external interface for all the registries involved in the discovery system and scattered all over the world. Using such an interface, active registries can permanently update information about the stored profiles like a P2P client with files it shares with others. In case of a new service publication, the registry stores the new entry locally. It then hashes the description name of the new entry, and finally sends the key to the appropriate registry in the indexing circle, with a pointer to the service entry. If a client sends a service discovery request to the local registry, the latter tries to find the service entry locally. If no entry is found, the local registry hashes the query and forwards the request to the registry in charge of the hashed key value that points to the final registry.

For this purpose, we reuse the message format of Kademlia:

- STORE: To publish a <hkey, value> pair, the registry locates the k nodes closest to the key and sends them STORE RPC messages.
- FIND_NODE: To retrieve the node in charge of the hkey corresponding to a published service, the requesting registry sends a FIND_NODE message containing a triple (IP address, port, nodeID) for the contacts that it knows to be closest to the key.
- FIND_VALUE: If a corresponding value is present on the recipient node, the associated data is returned. Otherwise, the RPC request is equivalent to a FIND_NODE and a set of k triples is returned in response.

D. Inter-Registry Indexing and Data Retrieval

1) Inserting a new node

Each registry/node maintains a routing table to locate other registries/nodes of the indexing circle. If a new node joins the system, these routing tables must be updated.

Algorithm 1: Insert Node()

```
0. Begin
1.  | if contact exists then
2.  |  | update contact information (IP address, keys, pointers);
3.  | else
4.  |  | create new entry to the table;
5.  |  | insert the node contact information (IP address, keys, pointers);
6.  | endif
7.  End
```

Fig. 2. Algorithm for node insertion

In practice the routing table used by a node does not point to all the nodes of the system (there are potentially millions of them) but only to a few nodes representing a routing zone, similarly to traditional routing tables for network routers. If a new node joins one of these routing zones, his positions must be added to the routing tables of the other nodes. The algorithm describing a new node insertion is described in Figure 2.

2) Publishing a new entry

The registry receiving a new service publication will forward this new entry to the appropriate registry in the indexing circle. If the entry already exists, the information will be updated. The algorithm for a new key insertion is described in Figure 3.

Algorithm 2: Adding new entry (*key*)

```
0. Begin
1.  | if keyword had already sources then
2.  |  | create a new pointer for the existing keyword
3.  | else
4.  |  | Add new <keyword, pointer> to the local record
5.  | endif
6.  End
```

Fig. 3. Algorithm for key insertion

V. Securing the Access to Distributed Registries

The scalable indexing and retrieval part of the discovery system described above must be protected against potential threats that can affect the publish/request operations of the service discovery.

A. Need for Anonymity

In a previous work [9], we proposed the use of an Attribute Based Encryption (ABE) mechanism [7] to prevent discovery messages against unauthorized access and to protect private information contained in these messages. Using this cryptographic mechanism, we are able to authenticate messages and to disclose discovery related information to authorized participants only. Applying this technique to a distributed registry system raises some issues since registries have no mean to decrypt discovery messages and to match between client requests and service profiles. The solution that we presented in [6] keeps the keywords to be matched in clear and encrypts the rest of the message in order to let the registries match the requests correctly. Unfortunately this tradeoff introduces new threats related to privacy protection: an untrusted registry or an external observer may intercept a message and deduce the behavior of the client. We suggest improving this solution by anonymizing the message senders: the registry can then be reached without hiding matching elements. Numerous anonymity techniques exist to protect message senders. The most commonly deployed is based on the use of proxies, placed between the endpoint users and the rest of the network in order to relay all the traffic issued by the users without showing the original address of the initiator. This technique requires the deployment of one proxy per user and does not protect against local traffic analysis that could be used to identify the initiator address. Cryptographic mixes overcome these limitations, which consist in creating a non direct path between the sender and the receiver in which a number of relays will exchange the initial message, each relay hiding the information about its predecessor. With this configuration, each node only knows the previous and the next relays, the final receiver being unable to retrace the route

of the message. Several variants of this concept have been produced over the years, notably Chaum Mixes [10], Onion Routing [16], Web-Mixes [11], SG Mixes [12], and Crowds [13]. All of these solutions provide a scalable and efficient mechanism for anonymizing a forward path between a sender and a receiver. There are some limitations concerning the backward path for which these systems do not provide any particular protection. Kate et al. [14, 15] fixed this problem by devising a pairing-based onion routing protocol in which they rely on pseudonyms to identify the nodes involved and to facilitate a two way anonymous path construction. This protocol is described in the next section. We chose to integrate this anonymity mechanism for a secure service discovery system relying on DHT-based registries.

