Secure service discovery in open networks with SLP

Markus Alexander Kuppe - 8kuppe@informatik.uni-hamburg.de
Vitali Amann - 5amann@informatik.uni-hamburg.de

University of Hamburg
March 12, 2010

ABSTRACT

Mobile Ad-Hoc and open networks become increasingly common due to the wide availability of mobile device technology. When such devices enter a network, they want to learn about services offered by other peers or advertise own services. Service discovery protocols have been around to answer this requirement for years. A most prominent and widely adopted one is Service Location Protocol (SLP). However, SLP has not been designed for open networks, but for trustworhty and reliable networks. Thus the protocol has to undergo a renewal to make it fit for open networks.
This paper focuses on the security implications that arise from SLP's usage in open networks and gives a detailed threat analysis. Afterwards security extensions are proposed to address the identified shortcomings.

Keywords

Ad-hoc networks, open network, service discovery, service location protocol, security, trust, pki, web of trust, multicast, secure SLP

1. SERVICE DISCOVERY BASICS

Recent advances in computer science make it possible to build small mobile devices with many features. They have a long battery lifetime, a powerful processor unit and can handle more and more advanced applications. Mobile devices are here not just Notebooks but also PDAs, mobile phones, MP3 Players and so on. Due to the growing abilities of mobile devices it became reasonable to connect them. The idea is to connect several mobile devices to a network to allow them to communicate with each other and to share their services. In this paper we want to observe the possibilities offered by service discovery protocol (SLP). We will compare its current weaknesses and create suggestions how to fix them. To begin with, we first introduce the usage of such mobile devices in an open network.
A possible scenario is for example a hotel where we can find several user groups. For hotel employees it would be useful if special hotel services would be multicast. Those services can be like "available rooms", "technical control", "room cleaning" and so on. In this way a hotel employee has a possibility to be mobile and still make his work and controlling several hotel properties. Another user group could be the guests. For this group there are other useful services like "room service", "environment map", "weather", "great attractions" and so on. A guest would have a possibility to order something to his/her location (not necessarily into the room) or easily find out some interesting attractions near the hotel. But as both groups are in same network, there is also a need to hide and/or secure some services from unauthorized users. In case a vicious person gets access to the employee services he could control technical features or checking other guests in/out. Those problems should be prevented early. The guests shouldn't even be able to see the services for hotel employees and neither should they be able to use them. To hide the services and encrypt communication is already a way to prevent attacks on those services (more details will be discuss in section 3).
Other possible scenario is for example a person with a mobile phone who offers phone calls over an open network (we assume such a person has a national flat rate). Another person with a notebook could use such a service to call someone. Possible attackers could track the user and get the phone number he is calling or an attacker could betray the service provider with an international phone call. In this case it is reasonable to encrypt the communication between service provider and user and there should be a kind of trust that the service user doesn't use the service in a wrong way. But to make such an infrastructures work in this way there are some requirements to fulfill:

- server and client have to be on the same network
- both should speak the same lingo (use same protocols for example)
- to use or provide sensitive services they need some kind of security to find services in the network

The main subject of this paper is service discovery and its security issues that arise in hostile environment like open networks. This paper separates the service discovery and service invocation from each other and we don't discuss security issues after service invocation here. However, to understand the service discovery security and their weaknesses we first need to understand the basics of service providing and service discovery.

1.1 Open network

In this paper we are talking about networks as free or open networks. An open network is a network for everyone even for bad guys. It provides the possibility that everyone who wants to join this network will join it. In this way the network allows the communication between every peer that is a part of this network. Also the idea of an open network is to create a network everywhere without complicated operations. So the open network is very dynamical. The network can be created centralized with servers or it can be created decentralized with peer to peer communication. All peers can connect or disconnect to the network anytime and without any restrictions. In this way there are no constrains and you can connect every possible devices with each other like personal computers, servers, notebooks but also printers, mobile phones, mp3 players and other portable or stationary devices. Open networks aren't necessarily based on IPs but in this paper we work with service location protocol (SLP) so we presume the communication is IP based. How the open network works is not a part of this paper. Furthermore we don't treat security issues in open network itself and assume that it is secure [Foundation, 2009].

1.2 Service discovery architecture

There are three important architecture approaches used for service discovery [Ververidis and Polyzos, 2008].

1.2.1 Directory-based architecture

In directory-based architecture network peers have three possibilities how they can act. A network peer can offer its services as SA (**s**ervice **a**gent) to other peers, it can use discovered services as UA (**u**ser **a**gent) and a network peer can act as DA (**d**irectory **a**gent) which caches services provided by other SAs and forwards them to user agents. Because an open network is a dynamic network it can't be assume that there is always a reachable directory agent. But it is also possible that more than just one DA is operating in an open network. A directory agent is an important instance in this architecture and is essential to keep the network alive. All devices (service agents) which want to provide their services have to register them by a directory agent. They register their services by sending service description (e.g. service name, server IP, description what the service can do, etc.) to the directory agent which stores all that information. As soon as a user agent wants to use a service, it sends via unicast a request for a searched service to the DA or it request all services which the DA stores. After receiving the information about the requested service the UA can connect to the SA (also see figure 1.1). This architecture reduces the entire communication in the network. The devices don't need to communicate via multicast anymore, hence they don't force their resources which mean they also save battery lifetime. But on the other hand the benefit of this architecture is their biggest handicap at the same time. The DA in some kind centralizes the network and makes it vulnerable. If a DA leaves the network, the network also loses all the services that DA had stored. First problem arises for the SAs and UAs that they need a technique to notice a missing DA. And the other problem is that service providers and users have to find a new DA and for this time their services aren't reachable. In the worst-case they don't find a new DA and indirectly get excluded from the network. There are some approaches to avoid such cases but they aren't part of this paper so we won't discuss them here.

