Split message-based anonymity for JXTA applications

Joan Arnedo-Moreno, Noemí Pérez-Gilabert
Internet Interdisciplinary Institute (IN3)
Universitat Oberta de Catalunya
Barcelona, Spain
jarnedo, nperezg@uoc.edu

***Abstract*—JXTA is an open peer-to-peer (P2P) protocols specification that, in its about 10 years of history, has slowly evolved to appeal to a broad set of applications. As part of this process, some long awaited security improvements have been included in the latest versions. However, under some contexts, even more advanced security requirements should be met, such as anonymity. Several approaches exist to deploy anonymity in P2P networks, but no perfect solution exists. Even though path-based approaches are quite popular, it is considered that, in dynamic groups, using a split message-based one is better. In this work, we propose an anonymity service for JXTA using such approach. The proposal takes advantage JXTA's core services, in a manner so that it can be easily integrated to existing end applications and services.**[1]

***Index Terms*—peer-to-peer; security; anonymity; JXTA; Java; split message;**

I. Introduction

JXTA [1] is a well-known open protocol specification that enables the deployment of peer-to-peer (P2P) applications, allowing a set of heterogeneous devices to group and collaborate regardless of the underlying network topology. Created by SUN in 2001, JXTA has iterated through successive revisions during its 10 years of history, slowly gaining popularity, with over 2,700,000 downloads and more than 120 active projects. Such projects range from real-time collaboration [2] to remote robot control [3] or collaborative web search [4].

The latest version at present time, JXTA 2.7 [5], has been made available at the start of 2011. Its main highlights include some long awaited basic security improvements, such as secure peer groups and advertisement signatures. However, there is is still a long way to go on that regard. As P2P systems evolve and are used in new scenarios, more advanced security capabilities become important. An example of them is message anonymity [6], which becomes indispensable in systems where identities must not be disclosed, such as a ballot or peer evaluation system. Even in scenarios where anonymity is not considered indispensable, such as a simple suggestion box, it may provide the added bonus that, by safeguarding user privacy, participation is encouraged.

[1]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.

There are several ways to attain anonymity on a P2P system. Unfortunately, no silver bullet exists. Every different approach has its own strengths and weaknesses and choice entirely depends on the system's requirements. Nevertheless, path-based protocols [7] are quite popular [8], [9], since they provide the highest degree of anonymity. However, this approach puts a heavy burden on the source peer, since an initial path must be preconstructed and encrypted using asymmetric cryptography. Furthermore, preconstructed paths must be periodically probed, assessing whether all the included peers are still online before they can be used. Otherwise, the message will be lost. In dynamic networks, where it is uncertain whether a particular peer will be online or not, it is often the case that paths must be continuously reconstructed. In such scenario, it is better to use a split message-based approach, where no path must be precomputed and a high degree of sender anonymity is still maintained.

In this paper, we extend our previous work [10] and study the feasibility of split message-based approaches by proposing an anonymity service for JXTA relying on this technique. This is achieved by adapting one of the latest proposals to P2P anonymity to JXTA's architecture. Additional contributions of this paper encompass some improvements over the original protocol in order to create a service that fully realizes the capabilities that JXTA already provides, minimizing the amount of required changes to an existing system in order to integrate anonymous messaging.

The paper is structured as follows. In Section II, we provide a thorough review of current split message-based approaches to anonymity within P2P systems. Section III describes our proposed anonymity method and how it is adapted to the specifics of JXTA. We also discuss some protocol tweaks and improvements on the original proposal. Finally, Section IV concludes this paper and outlines further work.

II. Related Work

Several surveys on anonymity on unstructured P2P networks exist [11], [12], each one proposing a different taxonomy to categorize current approaches. Our work is based on the latter, being the most recent one. According to the author, approaches can be divided into three categories: unimessage, split message and replicated message. In unimessage approaches, the anonymous message travels across the network as a single entity. In

split message ones, it is split into several pieces which travel independently of each other. Finally, in replicated message ones, multiple replicas of the same message are spread across the network. Path-based approaches and its variations, the most popular ones in peer-to-peer anonymity, are an important subset in the unimessage category.

In this section, we focus on split message-based approaches, since it is the chosen mechanism for our proposed anonymity service. This category basically relies on threshold systems [13], and it is mainly used for file publication. In threshold systems, a secret is split into n parts, which are distributed through the network. It is enough that t parts ($t < n$) are gathered to be able to recover the original secret. This idea is directly applied to documents or messages, which take the role of the shared secret, distributed among the peers. The peer which is able to recover the secret is the one that finally transmits the data, acting on behalf of the actual sender, and thus hiding its identity. The main strength of split message-based approaches, in comparison with approaches from other categories, is its high reliability and ability to cope with dynamic environments. However, it comes at the cost of a quite high latency and traffic overhead. Since secret distribution usually relies on heavy use of flooding mechanisms, many messages must be transmitted across the whole network.