B. Onion Routing and Pairing-Based Onion Routing

1) Onion routing

Onion routing [16] is a scheme for anonymous communication in which users can communicate while hiding their identities from third parties. This approach is called Onion Routing, because it relies upon a layered object to direct the construction of an anonymous, bidirectional, real-time virtual circuit between two communicating parties, an initiator and responder. Onion Routing hides routing information through the routing of an encrypted data stream follow a path through intermediary nodes until the destination. To begin a session between an initiator and a responder, the initiator node identifies a series of routing nodes forming a path to the destination. The initiator constructs an onion message which encapsulates that path. Figure 4 illustrates an onion constructed by the initiator Node W for an anonymous route to the receiver Node Z through intermediate routing nodes X and Y. The initiator then sends the onion along that route to establish a virtual circuit between himself and the receiver Z.

The onion message structure is composed of a superposition of encrypted layers. The core of this onion contains the clear message to send. The basic structure of the onion is based on the route to the receiver that is chosen by the initial sender.

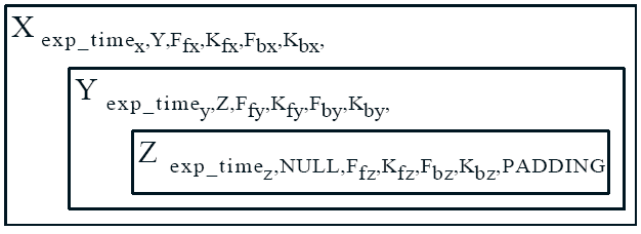

Fig. 4. Onion message format [16]

Based on this route, the initiator encrypts first for the receiver, then for the preceding node on the route, and so on back to the first routing node to whom he will send the onion. Each node knows who sent him an onion and to whom it should pass that onion although it knows nothing about the other nodes, how many layers there are in the chain, nor the current layer's place. The virtual circuit established between the node W and the node Z is described in Figure 5.

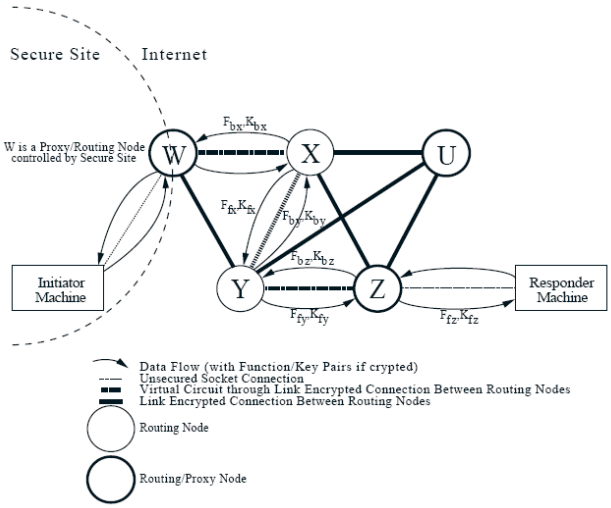

Fig. 5. Onion routing virtual circuit [16]

The most famous anonymity software using this technique is TOR[1] (The Onion Router) that is originally sponsored by the US Naval Research Laboratory that actually becomes an open source project.

2) Pairing-Based Onion Routing

This section describes how the pairing-based onion routing protocol works.

Pseudonyms and key agreements. In order to protect the anonymity of users involved in the system, each node selects a pool of pseudonyms for which they will generate private keys. These pseudonyms will be announced to the other nodes of the systems. When a node A wants to contact another one B, A will use the pseudonym of B as a session key KBA to encrypt the secure forward message. At the same time B using the pseudonym of A and his own private key to build the backward session key KBA used to secure the backward path.