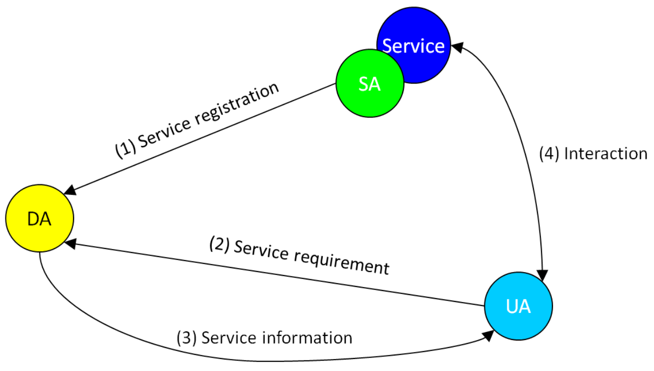

Figure 1.1: Directory-based architecture

1.2.2 Directory-less architecture

The directory-less architecture is quite contrary to the directory-based architecture. In this architecture the network peers can just act as service agents and as user agents. There is no directory agent so the service discovery should perform in other way. There are several approaches how this can be done. Service agents can distribute their services via multicast periodically send to the network. If a UA is interested in an offered service it requires the service information from the SA (also see figure 1.2). User agent can also discover services on their own by sending periodically a defined service request via multicast to the network till it gets an answer from a SA. Or a user agent can send a request to some network peers in its scope. In case a peer offers such a service it replies to the user agent otherwise the peer forwards the request to its neighbors and so on. In this architecture there is no central instance and it provides a much more stable network structure. However the communications via multicast drain more computation and battery lifetime from each network peer. Also to discover a service can take much longer compared to the directory-based architecture because there is no central authority that provides all services with their information.

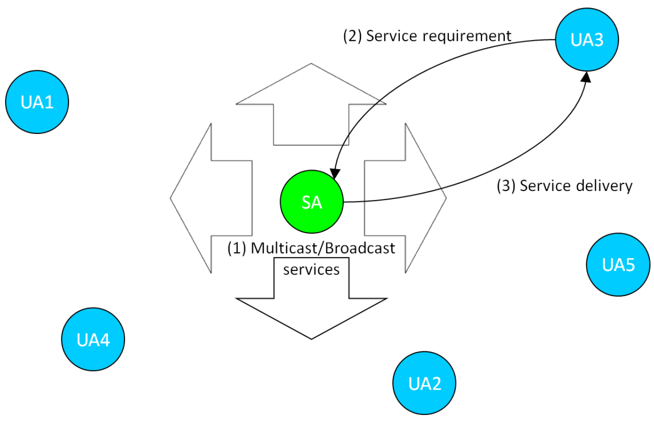

Figure 1.2: Directory-less architecture

1.2.3 Hybrid architecture

Hybrid architecture is a compromise of directory-based and directory-less architecture. Hybrid architecture combines the benefits of both other architectures. Like in directory-based architecture there are also three possibilities how a network peer can act (SA, UA and DA). To provide services a service agent first searches for a directory agent. In case a directory agent was found the service agent acts like in directory-based architecture and sends its service information to the DA and the DA is used by user agents to discover the services. In other case where no DA was located or all DAs left the network the SA acts like in directory-less architecture and periodically broadcast its services to the whole network. This architecture uses the benefits of the directory-based architecture to provide and discover services efficiently and to spare too many broad- and multicast communications in the network to keep battery lifetime from each network peer longer alive. At the same time the architecture offers a solution for the worst-case where no DAs are available in the network so the network won't die. The service location protocol, we will discuss below, is also based on the hybrid architecture.

1.3 Secure service discovery basics

Section 1.2 introduced some solutions how services can be provided and discovered in an open network. But there are also some security issues that prevent a secure usage of such a network. To safety use service discovery we have to fulfill at least the traditional security requirements:

- Authentication
- Authorization
- Integrity
- Confidentiality

Cotroneo et al. [2004] suggests securing the registration and deregistration of the services. If a service is to be registered/deregistered it is important that authentication, authorization, integrity and confidentiality are maintained. In that way only authorized service agents can register or deregister their services. While registering a service, the communication has to be secured (by encryption) to prevent changes in the transferred information (integrity and confidentiality). With these common techniques many attacks like replay attack, user tracking or manipulating service information can already be prevented or become at least difficult (see also following sections). But just to secure the registration and deregistration is not enough. After the service discovery phase a user agent and service agent need an authentication between each other to keep their trust. Only authorized network peers should access the registered services. Authentication is needed in first way for the service agent who gets the information that the service was delivered to the right node if authentication was successful. And if a service agent authenticates itself by the user agent, user agent can be sure to trust the delivered service. Specific techniques to solve this problem are for example "Web of Trust" and "Public Key Infrastructure" which will be discussed in section 3.1.

Another important feature is the availability. It is not a secure issue in a first way but a network should also be able to detect broken service providers and delete them from the services list to prevent possible exploitations.

2. SERVICE LOCATION PROTOCOL

The Service Location Protocol (SLP) is an IETF standard published in 1997. It has been superseded by version 2 [Guttman et al., 1999] in 1999. Since its publication, it has seen wide adoption ranging from embedded devices up to enterprise scale applications. This paper will be based on SLPv2, though most of it possibly also applies to SLPv1.

2.1 Protocols basics

SLP is a discovery-only protocol which explicitly leaves the service invocation out. A service is represented by a service description that consists of a Uniform Resource Locator (URL) [Berners-Lee et al., 1994] which uniquely locates the service in a network. Additionally a service description may contain attribute-value pairs.
A UA may query for services by optionally using a Lightweight Directory Access Protocol (LDAP) [Howes, 1997] style filter. A query is sent out via multicast [Armstrong et al., 1992] and answered by all matching SAs. This mode is called multicast convergence. If present, a DA acts as a service cache. In this mode, UAs as well as SAs are required to not communicate over multicast directly, but via the DA employing unicast communication. Messages are sent via User Datagram Protocol (UDP) as long as they do not exceed the Maximum Transmission Unit (MTU). In case of the latter Transmission Control Protocol (TCP) has to be used.
SLP requires SAs to register with it when a DA enters the network. Making SAs aware of newly available DAs is done by sending out multicast beacons by the DA. These messages are called DAAdvertisements.
SLP allows grouping services in scopes. A scope is a string that is part of all messages[1]. SAs and DAs may only answer to queries if configured for the given scope. If no scope is provided, the default scope is automatically applied. SLP specifies no means to learn of all existing scopes, which might lead to regard scoping as a security feature. However, since communication is unencrypted, a simple traffic sniffer allows an attacker to learn of all existing scopes.