The *FreeHaven Project* [14] is a well known and long lived anonymous storage system, strongly focused in data persistence. Anonymity is achieved through a community of servers, named *servnets*, which process and hold the actual document parts. Each one has a pseudonym, so they can be easily identified within the network and maintain a reputation index. Users who want to publish (*publishers*) or access data (*readers*) are not necessarily part of the servnet network. In that sense, only a small part of the P2P community actually manages the securely stored data.

Publishers send document to a servnet, which uses Rabin's information dispersal algorithm (IDA) [15] to break the document into parts, following the guidelines of a threshold system. The parts are then signed using a newly generated key pair (PK_{doc}, SK_{doc}) and distributed across the servnet network under an index $H(PK_{doc})$, the hash of PK_{doc}. Readers can retrieve the document by sending a request with $H(PK_{doc})$ to a servnet. The request is then flooded through the servnet network, and each time a servnet which holds a document part receives the request, such part is sent to the reader using an anonymous remailer, such as a mix network [16]. Once the reader has received and verified enough parts, it is able to recover the original document. Both document management schemes are summarized in Figure 1.

Closely following the steps of FreeHaven, we can find the Jigsaw Distributed File System (JigDFS) [17], a secure and reliable P2P file system based on JXTA. This proposal also uses IDA to split a document among different peers, however, it is applied in an iterative manner. Once a piece has been distributed, it may be split and distributed again in order to improve reliability. Thus, a document's resulting distribution

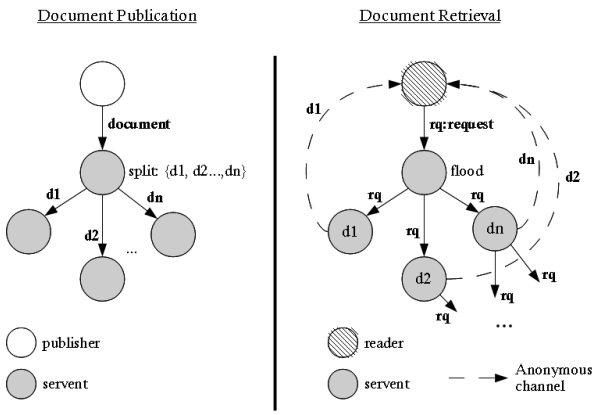

Fig. 1. FreeHaven document publication and retrieval.

scheme can be mapped into a tree-like structure. Furthermore, before each iteration is executed, files are password encrypted, so only legitimate users can access them, and noise data is added to confuse attackers. Peers which store document parts cannot actually read data even under collusion. However, this proposal's main goal is not complete anonymity, but only some degree of plausible deniability regarding document ownership once it has been stored. The underlying protocols are not truly anonymous, since senders and receivers can be clearly pinpointed.

Beyond document distribution techniques, the Secret-Sharing Mutual Anonymity Protocol [18], or *SSMP*, can be found, proposing a mutual anonymity approach also based on an information dispersal algorithm, but only focusing on document retrieval. In this approach, queries and responses are split instead of documents. When an initiator, O, wants to look up a document, it generates a secret key, T, and uses it to encrypt the document identifier, $e = E_T(id)$). Both e and T are split and probabilistically flooded across the network. All parts are indexed using a single query identifier, sq. As soon as any peer P_1 has received enough parts to be able to separately recover both e and T, the identifier is decrypted and a standard, plain text, document search query is flooded across the network. P_1 then effectively acts on behalf of the actual request initiator, using onion routing [7] to create an anonymous communication channel between itself and the document holder. When the chosen document must be transmitted, the same information distribution process is repeated, but this time splitting and flooding a new symmetric key (T') and the encrypted document itself ($e' = E_{T'}(doc)$). However, all parts are indexed using the original sq. Therefore, O is able to identify them as the response to its original query as they travel across the network, and obtain the requested document.

Reusing the previous ideas, the same authors later proposed *PUZZLE* [19], an anonymous protocol specially suited to Mobile P2P Networks (MOPNET). The basic idea is taking advantage of the broadcast feature of mobile nodes when distributing split secret parts in order to mitigate the traffic

overhead produced by flooding. Apart from that, the basic scheme for document search and retrieval is exactly the same. Nevertheless, we consider this proposal specially interesting, since the authors specifically cite JXTA as an example of MOPNET, and thus eligible for this protocol.

What could be considered the most recent iteration of the scheme used in the previous proposals can be found in *Rumor Riding(RR)* [20]. The IDA algorithm is no longer used to split queries, again focusing on document search. In fact, a publication mechanism is not contemplated.