Circuit construction. Before staring the contribution to the routing system, a user has to create a set of routes (information provided by directory servers with a list of available routes). After choosing a circuit, the user has to generate appropriate session keys for pseudonyms of each node involved in the system in order to encrypt the onion message. If one of the involved nodes receives the message, it will decrypt its onion layer with its private key, derive the backward session key, and forward the message to the next pseudonym.

C. Anonymizing Publish / Request Service Discovery Messages

This section describes the mechanisms used to protect service discovery message exchange in the WS-Discovery protocol [17] with distributed DHT-based registries.

Protecting the publish message: A server publishing a set of restricted services to untrusted registries will first encrypt all the data related to the identity, location and methods provided by these services. Only encryption can hide this kind of information. An ABE encryption can be applied to the

[1] http://www.torproject.org/

publish message, the profiles of the users that are allowed to decrypt the message and to discover the services being the encryption key argument. A part of the publish message must remain clear in order to enable an easy matching for the registry. In order to be authenticated by the client, the server has to sign the publish messages of his services using the private keys related to his service profiles. Figure 6 illustrates the construction of a partially encrypted publish message restricted to users with the role {professor}.

```
<s:Envelope>
  <s:Header>Encrypt[Header]{Professor}
  </s:Header>
  <s:Body> <d:Hello>
    <a:EndpointReference>
 Encrypt[EndpointReference]{Professor}
    </a:EndpointReference>
    <d:Types>Printer</d:Types>
    <d:Scopes>University</d:Scopes>
    <d:XAddrs>
     Encrypt[XAddrs]{Professor}
    </d:XAddrs>
</d:Hello></s:Body></s:Envelope>
```

Fig. 6. Publish (WS-Discovery Hello message) message structure

Anonymous publishing of services: The server must send a secured publish message to the local registry without giving information about the identity and endpoint address of the services. Such information might be deduced by correlating the service description (clear text in the message) and the address from which the message is sent. The server selects an onion routing path using the pseudonyms of the other nodes belonging to the same registry and uses this anonymizing path to securely send the message to the registry. The anonymous reverse path is used to return publication acknowledgements with a unique identifier that can be used by the server to update or deregister its services. Figure 7 details the message sequence to perform an anonymous service publication using $i > 3$ intermediary onion routing nodes.

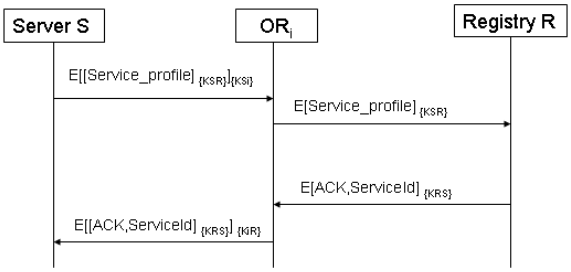

Fig. 7. Anonymizing publish messages

Protecting request message: in order to authenticate the published services and verify the authenticity of the published services, a user can verify the ABE signature of a selected service by using the description attributes of the service as a key for signature verification. For this reason, the request message does not need to be encrypted. Only correlation between the requested service profile and the user's address must be prevented by making the request anonymous.

Anonymous service request: the same anonymity method is used for the request and the publish actions. Before contacting the local registry, the request message must be routed through an onion mix to prevent any attempt of correlation between the requested service and the requester identity. In this case the user has to choose a path to the registry according to the pairing-based onion routing protocol.