2.2 Authentication and Integrity

Regarding security, SLP provides only pre-established trust relationships based on digital signatures using asymmetric keying. This allows DAs, SAs and UAs to authenticate each other.
On top of this, asymmetric keying allows to verify message integrity by all parties. Though selected parts of the message only are included in the signature.
The trust relationship between SLP agents is established by the network administrator who supplies the agents with the correct public and private keys. A key distribution protocol is not part of SLP.
Authentication is used during service registration and registration cancellation. The same signature is needed to cancel a service registration, which has been used to register

[1]Except service Request of type "service:directory-agent" and "service:service-agent"

the service description initially. Incremental service registrations are an optional feature in SLP that allows an SA to incrementally update a service description. RFC 2608 leaves out if incremental service registrations are required to come from the same signature.
The built-in algorithm is Digital Signature Algorithm (DSA) [Kravitz, 1993] with Secure Hash Algorithm 1 (SHA1) [Eastlake 3rd and Jones, 2001] used for hashing, though other algorithms are possible as vendor extensions.

2.3 Replay Prevention

"A replay attack is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed" [Wikipedia, 2009]. To prevent such replay attacks, SLP timestamps each signed message with a 32-bit unsigned fixed-point number UNIX time. The time stamp indicates when the signature expires. It is not part of the signed data and thus can be tampered with. Using a time stamp also does not fully prevent replay attacks as it leaves a door open to replay attack for the duration of the signature life-time. Hence a nonce[2] that is to be included in the initial request (Service-/AttributeRequest) and signed by the responder undeniably associates the request with one response. However, we will show in later sections of this paper (see 3.2) that replay attacks will be taken care of by the newly introduced encryption methods.

2.4 Availability

"Availability can be defined as the property of a system which always honors any legitimate requests by authorized entities." [Cotroneo et al., 2004]. SLP only has weak countermeasures to answer to availability attacks. First it employs multicast messaging to make the communication more robust. Second, multiple DAs can be deployed per network to further scalability and robustness of operation. DAs may redundantly store service information for all SAs, in case one of the DAs fails. Database replication between DAs is not part of RFC 2608 itself, but of a mesh enhancement to SLP [Zhao et al., 2003].
Neither message rate-limiting nor throttling is defined in SLP. Thus flooding attacks are possible where an attacker spams the network with maliciously messages like queries or service replies.

2.5 Confidentiality and Authorization

As shown in 2.1, SLP does not allow messages to be encrypted. This means that confidentiality cannot be enforced with SLP. An attacker might simply eavesdrop on the network traffic and learn about existing services over time. This does not even require a UA actively querying for services, because SAs periodically re-register a service when its lifetime expires. The same reason also renders any kind of UA authorization useless. Restricting UAs to certain service registrations without proper service announcement encryption is pointless.
A poor mans version to authorization can be achieved with scoping.

3. SECURE SLP

[2]Random, non-guessable and externally non-influenceable data

Having identified confidentiality and authorization (see 2.5) as essential security features in open networks [Cotroneo et al., 2004, Hollick, 2001], this paper will outline ways to secure SLP. These protocol extensions are to be backward compatible and secured agents are to be deployable in existing networks incrementally.

3.1 Web of Trust or Public Key Infrastructure and reputation-based trust

Before we can start implementing confidentiality, we need to focus on the trust relationships in SLP first. As shown in 2.2, SLP only supports pre-established asymmetric keys as means to trust. While this rather simplistic approach is acceptable in centrally managed networks like enterprise LANs, it is not for open networks. An open network qualifies itself as a network of nomadic devices without prior knowledge of each other. Thus mechanisms are needed that can establish trust between strangers.
Since SLP already comes with support for X.509 certificates, it appears to be easiest to base trust on such keys and just eliminate the need to manually set them up and implement proper key distribution protocols instead.
This has been addressed by at least two well known solutions:

Web of Trust

Web of Trust (WOT) is a concept to create trust between peers in a network and is an alternative to a Public Key Infrastructure model. WOT is based on a decentralized structure so there is no central authority needed and it is operating with public-key cryptography. Both are parts of an open network using service discovery with SLP. In Web of Trust a user A establishes trust to user B while sign B's public key with his private key. In that way other users can verify that A is trusting B. Trust in WOT has also a transitive relation. That means if user A trust user B then user A automatically trust everyone trusted by user B. Problem arises if a user revokes his trust. In that case other network peers don't get this information immediately like in PKI. So a potential vicious user can act at least a short time as a trustworthy person. This kind of trust would work well while SLP is in directory-less mode.

Public Key Infrastructure

Public Key Infrastructure is a concept to create trust between peers in a network. It is based on public-key cryptography and provides a centralized architecture. PKI requires at least one server which has to be reachable all the time and which has to provide several instances (registration, certificate and validation authority) so other users can request new and/or verify other certificates in real time. In some cases it is possible but quite difficult to provide a PKI in an open network so alternatives like Web of Trust are needed. To manage a PKI in an open network a best possibility is to have internet access, so the peers can use already available PKIs or to have at least one fixed and foremost trustworthy peer who could act as a server. Otherwise PKIs are nonsensically in an open network. This kind of trust requires a centralized structure, so it would just work while SLP is in directory-based mode. Both technologies solve the key distribution protocol successfully. However each has its own shortcoming in open networks.

To balance off said shortcomings a reputation-based trust model may be used on top of static key-based trust models. Reputation-based trust takes the agent behavior[1] over time into account. It then uses its behavior as input parameters for a continuous function that marks trustworthiness, indifference or mistrust of the agent. The key-based model is used to bootstrap the reputation-based model by means of recommendation. A detailed definition of a reputation-based trust model can be found in Secure Pervasive Discovery Protocol (SPDP) [Almenarez and Campo, 2003].
However in cases where devices are resource constrained by battery lifetime, a reputation based trust model might not be feasible at all. In order to measure peer behavior, a device needs to constantly monitor the network or listen for reputation related trust notifications by other peers. This prevents the device from hibernating to save energy. Moving this functionality off to infrastructure services is only possible if the open network provides such services.