The system's behavior is summarized in Figure 2. When a peer, the *initiator* I, wishes to issue an anonymous message, an AES symmetric key is generated and used to encrypt the query. Both the key and the ciphertext are arranged into different packets, called *rumors*, which are separately forwarded across the network following a random walk algorithm. The rumor's Time-To-live (TTL) is adaptatively chosen so there is a high probability that, at some time, they will have crossed the same peer. That peer, the *sower* S_I, is able to decrypt the ciphertext and retrieve the original query. S_I acts on behalf of the initiator, probabilistically flooding the query across the network until it reaches any peer able to process it, the *responder* R. Since queries are related to document retrieval, more than one responder may exist.

Once the query has been processed, the response is sent back from R to I using a similar strategy. However, this time the response is encrypted using I's public key prior to rumor generation. Then, a new pair of rumors is forwarded until they have crossed a new sower, S_R. However, the response is not recovered at S_R, but the response rumors are directly submitted to S_I. From S_I, both rumors independently backtrack the path the query rumors took at the start of the protocol, until they converge at I, which is the only peer able to recover the plaintext response.

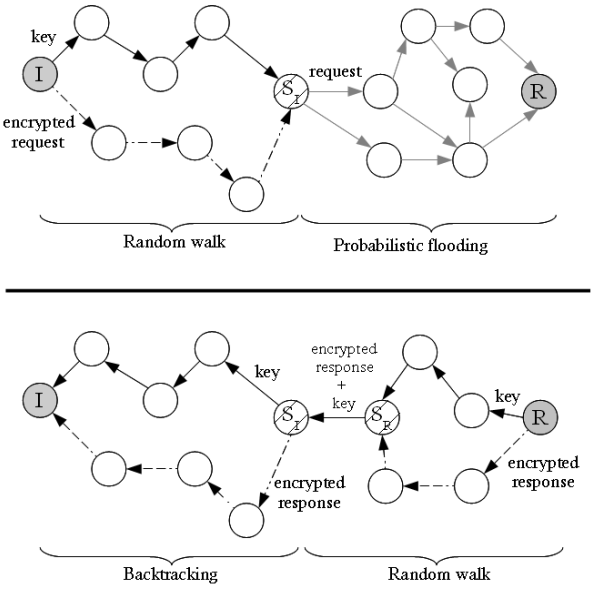

Fig. 2. Rumor Riding Query issuance and response.

Finally, it is interesting to make note of the fact that, in several of the existing proposals, even though split message approaches are used, at some point they rely on unimessage mechanisms, such as mix networks or onion routing [7], to establish a secure channel. Even though those mechanisms don't play a main role in the protocol, they are necessary. Therefore, these kind of approaches could actually be considered, up to some point, hybrid ones.

III. Split-message Based Anonymity Service

JXTA main purpose is providing mechanisms to share resources and services. In order to provide anonymity in services consumption, is important to review the JXTA messaging architecture. From this analysis, it is possible to identify the elements that can be taken advantage of in order to define a split message-based anonymity layer which is transparent and finely integrated to JXTA architecture, without the need of defining additional protocols or core primitives.

A. JXTA architecture overview

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 *Membership Service*, *Discovery Service* and *Pipe Service*.

The Membership Service allows joining a peer group and claiming an unique identity within the group's context. Through this service, each group member is provided with a credential, which may be used at any time to authenticate to other group members. Different implementations exist depending on the chosen way such identity is claimed and the credential format.

The Discovery Service manages group resources publication and discovery. Every resource in a JXTA group is described by an *Advertisement*, a metadata document. A resource cannot be accessed unless its corresponding Advertisement is previously retrieved. It is the Discovery Service's responsibility to manage and distribute Advertisements, since a resource is not considered available unless it is periodically published.

There are several Adviertisement types, but the most important ones are:

- *Peer Advertisement*: Describes a peer and the resources and services it provides, under a special service parameter entry. 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*, and abstract communication channel and the main mechanism to exchange data between applications.

Finally, 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, the *JxtaBiDiPipe*, which provides a bidirectional communication channel. The latter is usually preferred by services, since it allows a straightforward query-response message 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 makes available and listens to an *input pipe* in order to process inbound communication requests. This input pipe is defined using a Pipe Advertisement.
2) The service provider publishes the Pipe Advertisement to other group members 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 also uses the Discovery Service to retrieve the Pipe Advertisement. Then, a connection is established via the Pipe Service and the consumer may begin sending messages.
5) Once a message is received at the server side, the results depend on the pipe type, JxtaUnicast or JxtaBiDiPipe. In the former case, messages may be processed, but no response is possible. On the latter case, a linked outbound communication channel is created and two-way exchanges are made possible.