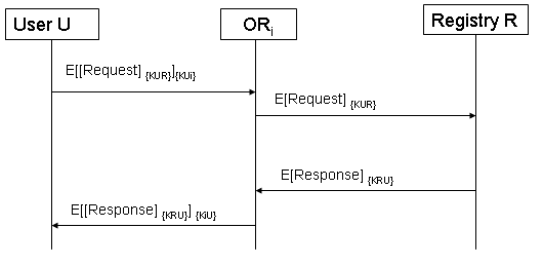

Fig. 8. Anonymizing request messages

VI. Architecture for a Secure Distributed Registry-Based Service Discovery

This section describes the steps executed during a secure service discovery relying on untrusted distributed registries (see Figure 9). As we explained previously our system relies on an important number of untrusted registries distributed all over world wide network. These registries are communicating through Kademlia for an optimal indexing and retrieval of services. Clients and services have to anonymize their requests before accessing to their local registry in charge of publishing and retrieving services. In this case we suppose that clients and servers have a prior knowledge about the location of the local registry.

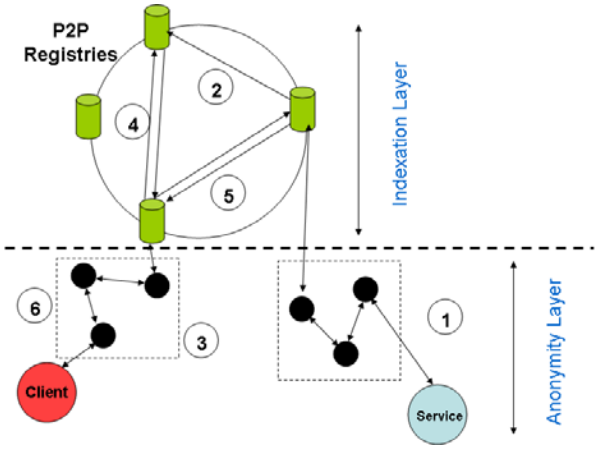

Fig. 9. Architecture for a secure distributed registry-based service discovery

Setup: Clients and services create a list of anonymous routes by requesting the pseudonyms of the nodes depending on the same local registry.

Step (1) the registry chooses one or more anonymous paths to the local registry then generates the publishing message according to the pseudonyms of the nodes of the chosen route. The publish message is routed anonymously to the local registry that decrypts the content and stores the service entry locally. The registry sends back an acknowledgement and a unique ID through the same anonymous path.

Step (2) after storing the service coordinates the registry hashes the service description and publishes the key obtained into the appropriate registry in charge of this value.

Step (3) the client chooses one or more anonymous paths to the local registry then generates the request message according to the pseudonyms of the nodes of the chosen route.

Step (4) the registry matches the request locally with the published service profiles. If the service does not exist locally, the registry hashes the request attributes to obtain a key. The local registry contacts the remote registry in charge of this key value in order to get the location of the registries that store the information about the requested services.

Step (5) the local registry contacts the remote registry holding the information about the requested services. The remote registry matches the request locally and sends back a response containing the entry related to the requested service.

Step (6) the local registry sends back the response to the client using the same anonymous route initiated by the client.

VII. Security Evaluation

This section informally evaluates our solution with respect to the security of service discovery.

Property 1: a user intercepting a clear message in the P2P network cannot identify the service provider/ requester.

Proof: Private information contained in clear exchanged messages are not labeled as private since the data holder is completely anonymous. Due to the onion based anonymizing routing, an attacker intercepting these data cannot link users involved in the system to the discovery data exchanged.

Property 2: Private services are not accessible for unauthorized users.

Proof: Published private services can be encrypt published service profiles using an ABE encryption scheme to restrict the access to the discovery of his service, metadata only being kept in clear. Only users with the restricted profile and corresponding ABE private key can access service description.

Property 3: Fake published services are detected by users.

Proof: Upon receiving a response to his service request, a client can verify the ABE signature attached against the requested service profile.

Property 4: Registries cannot identify message sources and destinations.

Proof: All discovery messages being anonymous, the registry cannot affect the privacy of the participants.

VIII. Conclusion

This paper describes a scalable solution combining anonymous routing and P2P indexing for securing service discovery without relying on trusted registries. Our proposal is based on a two layer approach: a first layer provides an anonymous access through which clients and servers publish their profiles or send requests to a set of local registries using onion based routing. A second layer deals with the inter-domain routing through which all local registries spread all over the internet can communicate and exchange discovery information using a P2P metadata exchange protocol such as Kademlia and achieve a scalable indexing and retrieval of service profiles. This architecture addresses scalability issues of decentralized approaches to service discovery without the usual requirement of deploying an infrastructure of trusted third parties of centralized approaches.

