PAPER *Special Section on Trust, Security and Privacy in Computing and Communication Systems*

JXTAnonym: An anonymity layer for JXTA services messaging*

Marc DOMINGO-PRIETO†a), *Nonmember and* **Joan ARNEDO-MORENO**†b), *Member*

SUMMARY

With the evolution of the P2P research field, new problems, such as those related with information security, have arisen. It is important to provide security mechanisms to P2P systems, since it has already become one of the key issues when evaluating them. However, even though many P2P systems have been adapted to provide a security baseline to their underlying applications, more advanced capabilities are becoming necessary. Specifically, privacy preservation and anonymity are deemed essential to make the information society sustainable. Unfortunately, sometimes, it may be difficult to attain anonymity unless it is included into the system's initial design. The JXTA open protocols specification is a good example of this kind of scenario. This work studies how to provide anonymity to JXTA's architecture in a feasible manner and proposes an extension which allows deployed services to process two-way messaging without disclosing the endpoints' identities to third parties.

key words: *peer-to-peer, security, anonymity, JXTA, Java, onion routing.*

1. Introduction

Just as the popularity of P2P applications has risen, so have concerns regarding their security. As P2P applications move from simple data sharing, for example BitTorrent [1], to a broader spectrum, such as e-learning environments [2], they become more and more sensitive to security threats. Therefore it becomes very important to design P2P frameworks which not only can be easily adapted to a broad set of application scopes, but also take into account an acceptable security baseline. Under today's standards, it is expected that it is possible to deploy some degree of privacy, ensuring that the contents of a message exchange are not revealed to an eavesdropper, and authentication, guaranteeing which is the identity of each endpoint during any message exchange.

Manuscript received January 1, 2010.
Manuscript revised January 1, 2010.

†The author is with the Estudis d'Informàtica, Multimèdia i Telecomunicació. Universitat Oberta de Catalunya. Rambla del Poblenou 156, 08018, Barcelona, Spain

*This work was partially supported by the Spanish MCYT and the FEDER funds under grant TSI2007-65406-C03-03 E-AEGIS and CONSOLIDER CSD2007-00004 "ARES", funded by the Spanish Ministry of Science and Education.

a) E-mail: mdomingopr@uoc.edu
b) E-mail: jarnedo@uoc.edu

DOI: 10.1587/trans.E0.??.1

As P2P systems evolve and become widespread in new scenarios, more advanced security capabilities should be considered. An example of them is message anonymity [3], allowing a peer to send requests to an arbitrary service in such a manner that nobody can determine the endpoints' identities. The need for this feature ranges from everyday situations such as a corporate suggestion box, a peer evaluation form or personal data sharing [4], to those related to much serious topics, such as freedom of speech or whistleblowing. Unfortunately, because of its architecture, P2P systems are specially weak to anonymity attacks unless this capability is included in the system's initial design. Even when providing the aforementioned basic security capabilities, it may be easy for other peers to acquire information by monitoring message flows or intercepting queries routed through them [5]. Consequently, message sources and final services are completely exposed to neighboring peers and super-peers.

JXTA (or "juxtapose") [6] is an example of a P2P system which already considers some basic security capabilities, but not anonymity. Consisting in a set of open protocols that enable the creation and deployment of P2P networks, it provides applications with the capability to easily discover and observe peers, exchange messages and publish resources. Even though in its successive revisions security mechanisms have slowly crept in, up to an acceptable degree [7], in its latest version (2.7RC1), available on January the 12th 2011, no previsions are made about being able to provide any degree of anonymity some day.

In this paper we extend our previous work in [8] and present JXTAnonym, an anonymity layer which allows peers to exchange messages with JXTA services without disclosing the identity of the participating parties to neighboring peers or super-peers. Thus, anonymous two-way messaging is made possible in JXTA. The proposed layer is based on a popular approach within the context of P2P applications, onion routing [9]. However, it is specifically adapted to the idiosyncrasies of JXTA, taking advantage of service access mechanisms already provided by the platform, instead of defining additional protocols. In addition, the amount of required changes on an existing system in order to integrate anonymous messaging is minimized. In that manner, peers which support anonymity may coexist with those who don't, without incompatibilities.

The paper is structured as follows. In Section 2 we provide a brief summary of the current mechanisms based on onion routing, focusing on those methods which provide bidirectional messaging capabilities in P2P networks. Section 3 describes how our base system has been extended to encompass bidirectional messaging, suiting to the idiosyncrasies of JXTA services. The outcome of our experimental results in order to evaluate its performance is provided in Section 4. Section 5 provides a security discussion of JXTAnonym. Finally, Section 6 concludes this paper and outlines further work.