JXTA messages sent through pipe connections follow a predefined structure comprised of a set of name/value pairs labelled under a namespace and organized as an ordered sequence. As a message passes down each JXTA layer, one or more named elements may be added to the message (for example, control data). Their order within the message structure always follows the same order they were added. As a message is processed back up the stack, each layer will remove these elements, until only application data remains.

Message exchanges can be secured in JXTA by using a group based on the *PSE (Personal Security Environment)* Membership Service implementation. Under this kind of peer group, each peer is provided with a credential based on public key cryptography. 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.

B. Anonymizing procedure

From all the protocols reviewed in Section II, we have chosen RR as the foundation for our work. The main reasons being that it is one of the most recent proposals, as well as being a "pure" protocol, instead of relying in unimessage approaches at some point, which would make the point moot. Even though RR is focused on document retrieval via flooding mechanisms, it can be easily adapted to point-to-point service access. A study of the performance of this general approach, based on logs from a Gnutella network, is addressed in the original paper [20].

We propose an anonymity layer that causes the minimum interferences on the JXTA messaging architecture, according to the review done in Section III-A. The anonymizing service works within the context of a peer group, meaning that only peers from the same peer group may exchange anonymous messages. Each peers is free to deploy the service, or not, and it is not assumed that every group member always does all the time. The service is tailored to JXTA's core services features. 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. Applications which execute end services or/and clients may communicate through the anonymity service, which acts as an invisible layer. However, as a requirement, the peer group must operate under the PSE Membership Service, guaranteeing that all group members have a properly initialized keystore.

An overview of the proposed architecture is summarized in Figure 3.

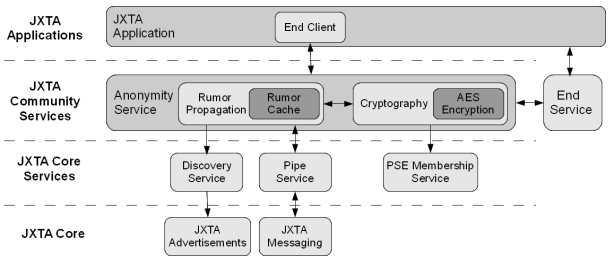

Fig. 3. Anonymity service operation in the context of JXTA's architectural design.

Our proposed RR-based Anonymity Service relies on five distinct phases: *Service Publication*, *Query Setup*, *Random Walk*, *Response Processing* and *Backtracking*. All peers which execute the Anonymity Service track rumors as they travel across the network using a local cache, `RumorDB`, a table with the following fields:

- `RumorID`: Rumor identifier.
- `CachedRumor`: A cached rumor of any type.
- `Inbound`: Peer ID from which the rumor was received.
- `Outbound`: Peer ID the rumor was forwarded to.
- `TimeStamp`: Time the rumor was routed.

Given the `TimeStamp` field, all entries expire after some time and are automatically flushed. This avoids cluttering from stagnant entries related to messages which have been dropped for some reason during transit.

1) ***Service publication****:* Just like any JXTA service, the Anonymity Service's Pipe Advertisement must be distributed among other peer members before it may receive incoming requests. Each peer is responsible for the publication of its own service instance's pipe and this procedure must be periodically executed.

In order to maintain the number of advertisements transmitted within the network at a minimum, the service's Pipe Advertisement is piggybacked within each peer's Peer Adver-

tisement, indexed by a hardcoded well-known service identifier. In fact, this is the same method the PSE Membership Service employs to readily distribute public key information. The main advantage of this method is that it is only necessary to manage a single advertisement type to publish or discover all data related to the Anonymity Service (service pipe and peer cryptographic data). Furthermore, the Peer Advertisement's publication and discovery is already part of JXTA's standard procedures, being every peer's presence mechanism. Therefore, this advertisement of any group member which is considered online always is always readily available.

A sample Peer Advertisement is shown in Figure 4 (some encoded data has been shortened for the sake of readability). The topmost grey section (a) denotes the Anonymity Service parameters for that peer, which only consist of its Pipe Advertisement. The lower grey section (b) denotes the PSE Membership Service parameters, the peer's public key encapsulated in a digital certificate.

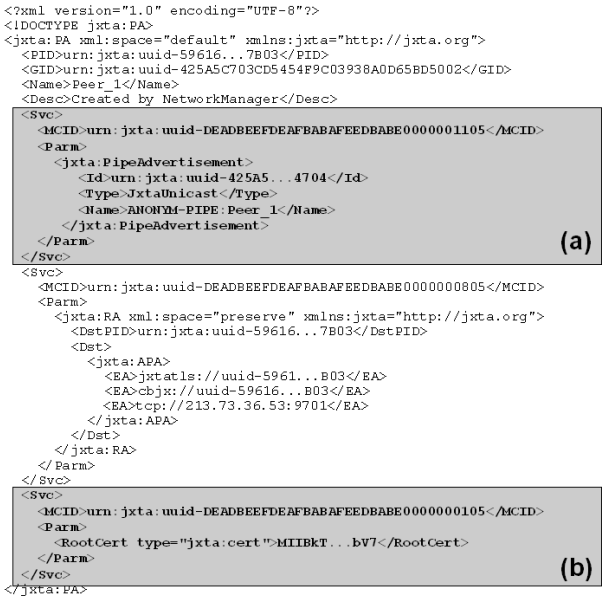

Fig. 4. Peer Advertisement supporting the Anonymity Service, containing (a) Anonymity Service Pipe Advertisement and (b) PSE public key data.