3.2 Confidentiality via Security Groups

Once a reliable and usable trust relationship has been established we can use it to create Security Groups (SGs) among those agents. SGs have been proposed by Hollick [2001]. A SG is a group of agents which share a common secret (symmetric key). This key is used to encrypt all communication between each other. Due to the fact that the same two peers might be part of different SGs, the key association cannot be based on the sender address alone[2]. Therefore complete encryption of the SLP message payload would not only cause severe performance penalties for non SG members, but force group members to try all associated keys in worst case scenarios. Thus Hollick [2001] suggests to revert to Internet layer encryption by using Internet Protocol Security (IPSec) [Kent and Seo, 2005] for peer communication. Group communication is left open though. This paper takes a different approach and uses application layer encryption on top of Internet layer multicast. This alleviates the network requirements and handles security within SLP entirely. Additionally this allows reusing multicast encryption for unicast channels as well[3].
Huang et al. [2007] discuss various approaches to secure group communication with multicast. All of these Group Key Agreement protocols (GKA) suffer from two basic security implications that differ from point-to-point communication [Prakash and Uthariaraj, 2008]:

One affects all The compromise of a single group member affects the security of the whole group

Re-keying on leave/join No past/future data is allowed to be decrypt-able by future/past group members

In the scope of this paper the first problem can only be addressed by allowing a group member to request a group key renewal once a compromise has been detected. On the other hand this might open the door to availability attacks if a group member spams the SG with re-keying requests. A viable countermeasure to would be a burst rate that limits a peer's capability of request key renewal.
Regarding the second security implication, this paper argues that it can be relaxed in the scope of SLP to simplify the encryption protocol overhead significantly. Unless a new SA joins a SG or an existing SA incrementally updates its service description[4], the SG's data is stale. A former group member does not learn anything new. Thus, only a change of a SA membership requires re-keying to exchange the group key. This also limits the effects of a compromise of group member (in at least very active) SGs where constant rekeying occurs regularly.
Open networks pose two more constraints on the group encryption protocol. The protocol must not require group members to know each other. Nor is a centralized architecture acceptable as it would introduce a single point of failure.
An agent has to be trusted by a member of the security group to join it. Optionally authorization can be enforced during group joining. SGs can be created by all agent types, though since advertising a SG has to be made with traditional SLP, an initiator of a SG always needs to assume the role of a SA anyway.

3.3 Initiating a Security Group

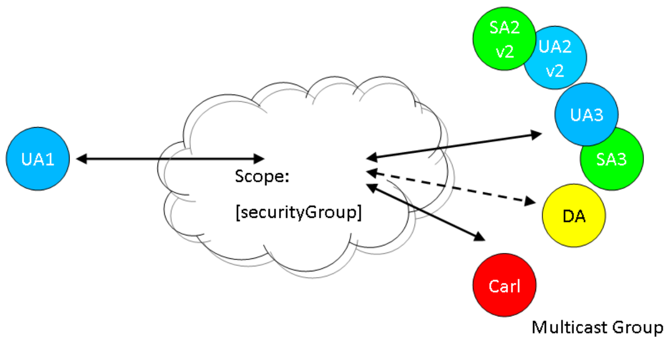

Figure 3.1: A dedicated scope denoting security groups. Since SG-advertisements are sent out using unencrypted SLPv2, attackers (Carl) are capable to discover SGs. However this does not pose a security threat.

Introducing SGs poses a chicken and egg problem on SecureSLP. How does a peer learn of the existence of a SG? Fortunately the functionality provided by traditional SLPv2 can be leveraged to advertise SGs via unencrypted service advertisements. However this allows an attacker to discover all available SGs. While the attacker will not be able to join the SG, it is essential that the advertisement does not leak confidential information as part of the service description. The only information which is essentially required is an SG identification that is best represented by public-key of the SG initiator.
Using a special service type that denotes a SG is one possibility how to make SGs discoverable by agents. This paper

[1]E.g. amount of network messages sent
[2]Unless each SG uses a unique multicast group and or port
[3]Unicast may reuse the group share secret in SLP

[4]Subsequently incremental updates will be seen as SG rejoins

however takes a different approach and assigns a new keyword that marks a dedicated scope for SGs. This is favorable to a special service type since it practically hides SGs from legacy SLPv2 agents that would not be able to interact with a SG anyway (see 2.1). Figure 3.1 depicts SG advertisement. The right side shows agents which are part of the same multicast group with the UA1 that queries for SGs. Agents marked with *v2* ignore the query as it is not send to the default scope and are both not configured for the special scope explicitly.

3.4 Security Groups and Directory Agents

As stated earlier in this paper, DAs cause security implications for the security of SLP, which have to be accounted for. Hollick [2001] does not address this topic and simply assumes the non-existence of DAs. This assumption is untenable in SLP as the fallback to DA mode is built into the protocol and UAs and SAs are required to use a DA if present (see 2.1). Even though a forged DA would not be able to tamper with service descriptions due to integrity checks, it can prevent SGs from being established by silently discarding SG advertisements. Therefore SLP agents must revert to multicast convergence if a DA cannot be authenticated to be legit.
In case a DA is to be used together with SGs, it has to become a member of each and every group in the network. Otherwise it will not be able to decrypt the service description and answer queries sent by UAs.

The previous requirement and the ones listed in 3.2 make the Group Diffie-Hellman (GDH) outlined in Bhaskar et al. [2007] a good candidate as a GKA for SecureSLP.