2. Related work on onion routing

Anonymity in P2P networks can be achieved using three different approaches: *Unimessage-based*, *Split message-based* and *Replicated message-based*. Our work focuses on the first one, since it is the most popular and efficient, while maintaining a high degree of anonymity. Also it assumes that nodes are completely autonomous [10]. These characteristics mesh with the structure of JXTA and allows anonymous bidirectional communications.

A unimessage-based approach, also known as *Onion Routing* [9], provides anonymity by sending the message through a random sequence of proxies before it reaches the destination. Those proxies are labeled *OnionRouters*. Generally, the path of OnionRouters is pre-constructed by the sender before the message is sent. The message is repeatedly encrypted in a manner that, during transit towards the destination, a single encryption layer can be taken out, one at a time, at each OnionRouter. Each time a layer is peeled off, the identity of the next hop is obtained. Thus, at each hop, an OnionRouter does not know whether the message is being sent to another proxy or to the actual final destination. In the same manner, the destination peer cannot know whether the received message comes directly from the source or it has been relayed through a set of OnionRouters.

The onion routing approach is also able to provide anonymous two-way communications in two ways:

- Constructing both a query (source to destination) and response (backwards, destination to source) onion at the source peer and including it inside the onion message together with the data to be sent. Therefore the destination peer, after replying the query, can forward its reply data to the source using the response onion path.
- Including a session ID within the onion route, which is stored on transit, thus creating a kind of return path virtual circuit [11]. When the destination peer responds the query, the message is routed back following the established path, but in the opposite direction.

Onion Routing is used as the core of many anonymity protocols to achieve providing mutual anonymity, being *APFS* (Anonymous P2P File Sharing) [11] and *Tor* [12] the ones that share more features when constructing of the anonymous path.

As its name suggests, the APFS protocol is used for P2P file sharing, providing mutual anonymity of the initiator and the responder in a connection. In APFS, each peer has to choose a proxy peer and create a Onion Route to it. This proxy peer will be the entry point to the anonymous network for that peer. Additionally, exists a well known bootstrapping peer called coordinator which maintains a list of all the peers connected to the network and which are acting as a servers. However, instead of peers address, their proxy address are stored.

Tor is a circuit-based anonymous communication service. The sender's anonymity is maintained by constructing a circuit from the source to an exit node, who will communicate with the destination. The construction of this route is performed by incrementally extending the circuit, hop by hop, and negotiating a different symmetric key with each intermediate node. This circuit is periodically recalculated and can be used at its entire length, or shortened to modify the exit node. Answers to queries are transmitted through the same circuit.

Mutual anonymity can be achieved on Tor by using hidden services. The basic idea of hidden services is that the service provider is hidden behind some peers who act as his Introduction Points (*IP*) while the service consumer is behind a Rendezvous Point (*RP*). The communication between these points is done by following circuits. The steps required to use a hidden service are:

- The service consumer contacts with one of the IPs, announcing his RP.
- The IP forwards the message to the service provider, who will create a circuit to the RP
- At this point, mutual anonymity communication between both can be established through the RP.

3. A proposal for JXTA anonymous messaging

JXTA provides mechanisms to share services. Such services are commonly consumed by exchanging bidirectional messages. In order to provide anonymity in services consumption, is important to inspect the JXTA messaging architecture. From this review, it is possible to identify the elements that can be taken advantage of in order to define an anonymity layer for which is transparent and finely integrated to JXTA architecture, without the need of defining additional protocols.

3.1 JXTA Messaging architecture

In this subsection, we provide an overview of those of JXTA's main architectural elements related to our proposal. A detailed explanation can be found in [13].

The main idiosyncrasy in JXTA's design, which sets it apart from other P2P frameworks, is introducing the concept of *Peer Group*, a segmentation of the global JXTA network. All peers publish and consume services within the context of a group, interacting with each other by using some JXTA core services, the most important ones being the *Discovery Service* and the *Pipe Service*.

Every resource in a JXTA group is described by an *Advertisement*, a metadata document. A resource cannot be accessed unless its Advertisement is previously retrieved. Advertisements must be periodically published, since they expire and are flushed from the network after some time (by default, 2 hours). The Discovery Service's responsibility is managing advertisements, allowing peers to publish and find available resources. The most important types of Advertisements are:

- *Peer Advertisement*: Describes a peer and the resources and services it provides. Each peer is responsible for the publication of its own Peer Advertisement, and is considered online only while it continues to do so.
- *Pipe Advertisement*: Describes a *JXTA Pipe*, the main mechanism in JXTA to exchange data between two applications or services.