*2) **Query Setup**:* This step starts the whole process and is executed at the end client application (the *initiator*, I) whenever it wishes to send a query to the end service (the responder, R). A rumor pair is initialized, beginning its trip through the group members, towards R.

The process is executed as follows:

1) A local application executing on peer I wishes to send a query message Qry to an end service executing on peer R.
2) Via the Discovery Service, R's Peer Advertisement is retrieved. The following information is looked up in the parameters section (see Figure 4):
   - The end service's Pipe Advertisement, $EndSvcPipeAdv$. Its ID, $EndSvcID$, is extracted.
   - R's public key, PK_r.
   - The Anonymity Service's Pipe Advertisement, $EndAnonymPipeAdv$.
3) If $EndSvcPipeAdv$ is of the *JxtaBiDiPipe* type, a response will be expected. A secret and public key pair, SK_i and PK_i, is generated.
4) Two time-to-live values, $TTLforward$ and $TTLbackwards$, are established. How they are chosen will be discussed later.
5) An $AnonymData$ structure is created as follows:
   - `Message` $= Qry$
   - `EndService` $= EndSvcID$
   - `ResponseKey` $= PK_i$, if a response is expected. Otherwise, this field is omitted.
   - `ResponseTTL` $= TTLbackwards$, if a response is expected. Otherwise, this field is omitted.
6) $AnonymData$ is encrypted using PK_r under a wrapped key scheme, such as [21], generating the bit string $EncAnonymData$.
7) A $RumorData$ structure is generated as follows:
   - `AnonymMsg` $= EncAnonymData$
   - `DestinationPipe` $= EndAnonymPipeAdv$
8) An AES [22] symmetric key, K is dynamically generated.
9) $RumorData$ is encrypted using K, resulting in the $EncRumorData$ bit string.
10) A random JXTA identifier, $RUID$ is generated.
11) A message rumor, $MessageRumor$, is created using a JXTA message with the following name-value pairs:
    - `RumorData` $= EncRumorData$
    - `RumorID` $= RUID$
    - `TTL` $= TTLforward$
12) A key rumor, $KeyRumor$, is created using a JXTA message with the following name-value pairs:
    - `RumorKey` $= K$
    - `RumorID` $= RUID$
    - `TTL` $= TTLforward$
13) The Peer Advertisements of two different group members which are executing the Anonymity Service are located. This is achieved by using the Discovery Service to look up peers which include such service in their Peer Advertisement, as previously explained in the Service Publication step.
14) The contained Pipe Advertisements are extracted from each advertisement. $MessageRumor$ is forwarded to one pipe and $KeyRumor$ to the other.

Once the rumors have been forwarded, if a response is expected, two entries, one for each rumor, are created in `RumorDB`. The `RumorID` field is assigned $RUID$ and the `Outbound` field is assigned the JXTA ID of the group member used to forward the rumor. In this case, `CachedRumor` and `Inbound` are left blank, serving as a special mark indicating that the entries correspond to rumors created at this peer, which is expecting a response. SK_i and

a callback mechanism to the end client application are also separately stored.

*3) **Random Walk**:* Whenever the Anonymity Service is executing in a peer, it is always listening to its input pipe, waiting for inbound rumors of any type: message and key. Its basic operation mode is processing inbound rumors received from any peer, P_{in}, and then immediately forwarding them to a different outbound peer, P_{out}, acting as a router, immediately updating `RumorDB`.

The rumor may be processed in two different manners when received, random walk or backtracking, depending on whether an entry already exists in `RumorDB` for that rumor ID and type, or not. For now, the former case is explained, since it is the way rumors are always processed immediately after query setup. In this scenario, both rumors are iteratively forwarded through group members until one of them is considered eligible to become the *sower*, S_i, which will forward the actual query on behalf of I.