4. CONCLUSION AND FUTURE WORK

Open networks become increasingly common due to the wide availability of mobile device technology. When such a device enters a network, it wants to learn about the services offered by other peers or advertise its own services. Service discovery protocols have been around to address this requirement for a while. A most prominent and widely adopted one is SLP. However, SLP has not been designed for open networks upfront. Thus the protocol has to undergo a renewal to make it fit for open networks.
This paper describes the fundamental security implications that arise from open networks. It picks up these implications in the scope of SLP and compiles a detailed threat analysis for SLP in open networks. It then continues to extend the current version of SLP with security enhancement to strengthen the protocol for use in untrustworthy and hostile environments by staying backward compatibility with earlier protocol versions. The major protocol modifications can be summarized as:

- SLPv2's pre-established trust model is replaced by a dynamic model that can deal with the dynamics of an open network. It is either implemented as a Public Key Infrastructure or a Web of Trust. On top a reputation based trust may be used when devices are not resource constrained by e.g. battery lifetime.
- Confidentiality is added to SLP by encrypted group and peer to peer communication. The symmetric group key is handled by using Group Diffie-Hellman, a known protocol for distributed group key agreement. This goes beyond the approach taken by Hollick [2001] who leaves confidentiality to the Internet layer. Thus our SecureSLP stays independent of additional network facilities.
- SecureSLP is aware of all three agent types in SLP namely User Agents, Service Agents and Directory Agents. Where Hollick [2001] excludes Directory Agents in his solution entirely, SecureSLP incorporates DAs into the protocol. This enables better performance and scalability when used in combination with the enhancements presented by Zhao et al. [2003].

Table 4.1 shows a comparison between traditional SLP and the secured version of SLP as proposed in this paper.

	SLP	SecureSLP
Authentication	+	+
Integrity	+	+
Confidentiality	−	+
Replay prevention	−	+
Authorization	−	○
Availability	−	−
Non-repudiation	−	−

Table 4.1: SLP and SecureSLP security comparision matrix

With confidentiality being addressed in SecureSLP, peer authorization becomes feasible, to support different discovery results based on an UA authorization. Different authorization levels may be represented by a dedicated Security Group per level and Service Agent. However, more research has to be undertaken in order to validate this approach and add a concrete implementation to SecureSLP.
Properties like non-repudiation and availability are regarded as unessential for SecureSLP in the scope of this paper. Whether this assumption holds true and e.g. message loss and agent unavailability is indeed tolerated by the protocol, has to be confirmed in future work. Even more important is concise performance and scalability measurements to prove that the protocol extensions maintain SLP's performance characteristics even in large open networks with many peers and strong fluctuations.

5. CONCEPTS FOR IMPLEMENTATION

In chapters above we discussed some possible security concepts and increments for the service location protocol. In this section we will introduce our practical solutions to fulfill the requested requirements.
First of all we need to establish security groups, where users can communicate using encryption. To create those groups we combined the already implemented SLP scopes with a group key agreement protocol. For this purpose we took the Tree-based Group Diffie-Hellman protocol (TGDH). TGDH is a group key agreement protocol which we use to create and share security group keys (details about TGDH will be introduced in section 5.2). We will describe why we picked this protocol and how we integrated it into the SLP.
Further we will discuss possible attacks on SLP which uses TGDH and the solutions we made to prevent them. (nachher etwas ausführlicher beschreiben)

5.1 Security Groups

Like discussed above we propose to create security groups in an open network to establish security between some users and still offer all SLP functions there. To create a secured group within an open network we worked out a concept and a new workflow for our SecuredSLP. First of all a security group in SecuredSLP is a SLP scope in which the group members use encryption to communicate with each other, which means the group members all share a secret. To create and share a secret we used the TGDH protocol. There are some requirements to a security group which we should look at:

- To join a secured group a user needs to know that there is such a group and this information should be announced as a plain text message, otherwise it isn't possible to get knowlege about such a group. The message should contain at least the information about the group name and where to get access to this group.
- After a user is allowed to join a group, he requires the shared secret of this group, the group key (see also sections 5.2 and 7.2).
- With a valid group key the user can decrypt all announced service information or offer services itself.
- If a user leaves a secured group, reasons could be for example an expired session key, the secured group key should be refreshed automatically.

We choose TGDH because it fulfill all the requirements above and we already found an implementation in java for it. But it is not necessary to use TGDH, so any other group key agreement protocol would work as well.
With TGDH each user within a secured group has several cryptographic keys.
überarbeiten!!!
Group Shared Key (GSK): This key is same for all group members and allows to access the secured group.
Private-/Public-Key: This is a key-pair that every user has and with those the users encrypt or decrypt their messages.

5.2 Tree-based Group Diffie-Hellman (TGDH)

Tree-based Group Diffie-Hellman is a protocol-suite for group key management. It handles the key distribution between all network members. TGDH is based on a binary tree structure. But the tree structure is just logical and isn't connected with the real position of the network members. This protocol-suite handles several dynamic group events like a new member joins or leaves the group and two networks merge together or one network partitions in several networks. So TGDH implements four protocols: *join*, *leave*, *merge* and *partition*[1]. But all of these protocols have some common structures with following features:

- Each network member computes the *group session key* out of the *group key* with a hash-function (the hash-function is the same for all members).
- Each *member session key* is just known to the member himself and shouldn't be published.
- In case a group gets larger, the *session keys* of new members will be included and some of old members have to refresh their *session key*.
- In case a group gets smaller, the *session keys* of the members, who left the group, will be deleted and at least one member of the group have to refresh his *session key*.
- All protocol-messages are signed by the sender. For the digital signature TGDH uses RSA or DSA with SHA-1 hash-function.

After any changes each member refreshes his *key-tree* independently from each other.

5.2.1 Keys in TGDH

There are several cryptographic keys which are used in TGDH.

Group key $K_{<0,0>}$ which is represented as the root of the TGDH tree (compare with figure 5.1).

Group session key K_{group} which is derived from the group key. $K_{group} = h(K_{<0,0>})$, where $h()$ is a cryptographic hash-function.

Member session key K_i is a session key for the member M_i.

Blinded key BK_i is a key which can be calculated from the member session key K_i with the function $BK_i = f(K_i)$, where $f() = g^k \ mod \ p$ with g as generator and p as a prime.