The Pipe Service is responsible for managing message exchanges using JXTA Pipes. The simplest pipe in JXTA, the *JxtaUnicast*, provides an asynchronous, unidirectional message transfer mechanism which can be easily established and managed. Nevertheless, there is a higher-level communication abstraction provided by the *JxtaBiDiPipe* which provides a bidirectional communication channel. The latter is usually preferred by services, since it allows the direct use of a straightforward query-response exchange. The description of JXTA's standard service model based on this procedure follows:

1. Each service provider starts a *JXTAServerPipe* using the Pipe Service, which exposes and listens to an *input pipe* in order to process communication requests. This input pipe is defined using a Pipe Advertisement.
2. This Pipe Advertisement is made public by the service provider using the Discovery Service.
3. The Advertisement is propagated within the group by *Rendezvous Peers*, special super-peers who efficiently distribute Advertisements.
4. To consume a service, a peer must retrieve the Pipe Advertisement (via the Discovery Service) and then a bidirectional connection must be established via the Pipe Service, creating a JxtaBiDiPipe.
5. Once a communication request is received at the server side, an independent JxtaBiDiPipe is created and bound with the request. A message exchanges may begin from now on.

Message exchanges can be secured in JXTA by using a group based on the *PSE (Personal Security Environment) Membership Service*. Under this kind of peer group, each peer is provided with a credential based on PKIX [14] certificates. This guarantees that each peer has initialized a valid pair of public-private keys and that the public key of each peer is automatically distributed inside its Peer Advertisement, in a special service parameter entry.

3.2 Anonymizing procedure

We propose an anonymity layer that causes the minimum interferences on the JXTA messaging architecture, according to the review done in Section 3.1. JXTAnonym, an anonymizing service, is deployed in those peer group members which want to anonymously exchange messages, creating an anonymous subnetwork within the context of a peer group. Group members may freely join and leave this network.

The proposed service is tailored to JXTA's core services features, such as invisible publication, discovery and access to services (via the Discovery Service), message management via the Pipe Service and usage of the PSE secure environment for cryptographic data generation and distribution. Therefore, the deployment procedure follows the same steps as for any other peer service, making use of JXTA's service model without the need of modifying JXTA's initial design.

The service's anonymity mechanism is based on an onion routing approach, examined in detail in Section 2, protecting the identity of end clients (consumers) and services (providers) from the rest of the group members. In addition, the end service is also unable to establish the end client's identity. Its general architecture is summarized in Figure 1.

The execution of JXTAnonym in any peer encompasses three different steps: *JXTAnonym Service Publication*, *Message Setup* and *Message Processing*.

3.2.1 JXTAnonym Service Publication

Just like any other JXTA service, an instance of the JXTAnonym service in a peer group member listens to incoming queries using an input pipe. This pipe is made available to other peer group members by periodically publishing its Pipe Advertisement, via the Discovery Service. However, we propose that Pipe Advertisements are not published as standalone documents.

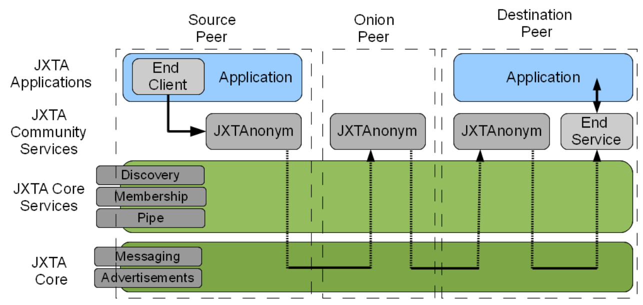

Fig. 1 JXTAnonym service operation in the context of JXTA's architectural design.

They are piggybacked inside the service parameters section of the Peer Advertisement, identified by a hard-coded well-known JXTAnonym service ID. The inclusion of service related-data in this section is also used by some other JXTA services, such as the PSE Membership Service. The main advantage of this approach is that it is guaranteed that the service's Pipe Advertisement is always being published while the peer is online, thus being always available and up to date. In addition, each peer's Peer Advertisement contains all the information required by JXTAnonym: the local service Pipe Advertisement and the peer's cryptographic data.