1) A rumor is received from peer P_{in}.
2) The `RumorID` field value, $RUID$, is extracted from the rumor.
3) The local cache is looked up for a rumor of the same type with a `RumorID` field matching $RUID$.
4) Random walk must be applied if one of these conditions apply:
   - No enty currently exists for this rumor in the local cache.
   - An entry exists, but the value in the `Outbound` field is different from P_{in}'s ID.
5) The complementary rumor type, with a `RumorID` field also matching $RUID$, is looked up in the local cache.
6) If a match exists, the current peer has now a valid message and key rumor pair, $MessageRumor$ and $KeyRumor$, and becomes eligible to become a sower.
   a) The `RumorKey` field value, K, is extracted from $KeyRumor$.
   b) The `RumorData` field value in $MessageRumor$ is decrypted using K. The result is the $RumorData$ structure.
   c) The `DestinationPipe` field value, $AnonymSvcPipeAdv$, a pipe advertisement, is extracted from $RumorData$.
   d) The `AnonymMsg` field value, $EncAnonymData$, is extracted from $RumorData$.
   e) A JXTA message, $EndMessage$ is created with the following field name-value pairs:
      - `AnonymMsg` $= EncAnonymData$
      - `RumorID` $= RUID$
   f) $EndMessage$ is forwarded using $EndAnonymPipeAdv$, which points to R's instance of the Anonymity Service. Therefore, $P_{out} = R$
7) If no match exists:
   a) The `TTL` field value is decreased by 1.
   b) If $TTL = 0$, the rumor is discarded. Otherwise it will be forwarded to another peer, P_{out}.
   c) If no entries existed at step 4, a random value different from P_{in} is chosen for P_{out}.
   d) If other entries already existed, P_{out} is also chosen at random, but it must different to every entry's `Inbound` and `Outbound` field values.
8) The rumor is forwarded to P_{out}.

Once a rumor is forwarded, the local cache is updated. The `RumorID` field is assigned $RUID$ and the rumor itself is stored in the `CachedRumor`. The JXTA ID's of P_{in} and P_{out} are respectively assigned to the `Inbound` and `Outbound` fields. It is worth remarking that, during this step, given its random nature, it is possible that sometimes the same rumor passes through the same peer more than once (that would correspond to the second condition in step 4). Rumor processing does not change because of that, an additional entry is just added to the cache. Successive entries are differentiated by their timestamp, and, most of the time, the values in `Inbound` and `Outbound`. This situation does not affect the protocol, as will be explained later, in the backtracking step.

*4) **Response Processing**:* Once the Anonymity Service's instance executing in R receives $EndMessage$ from the sower, this step is executed. Given the nature of the random walk process, the possibility exists that multiple peers are eligible to become sowers, and therefore, a few copies of the same request are received. By temporally storing $RUID$ for served requests, it is possible to guarantee that, even though several repeated queries are received, only one is actually processed and responded. The rest are discarded.

1) Using the PSE Membership Service, R's secret key, SK_r is looked up.
2) The `AnonymMsg` field value is decrypted using SK_r, exposing the $AnonymData = \{Qry, EndSvcID, PK_i, TTLbackwards\}$ structure.
3) Using the Discovery Service, R's Peer Advertisement is retrieved. A Pipe Advertisement, $EndSvcPipeAdv$, with an ID equal to $EndSvcID$, is extracted from the service parameters section. This is the end service's advertisement.
4) A connection is opened using $EndSvcPipeAdv$ and Qry is sent through it.
5) If no response is expected ($EndSvcPipeAdv$ is of the *JxtaUnicast* type), the process is over.
6) If a response is expected ($EndSvcPipeAdv$ is of the *JxtaBidiPipe* type), the Anonymity Service waits for the response, Rsp.
7) At this point, a rumor pair is created, following the same steps 4-13 in Query Setup, with the following changes:
   - The $AnonymData$ structure only has a single field, `Message` $= Rsp$. It is encrypted using PK_i.
   - The `TTL` field, in the both the message and key rumor, is initialized to $TTLbackwards$.

- The `RumorID` field in the rumors is initialized to $RUID$.

8) Once the rumors have been created, instead of following two random separate paths, they are both sent back to the sower using the Anonymity Service's pipe.

When the sower receives both new rumors, the backtrack process begins.

*5) **Backtracking**:* In the backtrack step, rumors are routed back to the initiator following the inverse path they used to reach the sower, until they arrive to I. Whenever a peer executing the Anonymity Service receives a rumor, it is able to discern whether a rumor is being backtracked, instead of executing a random walk, in the following manner. When a rumor is received from P_{in}, it is looked up in the local cache. If an entry already exists and the value in the `Outbound` field is equal to P_{in}'s ID, backtracking is detected, since this is a circumstance that random walk step ensures will never happen (see step 7). In this scenario, it is worth remarking that it is always guaranteed that an entry exists in a peer's local cache for a given rumor being processed