Furthermore there are also key sets a member has knowledge about. Each member knows all keys and their corresponding blinded keys in his path from the leaf to the root in the tree. For example member M_2 in the figure 5.1 knows the set of

[1]To secure SLP we only use the join and leave protocols. For that reason the other protocols won't be discussed in detail.

keys $\{K_{<2,1>}, K_{<1,0>}, K_{<0,0>}\}$ and the set of blinded keys $\{BK_{<2,1>}, BK_{<1,0>}, BK_{<0,0>}\}$. With that information it is possible to compute any other key:

$$\begin{aligned} K_{<l,v>} &= (BK_{<l+1,2v+1>})^{K_{<l+1,2v>}} \ mod\ p \\ &= (BK_{<l+1,2v>})^{K_{<l+1,2v+1>}} \ mod\ p \\ &= g^{K_{<l+1,2v>}K_{<l+1,2v+1>}} \ mod\ p \\ &= f(K_{<l+1,2v>}K_{<l+1,2v+1>}) \end{aligned}$$

It is also possible to compute the group key out of that information. For member M_2 the calculation would be:

$$\begin{aligned} K_{<0,0>} &= (BK_{<1,1>})^{K_{<1,0>}} \ mod\ p \\ &= (BK_{<1,1>})^{(BK_{<2,1>})^{K_{<2,0>}}} \ mod\ p \end{aligned}$$

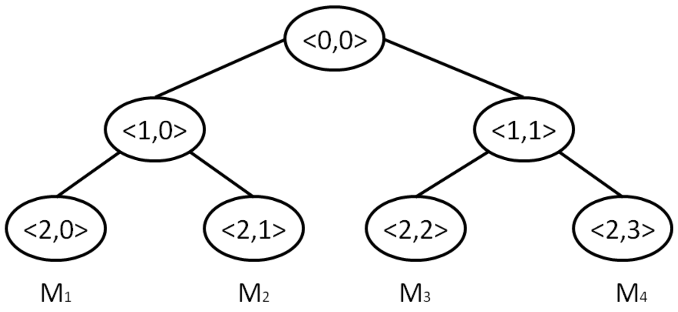

Figure 5.1: Example of a tree structure in TGDH

5.2.2 Join protocol

Assume a group with three members $\{M_1, M_2, M_3\}$ and a new member M_4 who wants to join this group. M_4 initialize the join protocol by sending a *JoinMessage* to the group via multicast. The message contains the blinded key of M_4. First of all the join position for M_4 is calculated and a *sponsor* is chose. The sponsor is the member whose leaf is placed on the insert position of the new member. In this case all members insert a new joint and a new leaf in their tree and delete all blinded keys in the path of the sponsor. Additionally the sponsor generates the new group session key and all keys and blinded keys in his path. Finally the sponsor sends the new tree $\widehat{T}$ and a set of all blinded keys via a multicast message. Member M_1 and M_2 can just calculate the group key after they received the new tree $\widehat{T}$. The tree update is shown in figure 5.2.

5.2.3 Leave protocol

To leave the group a member sends a *LeaveMessage* via multicast to the group. Same as in join protocol, a sponsor is chose. All members remove the leaf and his father joint from the tree. Also they remove all keys and blinded keys in the corresponding path. Additionally the sponsor generates new session key and computes all keys and blinded keys in his path. In the end sponsor sends the new tree $\widehat{T}$ and a set of all blinded keys via broadcast to the group. After receiving the new tree, all other members are able to compute the new group key. The corresponding tree update is illustrated in figure 5.3.

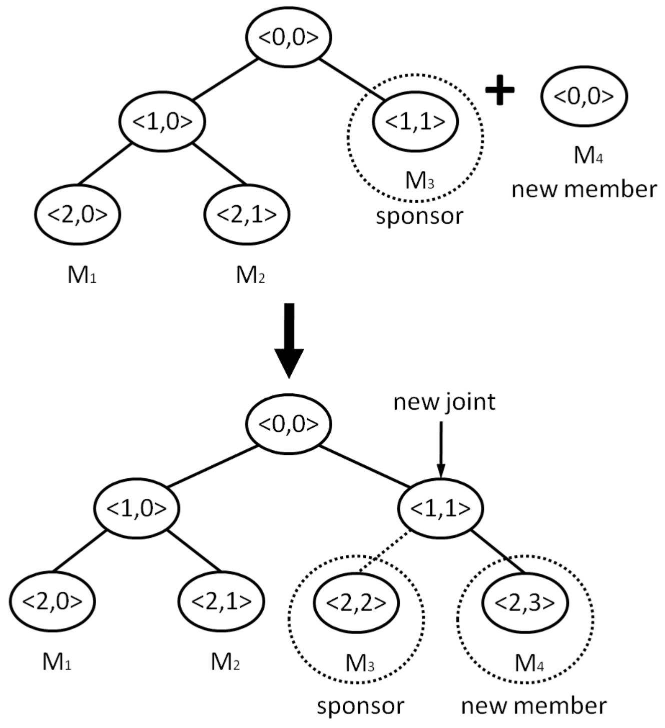

Figure 5.2: Tree update after a join of a new member

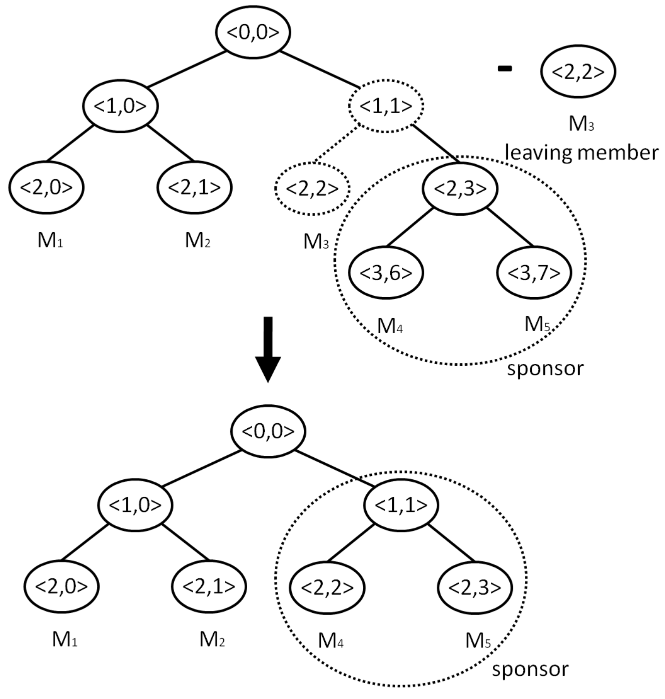

Figure 5.3: Tree update after a member left the group

6. INTEGRATION OF A GROUP KEY AGREEMENT PROTOCOL INTO SLP

There are two main approaches how to integrate a group key agreement protocol into the SLP. It is possible to integrate the group key agreement hardcoded into SLP or as a module adjacent to SLP. Both got their benefits and disadvantages.