Once a peer decides to stop participating in the anonymous network, the service parameter entry is removed from the Peer Advertisement, continuing its publication as normal. Due to the JXTA's architecture, this new Advertisement will be propagated across the Peer Group, replacing the old one. If a peer becomes unreachable, its Peer Advertisement is not going to be updated any longer, being flushed after the expiration time is reached. Once this happens, the JXTAnonym Pipe Advertisement will stop being available too, guaranteeing that at any time only active instances of the service are published.

3.2.2 Message Setup

This step initiates the onion routing process and is performed by the peer who wants to anonymously access an end service deployed at a remote peer within the same group.

First of all, we will describe the onion layering procedure in $JXTAnonym$, since it is used at several steps in the message setup process. The layering procedure accepts two parameters: *OnionCore*, which is a JXTA Message structure which may contain any arbitrary data, and $PeerAdvList = Adv_1, \cdots, Adv_n$, an ordered sequence of Peer Advertisements.

For each Peer Advertisement Adv_i, from $n \ldots 1$, the following process is iteratively executed. $Onion_i$ is considered the result of each iteration, being $Onion_1$ the final result:

1. This iteration's input, *inputData*, is chosen.
   a. For the first iteration ($i = n$) the *OnionCore* structure is considered *inputData*.
   b. For the rest of iterations ($i = n-1, \ldots, 1$), the result of previous iteration, $Onion_{i+1}$ is considered *inputData*.
2. The public key PK_i is retrieved from Adv_i's Membership Service definition entry. Under the context of a peer group which implements the PSE Membership Service, it is guaranteed that PK_i actually exists.
3. *inputData* is encrypted using PK_i under a wrapped key scheme [15], generating *EncInputData*.
4. Adv_i's PID field (the peer's unique identifier) is retrieved.
5. A new *OnionLayer* structure is generated by creating a JXTA Message with the following fields:
   - `NextHop` $= PID$
   - `OnionRoute` $= EncInputData$
6. The *OnionLayer* structure becomes this iteration's result ($Onion_i$).

Once the layering procedure has been established, the message setup process description follows:

1. A peer S decides to use a service (*EndService*) executing on peer D.
2. S's corresponding client (*EndClient*) creates a query message ($JXTAQueryMessage$), specifying which is the destination peer (peer D), and a callback structure (*PipeListener*). Such structure is used by JXTA applications to allow asynchronous processing of the incoming response, so they do not need to block until the reply is received.
3. *EndClient* locates the *EndService*'s *JXTABidiPipe* Pipe Advertisement, from now on *EndServicePipe*, using the Discovery Service.
4. So far, steps 1-3 describe the required steps to access a generic JXTA service. At this point, *EndClient* would open a connection using *EndServicePipe* and use it to directly send $JXTAQueryMessage$. Instead, it decides to use the anonymity service, $JXTAnonymSvc$.
5. If an instance of $JXTAnonymSvc$ is being executed at S, an anonymous client ($JXTAnonymClient$) is also available. Such client is invoked, receiving *EndServicePipe*, D's identity, *JXTAQueryMessage* and *PipeListener* as parameters. From now on, $JXTAnonymClient$ will manage the rest of the message setup process. *EndClient* considers message processing completed as far as it is concerned.
6. A random symmetric key ($FinalSymmetricKey$) is generated and used to encrypt both *JXTAQueryMessage* and *EndServicePipe*, obtaining $QueryEncryptedMessage$.