1) A rumor of any kind is received from P_{in} and backtracking is detected.
2) `RumorDB` is checked for entries with `RumorID` $= RUID$.
3) The entries `Inbound` field is checked. If it any of them is blank, then this peer was the initiator.
   a) The rumor is stored in the `CachedRumor` field.
   b) A rumor of the alternate type but the same ID is looked up. If no match exists, the peer cannot proceed and must wait for it.
   c) If a match exists, the contained response, Rsp, is retrieved in a manner similar to the one used by the sower to unbundle the initial rumors, in step 6 of the random walk process. SK_i is used to decrypt $AnonymData$.
   d) The response is pushed to the client application using the stored callback mechanism.
4) Otherwise, the entry with the most recent timestamp is used in the following manner:
   a) The `TTL` field value in the rumor is decreased by 1. Nevertheless, given then chosen values at the response processing step, it is guaranteed that the rumors will reach the initiator before the `TTL` field value reaches 0.
   b) The rumor is routed back using the `Inbound` entry's value. If the peer has gone momentarily offline, the message is held for some time, waiting until it comes back online.
   c) Finally, this rumor's entry is deleted from the local cache, cleaning up.

C. Discussion on protocol tweaks and improvements

In our adaptation of the Rumor Riding protocol to JXTA, we have also included some minor tweaks and improvements over the original proposal. Following, we discuss such them.

*1) **Rumor matching using identifiers**:* In the original proposal, a CRC code is appended to the original query prior to encryption (Query Setup, step 9). During the Random Walk process, step 3, rumor matching is achieved by decrypting each message and checking the CRC code, one by one, instead. A concordance is considered to be found when a CRC check passes, meaning that the message has been correctly decrypted. Otherwise, it is concluded that no message rumor is currently cached for that key. This procedure is not motivated by security, since the authors state that "...the CRC function is to avoid using a complex text understanding technique to distinguish a meaningful M...", M being the original query. However, in JXTA, it is very easy to check whether a query has been properly decrypted, since they have a clear structure. Even if it was not the case, this is a very costly procedure just to check rumor matching. It is much more efficient to know whether a message and key rumor match before even attempting decryption, for example, using a simple common identifier ($RUID$).

*2) **Final query privacy**:* Originally, once the initial random walk rumors conflate into a potential sower, the message is decrypted and the original query is probabilistically flooded across the whole network in plain text. Even though anonymity is maintained, a considerable amount of peers will receive the query and be able to easily eavesdrop on the transmission. For that reason, we propose that the query is initially encrypted using the service's public key in Query Setup, step 6. Thus, the sower transmits an encrypted query in Random Walk, step 6, to the final destination, and privacy is maintained along the full source-destination path. It must be noted that, using our proposed procedure (by not flooding the final message), some anonymity is lost, since the sower, but only the sower, must know which is the final destination in order to directly transmit the query. However, other anonymity approaches, such as probabilistic path-based ones [9], also share this characteristic, and it is considered an acceptable loss. In addition, even though the sower knows the final destination peer, in our scheme, it is unable to guess which specific service is being accessed.

*3) **TTL value range choice**:* Given the nature of the random walk protocol, it is very important to choose an appropriate TTL value range which ensures a high probability that both rumors will traverse the same peer at some point, before their TTL values reach 0. In order to be able to choose an acceptable value, we have calculated through simulation an approximate value for the probability of such event happening given different peer group sizes and TTL values (see Figure 5). 100 million repetitions have been tested for each combination of group size and TTL value. In the Figure, we just show some significative results, up to values where the probability almost reaches 1.0 for all cases. We have considered group sizes up to 32 peers, since it is is the typical maximum [23].

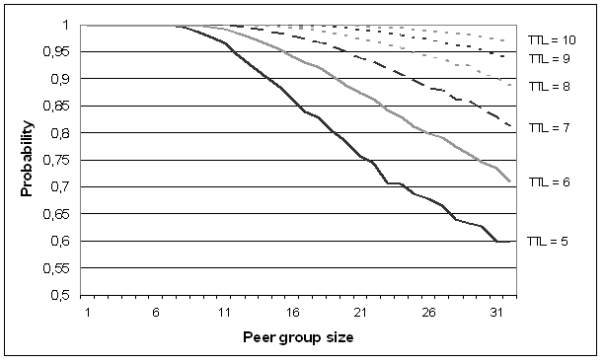

Fig. 5. Probability that both rumors traverse the same peer.

An initial TTL value may be randomly chosen from any range which ensure a high probability, given the current group size. As the figure shows, for TTL values greater than 10 the probability will quickly approach 1.0 for any group size. $TTLforward$ and $TTLbackwards$ (see Query Setup, step 4) are calculated from this initial value in a manner that it is not possible for an attacker to guess the initial TTL vale one the rumors ar crossing the network, but guarantee that both the query and the response will reach the intended destination. A random value between 0-6 is chosen and added to the initial TTL value to establish $TTLbackwards$. $TTLforward$ is calculated by subtracting a random value between 0-3 from $TTLbackwards$.