References

[1] S. Trabelsi, J. C. Pazzaglia, and Y. Roudier, “Enabling secure discovery in a pervasive environment”, SPC 2006, 3rd International Conference on Security in Pervasive Computing, April 18 - 21, 2006, York, UK - also published in LNCS Volume 3934 , pp 18-31

[2] S. Trabelsi, Y. Roudier, and J. C. Pazzaglia, “Discovery: Threats and solutions”, SAR-SSI 2007, 2nd Conference on Security in Network Architectures and Information Systems, 12-15 June 2007, Annecy, France

[3] D. Chakraborty, A. Joshi, Y. Yesha, T. Finin, "Toward Distributed Service Discovery in Pervasive Computing Environments", Article, IEEE Transactions on Mobile Computing, pp. 97- 112, 2006.

[4] R. S. Cardoso, P.-G. Raverdy, V. Issarny. "A Privacy-Aware Service Discovery Middleware for Pervasive Environments", In Proceedings of IFIPTM 2007 Joint iTrust and PST Conferences on Privacy, Trust Management and Security. 2007.

[5] P-G. Raverdy, V. Issarny, R. Chibout, A. de La Chapelle. “A Multi-Protocol Approach to Service Discovery and Access in Pervasive Environments”, In Proceedings of MOBIQUITOUS – The 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services. San Jose, CA, USA, 2006.

[6] S. Trabelsi and Y. Roudier, "Secure service publishing with untrusted registries: Securing service discovery", SECRYPT 2007, International conference on Security and Cryptography, July 28-31, 2007, Barcelona, Spain

[7] V Goyal, et al, “Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data”, Proceedings of 13th ACM Conference on Computer and Communications Security (CCS 2006), Alexandria, USA, October 2006

[8] P. Maymounkov and D. Mazieres, “Kademlia: A Peer-to-peer Information System Based on the XOR Metric”. In Proceedings of the first International Workshop on Peer to Peer Systems IPTPS, Cambridge, MA, USA, 2002.

[9] S. Trabelsi, J. C. Pazzaglia, and Y. Roudier, “Secure Web service discovery: overcoming challenges of ubiquitous computing” ECOWS 2006, 4th IEEE European Conference on Web Services, 4-6 December, 2006, Zurich, Switzerland.

[10] D. Chaum, “Untraceable Electronic Mail, Return address, and Digital Pseudonyms”, Communications of the ACM 24/2, pp. 84-88, 1981.

[11] O. Berthold, H. Federrath, and S. Köpsell. "Web MIXes: A System for Anonymous and Unobservable Internet Access," H. Federrath, editor, Designing Privacy Enhancing Technologies, LNCS 2009, pp 115-129, 2001.

[12] D. Kesdogan, J. Egner, and R. Büschkes. "Stop-and-Go-MIXes Providing Probabilistic Anonymity in an Open System," Information Hiding 1998, LNCS 1525, pp 83-98, Springer Heidelberg, 1998.

[13] M. Reiter, A. Rubin. Crowds: "Anonymity for Web Transactions," ACM Trans. on Information and Systems Security, pp 66-92, 1 (1) 1998.

[14] A. Kate, G. Zaverucha, and I. Goldberg. “Pairing-Based Onion Routing”, in proceedings of 7th Privacy Enhancing Technologies Symposium (PETS 2007), 2007.

[15] Aniket Kate, Greg Zaverucha, and Ian Goldberg , “Pairing-Based Onion Routing with Improved Forward Secrecy”, Cryptology ePrint Archive, Report 2008/080, February 2008.

[16] D. Goldschlag, M. Reed, and P. Syverson, “Hiding routing information”. In First International Workshop on Information Hiding, pp. 137–150, (1996).

[17] WS-Discovery Specifications http://msdn.microsoft.com/ws/2005/04/ws-discovery/