7. A new record is created in a local table *PendingQueries*, containing $FinalSymmetricKey$ and $PipeListener$, the former being the primary key.
8. A set of random data (*RndData*) is generated. The length of this data should be between 2411 and 4614 bytes. The reasons will be explained in detail in Section 5.
9. S generates a $ResponseCore$ structure. This structure is a JXTA message composed of the following name-value pairs:
    - `RandomData` = *RndData*
    - `SymmetricKey` = $FinalSymmetricKey$
10. At this point, a number of OnionRouter peers must be chosen in order to create a response path. It is considered that 3 hops is good enough [12]. This is done by using the Discovery Service to get a set of Peer Advertisements $RP = Adv_1, Adv_2, Adv_3, Adv_4$ where is it true that $\forall Adv_i, (i < 4), Adv_i \neq Adv_S \& Adv_i \neq Adv_D$ and $Adv_4 = Adv_S$, and in all of them the $JXTAnonymSvc$ service parameter field exists (all of them deploy *JXTAnonym*). It is worth remarking that, since S holds the end client waiting for the response, thats the reason why it must be the last peer in the response path.
11. Once the response path is established, an onion structure *ResponseOnion* is created using the layering procedure previously described. The input parameters are $ResponseCore$ and RP.
12. From *ResponseOnion* the NextHop (*ResponseFirstHop*) and the OnionRoute (*ResponseRoute*) fields are extracted.
13. A *QueryCore* structure is created. This structure is a JXTA Message composed of the following name-value pairs:
    - `NextHop` = $ResponseFirstHop$
    - `OnionRoute` = *ResponseRoute*
    - `SymmetricKey` = $FinalSymmetricKey$
14. Now the query path, QP must be generated. This process is identical to step 10, but in this case, $Adv_4 = Adv_D$ instead of Adv_S. Ideally, $RP \neq QP$.
15. Once the query path is established, an onion structure *QueryOnion* is created using the layering procedure previously described. The input parameters are $QueryCore$ and QP.
16. From *QueryOnion* the NextHop (*QueryFirstHop*) and the OnionRoute (*QueryRoute*) fields are extracted.
17. An $OnionMessage$ structure is generated. This structure is a JXTA Message composed of the following name-value pairs:
    - `OnionRoute` = *QueryRoute*
    - `EncryptedMessage` =*QueryEncryptedMessage*
18. The Peer Advertisement of $QueryFirstHop$ is obtained via the DiscoveryService, and its JXTAnonymSvc Pipe Advertisement extracted. A new connection to this pipe is created and *OnionMessage* sent through it.

3.2.3 Message Processing

This step is performed when an anonymous message is received at any peer that has deployed $JXTAnonymSvc$. The process starts when a running instance of $JXTAnonymSvc$ receives an incoming message ($OnionMessage$), which contains a $EncryptedMessage$ and *OnionRoute* fields. Then, the value in the *OnionRoute* field is extracted and decrypted using the peer's local private key, accessible using JXTA's PSE Membership service. At this point, three things may happen:

1. The extracted data becomes an *OnionLayer* (see Figure 2). Thus, the message must be routed to another peer.
    a. The values stored in the $NextHop$ and *OnionRoute* fields, respectively PID and *nextRoute*, are extracted.
    b. A new $OnionMessage$ structure is generated, where:
        - `OnionRoute` = *nextRoute*
        - `EncryptedMessage` =*EncryptedMessage*
    c. Using the Discovery Service, PID's Peer Advertisement is located. From this advertisement, the $JXTAnonymSvc$ service Pipe Advertisement is extracted.
    d. A new connection to the next hop is established using the previously recovered Pipe Advertisement. The newly generated $OnionMessage$ is sent through it.

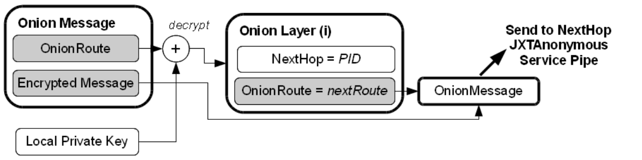

Fig. 2 Processing of OnionMessage at an Onion Peer

2. The extracted data becomes a *QueryCore* (see Figure 3). Thus, the current peer is the destination, D, and holds *EndService*. The query must be processed by this service. In this scenario, the $EncryptedMessage$ field holds the value $QueryEncryptedMessage$.
    a. The values stored in the $NextHop$, *OnionRoute* and *SymmetricKey* fields, *ResponseFirstHop*, *ResponseRoute* and *FinalSymmetricKey* respectively, are extracted.

b. $QueryEncryptedMessage$ is decrypted using $FinalSymmetricKey$, obtaining the original $JXTAQueryMessage$ and $EndServicePipe$.
c. A bidirectional connection is established to the *EndService* using *EndServicePipe* and the client's query, $JXTAQueryMessage$, is sent through it. As far as the *EndService* is concerned, a normal connection has been established. It is oblivious to the anonymity layer.
d. The process waits until a response, *JXTAResponseMessage*, is received.
e. The response is encrypted using *FinalSymmetricKey*, generating *ResponseEncryptedMessage*.
f. A new $OnionMessage$ structure is generated, where:

- `OnionRoute` = $ResponseRoute$
- `EncryptedMessage` =*ResponseEncryptedMessage*

g. $OnionMessage$ is sent to the JXTAnonym's pipe of $ResponseFirstHop$.