IV. Conclusions

We have proposed a split message-based anonymity layer for JXTA, as an alternative to the more popular unimessage-based ones. The main goal is providing JXTA with a set of anonymous messaging approaches from which developers may choose, each suitable to different scenarios. At present time, we are finishing the implementation and in short will be able to test its behavior in real JXTA networks and compare our results with the ones presented in the original proposal.

Apart from providing JXTA with anonymous messaging, the main contributions of the chosen approach are twofold. First of all, it is tightly integrated to JXTA's architecture, working only within the context of a standard service's operation method. Thus, it has not been necessary to define new protocols or primitives aside from the ones already available in JXTA. A further advantage of this is that pipe and cryptographic data publication is seamlessly intergraded within JXTA's standard presence mechanism. Thus, the anonymity layer is almost invisible to the actual service being accessed. From the end service provider's standpoint, the original message has been normally received through its published input pipe. In addition, we propose some tweaks and improvements over the original protocol, which we hope increase its overall performance.

Once the implementation is finished, further research goes to extensive testing of our proposed anonymous service in real JXTA networks, so it is possible to assess its feasibility in different scenarios and compare the results to other anonymity approaches (such as unimesage or replicated message ones). This testing will allows us to fine tune some of the protocol parameters, such as some timeouts. Finally, it is also worth studying how to apply mechanisms that are able to thwart global attackers.

References

[1] Sun Microsystems Inc., "JXTA v2.0 protocols specification", 2007, https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html.

[2] ConneX, "Connex Project Homepage", 2007, http://connex.sourceforge.net.

[3] Matsuo K., Barolli L., Arnedo-Moreno J., Xhafa F., Koyama A., and Durresi A., "Experimental results and evaluation of smartbox stimulation device in a P2P e-learning system", 2009, pp. 37–44.

[4] Namrata Lele, Le-Shin Wu, Ruj Akavipat, and Filippo Menczer, "Six-earch.org 2.0 peer application for collaborative web search", in *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, 2009, pp. 333–334.

[5] Project Kenai, "JXSE: The Java Implementation of the JXTA Protocols", 2011, http://jxse.kenai.com.

[6] Hansen M. Pfitzmann A., "Anonymity, unlinkability, undetectability, unobserv- ability, pseudonymity, and identity management a consolidated proposal for terminology", 2008, http://dud.inf.tu-dresden.de/Anon_Terminology.shtm.

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

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

[9] Rubin A.D. Meiter M.K., "Crowds: Anonymity for web transactions", *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–93, 2004.

[10] M. Doming-Prieto and J. Arnedo-Moreno, "JXTAnonym: An anonymity layer for JXTA services messaging", *IEICE Transactions on Information and Systems*, vol. E95-D, no. 1, January 2012.

[11] Bhatti S. Rogers M, "How to disappera completely: A survey of private peer-to-peer networks", in *In Proceedings of International Workshop on Sustaining Privacy in Autonomous Collaborative Environments (SPACE)*, 2007.

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

[13] Yvo G. Desmedt and Yair Frankel, "Threshold cryptosystems", in *CRYPTO '89: Proceedings on Advances in cryptology*, New York, NY, USA, 1989, pp. 307–315, Springer-Verlag New York, Inc.

[14] Freedman M.J. Dingledine R. and D. Molnar, "The free haven project: Distributed anonymous storage service.", *Lecture Notes in Computer Science*, p. 67, 2001.

[15] Michael O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance", *J. ACM*, vol. 36, pp. 335–348, April 1989.

[16] U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman, "Mix-master protocol version 2. technical report", 2004, Internet-Draft.

[17] J. Bian and R. Seker, "Jigdfs: A secure distributed file system", in *IEEE Symposium on Computational Intelligence in Cyber Security, 2009. CICS '09*, 2009, pp. 76 – 82.

[18] Han J., Liu Y., Xiao L., Xiao R., and L.M. Ni, "A mutual anonymous peer-to-peer protocol design", in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, 2005, p. 68a.

[19] Han J. and Liu Y., "Mutual anonymity for mobile p2p systems", *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, no. 8, pp. 1009 –1019, 2008.

[20] Liu Y., Han J., and Wang J., "Rumor riding: Anonymizing unstructured peer-to-peer systems", *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 3, pp. 464–475, 2011.

[21] J. Staddon B. Kaliski, "PKCS1: RSA Cryptography Specifications. Version 2.0", 1998.

[22] FIPS Federal Information Processing Standard, "Advanced encryption standard (aes)", 2001.

[23] Deters R. Halepovic E., "The JXTA performance model and evaluation", 2005, vol. 21, pp. 8377–390.