6.1 Hardcoded integration

With this solution we have to take the SLP source code apart and change many methods to combine a group key agreement protocol with SLP. That could be a neatly solution but would change the SLP too much and the implementation would be much complicated and expensive in time. The SLP probably wouldn't be compatible to the versions below and that isn't our goal. This approach would deliver a new protocol which probably would scale better as the modular integration solution. eventuell paar Einzelheiten ergänzen

6.2 Modular integration

With this solution we don't make many changes in the SLP source code. We use a group key agreement protocol as a module which we combine with SLP without fully integrates it into the SLP source code. In this way the module can easily be replaced anytime. So we are not bound to a special group agreement protocol. Also it is possible to use many several group agreement protocols at the same time if needed.
For this purpose we have minor changes in the source code. Like discussed above we took TGDH as our group agreement protocol. To make SLP work with TGDH we had to change the header of SLP messages and implement additional functions into SLP to communicate with our group agreement protocol. hier vielleicht mehr details

6.2.1 Workflow of SLP with a modular integration

hier noch ein Bild einfügen

7. IMPLEMENTATION

To combine SLP with our group key agreement protocol we changed/added following protocol properties:

- SLP message
- Key enchantment
- Communication

7.1 SecuredSLP message format

The header of a SLP message was modified to distinguish SLP messages from SecuredSLP messages. We added the S, `Security Group Length` and `Security Group Name` flags into the header and changed the `Version` of SLP in the source code (compare figure 7.1 and figure 7.1).

Version: Is now set to 3.

S: This flag shows that the message body is encrypted.

Security Group Length: This flag specifies the length of the `Security Group Name` string.

Securety Group Name: This flag shows the name of the security group the message belongs to.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    |  Function-ID  |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Length, contd.|O|F|R|       reserved          |Next Ext Offset|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Extension Offset, contd. |              XID              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Language Tag Length      |         Language Tag          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7.1: SLPv2 Header

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    |  Function-ID  |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Length, contd.|O|F|R|S|     reserved          |Next Ext Offset|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Extension Offset, contd. |              XID              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Language Tag Length      |         Language Tag          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Security Group Length    |      Security Group Name      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7.2: SecuredSLP Header

Additionally we use a HMAC (96 Bit) over the header and add it into the encrypted message body so the header and payload are linked with each other. This method is also used in IPSec under the ESP mode and allows us to prevent replay attacks and header manipulation of each SecuredSLP message. See also quellen nachher einfügen

7.2 Key enchantment in SecuredSLP

Like discussed above we use a group key agreement protocol to establish security between group members. To get knowledge about a security group, clients use extra discovery iteration. The information about a security group is announced as plain text; otherwise no one would be able to join such a group. To join a security group a user has to send a join message to the group. Then the group key agreement protocol handles the key distribution. In our implementation we used TGDH, so the distribution works like discussed in section 5.2. After receiving the group key, security group members are able to discover services which are pronounced in this security group or provide services themselves. Also there is a rekeying phase after a user leaves the group. That is important to keep the security under the group members and to avoid that a key gets compromised.

7.3 Communication in SecuredSLP

Due to the additional changes in the protocol it is necessary to take a look at the communication between network members. SecuredSLP still provides unencrypted communication like in SLPv2 so it is still possible to create or join a unsafe SLP scope. In that case the communication works like in SLPv2 (compare section 1) without a group agreement protocol. In case SecuredSLP tries to establish connection to a security group the communication can be described as following:

1. User discovers a security group and sends a *JoinMessage* to the group key agreement protocol (TGDH in our case)
2. Group key agreement protocol initializes the rekeying

phase for the security group and computes new group key.

3. New member receives the group key and is able to communicate with the other security group members. Compare with figure 7.3

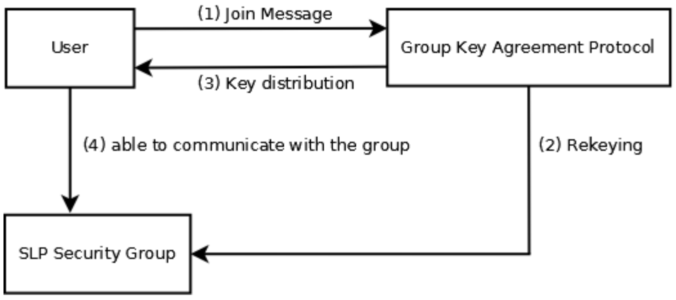

Figure 7.3: Communication sequence of a user joining a SecuredSLP security group

Like told above the group agreement protocol is just a module adjacent to SLP and works asynchronous with it. Hence the key distribution is more complicated and fault-prone. Also there is just one valid key in the network at time so most problems arrive if a rekeying phase is initialize and there are still messages in the network encrypted with the old group key. Those messages will be refused by SecuredSLP.

8. POSSIBLE ATTACKS ON SECUREDSLP

In this section we want to discuss which general attacks are possible on SecuredSLP and how they are prevented or why they can't be prevented.