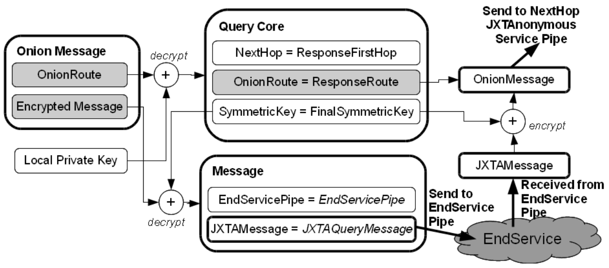
Onion Message
OnionRoute
Encrypted Message
Local Private Key
decrypt
decrypt
Query Core
NextHop = ResponseFirstHop
OnionRoute = ResponseRoute
SymmetricKey = FinalSymmetricKey
Message
EndServicePipe = EndServicePipe
JXTAMessage = JXTAQueryMessage
Send to NextHop JXTAnonymous Service Pipe
OnionMessage
encrypt
JXTAMessage
Received from EndService Pipe
Send to EndService Pipe
EndService

Fig. 3 Processing of OnionMessage at Peer D

3. The extracted data becomes in a *ResponseCore* (see Figure 4). Thus, the current peer is the source, S. The query-response has completed its round-trip. In this scenario, the $EncryptedMessage$ field holds the value $ResponseEncryptedMessage$.

a. The value stored in the *SymmetricKey* field is extracted ($FinalSymmetricKey$). The *RandomData* field is ignored.
b. $ResponseEncryptedMessage$ is decrypted using $FinalSymmetricKey$, obtaining the original $JXTAResponseMessage$.
c. The *PendingQueries* local table is searched for the record which uses *FinalSymmetricKey* as its primary key.
d. Using the *PipeListener* structure, the message is provided to the original *EndClient*. As far as the end client is concerned, JXTA has acted just like during any standard service access. The anonymity layer is invisible.

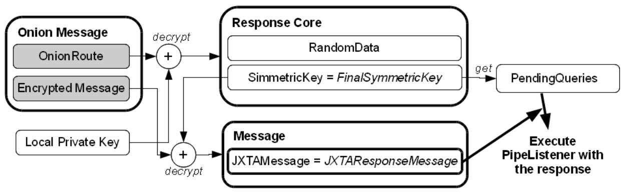

Fig. 4 Processing of OnionMessage at Peer S

e. The record is deleted from the *PendingQueries* table.

4. Experimental result evaluation

As a proof-of-concept, we have implemented our proposal and deployed it on a private PlanetLab-based network. We report the key features of our implementation and its performance in order to assess the feasibility of deploying anonymous messaging in a JXTA network.

The testbed deployed a JXTA peer group consisting in a single node operating as a Rendezvous Peer and 32 nodes as a Edge Peers. This is considered a typical group [16]. This peer group was based on the PSE Membership Service and all Edge Peers deployed the *JXTAnonym* service, as well as an additional test end service called *EchoService*, which was anonymized. This service basically replies with the same content as the query.

At this point, some tests were conducted in order to test the performance of using JXTA services with and without anonymity. The goal of these tests is not finding a performance improvement when using JXTAnonym, since applying security is expected to always result in a performance overhead. The goal is to determine if whether an onion routing approach is feasible in a JXTA network, a peer-to-peer framework developed without considering anonymity in its design and architecture. With feasible we refer that the produced overhead is acceptable compared with the benefits it provides.

Our set of tests measure the average time it takes to consume a service, from the moment the query is sent until the response is received, and the percentage of lost queries during transit. This test was executed in three different scenarios and with two different messages loads. The two different message loads consist in sending messages at maximum speed from just one peer or from all peers at once, evaluating the network behavior when it is very saturated. In both tests up to 250 messages are sent.

The first scenario consists in directly consuming the EchoService through a JXTABiDiPipe, as it is usually done in JXTA. This measure is the expected time during JXTA's normal operation. The second scenario sends the queries and the responses to the EchoServices through three intermediates nodes, as JX-

TAnonym would do, but without using any kind of security. This scenario is called 8 hops. Using this test, we can discern which part of the overhead produced by anonymity is due to multiple hops and which is produced by additional computing and transmission costs at each onion peer. The third one evaluates the cost of anonymizing EchoService using JXTAnonym, also using three intermediates nodes.

The average time it took to consume the EchoService in the previously mentioned scenarios is shown in Figure 5. It can be seen that when only one peer consumes the EchoService, the performance is better in all cases, as expected. Consuming the service directly through a BiDiPipe took 1.6 seconds, while consuming it after 8 hops took around 6 seconds. This is a good result, since the difference (4.4s) is less than 6.4 seconds, the better time one can expect considering using 4 extra BiDiPipes to perform 8 hops. This is because instead of using 4 BiDiPipes, 8 simple unidirectional pipes were used. Also, the usage of encryption and the higher message length in JXTAnonym compared with 8 hops produces a really low overhead, just 200ms.

However, when all peers consume the EchoService at the same time, the time required to use it increases significantly when extra hops are performed. This is because when the 32 nodes send messages at the same time, each connection to the EchoService produces 8 extra messages, which is 256 extra messages for each of the 250 messages sent. Consuming the service directly through a BiDiPipe, 8 hops and JXTAnonym took 1.8, 12 and 16.9 seconds respectively. Although there is an important overhead in this scenario when using extra hops, it has to be pointed out that this is the worst case, where each peer sends messages at maximum speed. A real scenario will be between this and the previous scenario. The difference between 8 hops and JXTAnonym (4.9s) is produced due to the use of encryption and higher message length in a saturated network.

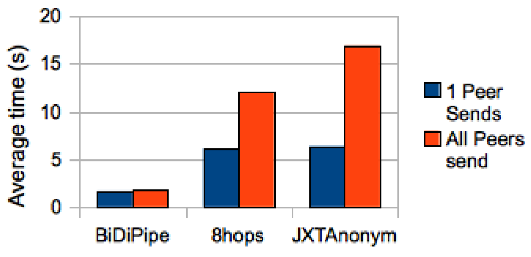

Fig. 5 Average time it takes to use EchoService

Figure 6 shows the amount of lost messages in previous scenarios. Only when BiDiPipe was used no packet loss was produced. This is because BiDiPipes are a higher level implementation of communication than unidirectional pipes, and perform extra communications to set up connections. BiDiPipe was set as no reliable but with a timeout of 1 minute. This decreases the lost packets but increases the communication time, as has been seen previously. When extra hops are performed using unidirectional pipes some packets are lost. In 8 hops scenario the percentage of lost messages ranges from 6 to 20 % while in JXTAnonym it ranges from 10 to 21 %, depending the congestion of the network. These results show that the increase of lost messages in JXTAnonym compared with 8 hops is very low. In conclusion, using unidirectional pipes produce better exchange time but increase the lost messages. Furthermore, modify the expiration time of Advertisements can change these results. A high expiration time (as default) increases the lost messages. In opposite, a lower expiration time minimizes the number of lost messages but increases the congestion of the network, since advertisements should be updated more often.

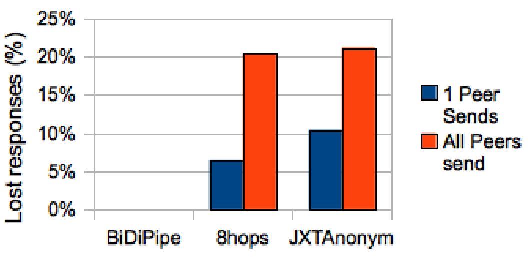

Fig. 6 Percentage of lost petitions when using the EchoService

5. Security discussion

JXTAnonym achieves anonymity by using Onion Routing, whose security has been analyzed in deep [17]. Its main vulnerable is due to traffic analysis attacks, such as packet size analysis. For this reason, as mentioned in Section 3.2.2, the ResponseCore structure contains some random data. This is used to protect the scheme against packet size analysis and make it difficult for an attacker to predict, just for its size, whether a message is the last hop, an intermediate hop, a query message or a response message. The size of each OnionCore is around 208 bytes whereas creating a new OnionLayer adds 399 bytes. From this data extra hops and extra paths are calculated, obtaining that the size of the random data has to be between 2411 and 4614 bytes.

6. Conclusions

In this paper, we have presented JXTAnonym, an anonymity layer that uses the onion routing approach to allow bidirectional message exchanges when accessing services using the JXTA protocols. JXTAnonym provides consumer and provider anonymity in any JXTA service access. The anonymity service has been implemented as a standard JXTA service, and any

JXTA Peer which belongs to a PSE Peer Group can use it. This restriction is necessary because a valid pair of public-private keys are necessary for implementing onion routing, and with this assumption is guaranteed.

Apart from the fact that JXTA currently does not provide anonymous messaging by itself, the main contributions of the chosen approach are twofold. First, the tests performed over JXTAnonym support that anonymity is feasible in JXTA. Although the time used to consume a service increases when using JXTAnonym, this amount is not so high if we take into consideration the amount of connections which are performed and the fact that encryption is employed. Second, JXTAnonym is completely invisible to end services and clients in terms of processing the received data. Their internal operation stays the same whether anonymity is used or not.