8.1 Denial of Service (DoS) attacks

Like all distributed systems SecuredSLP suffers from DoS attacks. In SecuredSLP there are two attacking points for a DoS attack. The first one is to overfeed the UA/SA/DA with information or requests. In this scenario an attacker requests for example information about a service from a DA and prevents the usage of that DA by other network members. The seconds attacking point appears just if the secure mode gibts dafür einen extra namen? of SecuredSLP is enabled and a group key agreement protocol is used. In that case an attacker could initiate join/leave messages so the network would stuck in the rekeying phase. This problem is especially present in the case of TGDH as the group key agreement protocol. Each time a user joins or leaves the network the TGDH initiates a rekeying phase so an attacker could exploit this feature.
There is no neat way to prevent attacks on UA/SA/DA because that's their purpose to send or answer requests. It would be possible to drop messages from a user who sends too many requests but this is not part of our paper so it won't be discussed here in detail. To prevent or reduce DoS attacks on the group key agreement protocol it is possible to use another protocol as TGDH. The keys could also be distributed by a network administrator (if possible) so there wouldn't be rekeying phase all the time. In case of rekeying the administrator would handle the distribution of the new keys. This solution is just possible in closed networks and the rekeying phase would take much longer and would also be an extra work.

8.2 Replay attacks

In SecuredSLP a possibility of a replay attacks is very low. To prevent replay attacks we made sure that the header of each message can't be manipulated. Therefor we make a HMAC over the message header and include it into the message which is encrypted. A SecuredSLP message contains a `XID` which marks the message and binds it to a sequence of messages (compare with figure 7.1) and the payload is encrypted and signed by the sender. That prevents that an attacker can change the header information without the receptor getting knowledge about. The only replay attack would be to use a current message to overfeed a peer with it which could count as a DoS attack (see section 8.1).

8.3 Evedrop attacks

Evedrop attacks are still possible on the SecuredSLP. Everyone can log the networktraffic and learn about its behavior. The benifit in SecuredSLP is that the information stays confidentiality because all messages are encrypted and signed. An attacker can just intercept messages but not decrypt them as we assume the encpyrtion algorithms are safe.

8.4 Manipulation attacks

Like told above we exclude the manipulation of SecuredSLP messaged because the messages are encrypted and signed and the header of each message is linked with the payload using HMAC.

8.5 Man-in-the-middle attacks

Man-in-the-middle attacks still work on SecuredSLP. For example an attacker can act as a DA in a security group and communicate with his prey and get information about the services he provides or use.

9. EVALUATION

References

Florina Almenarez and Celeste Campo. SPDP: A Secure Service Discovery Protocol for Ad-hoc Networks. In *In Workshop on Next Generation Networks*, pages 9–9. Springer Verlag, 2003. doi: 10.1.1.124.8021.

S. Armstrong, A. Freier, and K. Marzullo. Multicast Transport Protocol. RFC 1301 (Informational), February 1992. URL http://www.ietf.org/rfc/rfc1301.txt.

T. Berners-Lee, L. Masinter, and M. McCahill. Uniform Resource Locators (URL). RFC 1738 (Proposed Standard), December 1994. URL http://www.ietf.org/rfc/rfc1738.txt. Obsoleted by RFCs 4248, 4266, updated by RFCs 1808, 2368, 2396, 3986.

Raghav Bhaskar, Daniel Augot, Cedric Adjih, Paul Muhlethaler, and Saadi Boudjit. AGDH (Asymmetric Group Diffie Hellman): An Efficient and Dynamic Group Key Agreement Protocol for Ad hoc Networks. In *New Technolgies, Mobility and Security (NTMS) conference*, 2007. URL http://research.microsoft.com/en-us/um/people/rbhaskar/papers/ntms07.pdf.

Domenico Cotroneo, Almerindo Graziano, and Stefano Russo. Security requirements in service oriented architectures for ubiquitous computing. In *MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, pages 172–177, New York, NY, USA, 2004. ACM. ISBN 1-58113-951-9. doi: 10.1145/1028509.1028522.

D. Eastlake 3rd and P. Jones. US Secure Hash Algorithm 1 (SHA1). RFC 3174 (Informational), September 2001. URL http://www.ietf.org/rfc/rfc3174.txt. Updated by RFC 4634.

P2P Foundation. Free and open network definition, 2009. URL http://p2pfoundation.net/Free_and_Open_Network_Definition?title=Free_and_Open_Network_Definition&oldid=24960. [Online; Stand 1. August 2009].

E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2. RFC 2608 (Proposed Standard), June 1999. URL http://www.ietf.org/rfc/rfc2608.txt. Updated by RFC 3224.

Matthias Hollick. Security awareness in service discovery for multimedia collaboration. In *MM&#38;Sec '01: Proceedings of the 2001 workshop on Multimedia and security*, pages 60–63, New York, NY, USA, 2001. ACM. ISBN 1-58113-393-6. doi: 10.1145/1232454.1232476.

T. Howes. The String Representation of LDAP Search Filters. RFC 2254 (Proposed Standard), December 1997. URL http://www.ietf.org/rfc/rfc2254.txt. Obsoleted by RFCs 4510, 4515, updated by RFC 3377.

Chun-Ying Huang, Yun-Peng Chiu, Kuan-Ta Chen, and Chin-Laung Lei. Secure multicast in dynamic environments. *Comput. Netw.*, 51(10):2805–2817, 2007. ISSN 1389-1286. doi: http://dx.doi.org/10.1016/j.comnet.2006.11.027.

S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), December 2005. URL http://www.ietf.org/rfc/rfc4301.txt.

Kravitz. Digital signature algorithm, July 1993.

A.J. Prakash and V.R. Uthariaraj. Multicast cryptosystem: A cryptosystem for secure multicast communication. pages 119–124, Oct. 2008. doi: 10.1109/NPC.2008.73.

C.N. Ververidis and G.C. Polyzos. Service discovery for mobile ad hoc networks: a survey of issues and techniques. *Communications Surveys & Tutorials, IEEE*, 10(3):30–45, Quarter 2008. ISSN 1553-877X. doi: 10.1109/COMST.2008.4625803.

Wikipedia. Replay attack — wikipedia, the free encyclopedia, 2009. URL http://en.wikipedia.org/w/index.php?title=Replay_attack&oldid=281621146. [Online; accessed 4-April-2009].

W. Zhao, H. Schulzrinne, and E. Guttman. Mesh-enhanced Service Location Protocol (mSLP). RFC 3528 (Experimental), April 2003. URL http://www.ietf.org/rfc/rfc3528.txt.