These contributions are mainly founded in the fact that JXTAnonym is built over simple JXTA elements, such as Pipe Service, Discovery Service or Advertisements. This allows using JXTAnonym without modifying the JXTA binary release, since no new JXTA protocols are defined. Moreover, Peer Group members can freely choose whether they want to belong or not to the anonymity set within the peer group. Obviously, the higher the number of peers who decide to use JXTAnonym, the higher the anonymity degree obtained.

Further research goes toward extending the anonymity layer to the publication and retrieval of Advertisements, as well as providing anonymous multicast communications. Finally, it is also worth studying how to apply mechanisms that thwart global attackers.

References

[1] B. Cohen, "Incentives build robustness in bittorrent," 1st Workshop on the Economics of Peer-2-Peer Systems, 2003.

[2] K. Matsuo, L. Barolli, F. Xhafa, A. Koyama, and A. Durresi, "Implementation of a JXTA-based P2P e-learning system and its performance evaluation," International Journal of Web Information Systems, vol.4, no.3, pp.352–371, 2008.

[3] A. Pfitzmann and M. Hansen, "Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology." http://dud.inf.tu-dresden.de/Anon_Terminology.shtml, Feb. 2008. v0.31.

[4] M. Iguchi and S. Goto, "Anonymous P2P Web Browse History Sharing for Web Page Recommendation," IEICE TRANSACTIONS on Information and Systems, vol.E90-D, no.9, pp.1343–1353, 2007.

[5] J. Gu, J. Nah, H. Kwon, J. Jang, and S. Park, "Random Visitor: Defense against Identity Attacks in P2P Networks," IEICE TRANSACTIONS on Information and Systems, vol.E91-D, no.4, pp.1058–1073, 2008.

[6] L. Gong, "JXTA: A Network Programming Environment," Internet Computing, IEEE, vol.5, no.3, pp.88–95, 2008.

[7] J. Arnedo-Moreno and J. Herrera-Joancomart, "A survey on security in JXTA applications," Journal of Systems and Software, vol.82, no.9, pp.1513–1525, 2009.

[8] J. Arnedo-Moreno and M. Domingo-Prieto, "An anonymity layer for JXTA services," 2011 Workshops of International Conference on Advanced Information Networking and Applications, pp.102–107, IEEE Press, 2011.

[9] P. Syverson, D. Goldsclag, and M. Reed, "Anonymous connections and onion routing," Proceeding of the IEEE 18th Annual Symposium on Security and Privacy, pp.44–54, 1997.

[10] X. Ren-Yi, "Survey on anonymity in unstructured peer-to-peer systems," Journal of Computer Science and Technology, vol.23, no.4, pp.660–671, July 2008.

[11] V. Scarlata, B. Levine, and C. Shields, "Responder anonymity and anonymous peer-to-peer file sharing," Proceeding of the ACM CCS, pp.17–26, 2001.

[12] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second generation onion router," Proceeding of the 13th USENIX Security Symposium, pp.303–320, 1998.

[13] Sun Microsystems Inc., "JXTA v2.0 protocols specification," 2007. http://java.net/projects/jxta-spec.

[14] CCITT, "The directory authentication framework. recommendation," 1988.

[15] B. Kaliski and J. Staddon, "PKCS1: RSA Cryptography Specifications. Version 2.0," 1998. http://www.ietf.org/rfc/rfc2437.txt.

[16] E. Halepovic and R. Deters, "The JXTA performance model and evaluation," Future Generation Computer Systems, vol.21, no.3, pp.377–390, 2005.

[17] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an analysis of onion routing security," in Designing Privacy Enhancing Technologies, Lecture Notes in Computer Science, vol.2009, pp.96–114, 2001.

Marc Domingo-Prieto holds the degree of Computer Systems and the master in Computer Architecture, Networks and Systems from Universitat Politècnica de Catalunya (UPC). He is a research assistant in the K-ryptography and Information Security for Open Networks (KISON) research group in the Open University of Catalonia (UOC). His research interests include scalable distributed algorithms and applications, energy efficient and security in mobile and peer-to-peer applications.

Joan Arnedo-Moreno is a lecturer at Estudis d'Informàtica, Multimimèdia i Telecomuncicació in the Open University of Catalonia (UOC) and works as a part-time assistant at the Universitat Politècnica de Catalunya (UPC). From the latter, he earned his degree in Computer Science in 2002 and his PhD. degree in 2009. He has published several papers in international conferences and journals and has been invited as keynote speaker at several conferences. Both his teaching and research interests are related to the fields of networking and security, more specifically in peer-to-peer systems.