An anonymity layer for JXTA services

Joan Arnedo-Moreno, Marc Domingo-Prieto
Estudis d'Informàtica, Multimèdia i Telecomunicació
Universitat Oberta de Catalunya
Barcelona, Spain
jarnedo, mdomingopr@uoc.edu

***Abstract*—JXTA is a set of open protocols that enable the creation and deployment of peer-to-peer (P2P) networks, allowing the execution of services in a distributed manner. Being a generic P2P middleware, it has slowly evolved in order to appeal a broad set of different applications. Part of this evolution includes providing basic security capabilities in its protocols in order to achieve some degree of message privacy and authentication. However, under some contexts, more advanced security requirements should be met, such as anonymity. In this work, we propose how to adapt JXTA messaging so that services may be anonymously accessed, by taking advantage of JXTA's idiosyncracies and capabilities, in a manner that is completely invisible to the existing protocols.** [1]

Keywords: peer-to-peer, security, anonymity, JXTA, Java, onion routing.

I. Introduction

The adoption of peer-to-peer (P2P) architectures allows the deployment of resources and services in a self-organizing and scalable manner. However, just as the popularity of P2P applications has risen, concerns regarding the degree of security they can provide have also increased, specially since it is no longer possible to trust a central server which capitalizes all security operations. Fortunately, P2P middlewares and applications are becoming more sensitive to such issue, conscious that ignoring this it may jeopardize their success.

A security baseline in a P2P system usually includes at least some degree of privacy, ensuring that the contents of a message exchange are not revealed to an eavesdropper, and authentication, guaranteeing the which is the identity of each endpoint during any message exchange. Both security services may be deployed in a more or less straightforward manner using protocols which rely on data encryption and Message Authentication Codes (MAC) or digital signatures. An example of such protocol is SSL [1], widely used in Internet applications to provide secure communications.

As P2P systems evolve and are used in new scenarios, more advanced security capabilities become important. An example of them is message anonymity [2], for which privacy is a necessary but not sufficient requisite. Usually, the concept of anonymity in P2P networks is associated with situations where exposure has very strong implications, such as maintaining the right to free speech or circumventing legal responsibilities [3]. However, there are everyday situations which require anonymity, but even in the worse case scenario, nobody may end up in jail. Some straightforward examples are a corporate suggestion box, a small ballot system or a peer evaluation form. In such scenarios, actual anonymity provides the added bonus that, by also increasing the users' trust in the system, participation is encouraged.

It's worth remarking that, in each of these scenarios, in order for messages to remain truly anonymous, it must not possible for anybody within the P2P network to discover the message source, no matter what tool is used or its role in the system. It is not enough that the system just hides source message information to the users at the upper protocol layers or such data is managed only by a trusted entity, since it may still be possible to extract the data using external methods or simply due to misuse. Low level protocols must actually make it inaccessible. Even the mere case that users know that some administrator is still able to expose them may be enough to stymie participation.

JXTA [4] is an example of a P2P system which already considers some basic security capabilities, but not anonymity. The term is briefly mentioned in its specification, but deploys no actual serious mechanism. In this paper, an anonymity layer is presented, adapting the current anonymity approaches to the particular idiosyncracies of JXTA, so any application may send requests to services without disclosing its identity. The main contribution of this work is defining the anonymity layer in such a manner that it may is invisible to accessed services. Furthermore, the proposal fully realizes the capabilities that JXTA already possesses, minimizing the amount of required changes on an existing system in order to integrate anonymous messaging.

The paper is structured as follows. In Section II we provide a brief summary of the current approaches to anonymity in the context of P2P applications. Section III describes the chosen anonymity method and describes how it is adapted to the idiosyncracies of JXTA services. A security and performance evaluation of the anonymity service is provided in Section IV. Section V concludes this paper and outlines further work.

II. Current Approaches to Anonymity in P2P

A thorough review of how anonymity may be attained in the context of P2P networks is found in [5]. In this survey, the author categorizes that anonymizing techniques according to three different approaches: unimessage, split message and

[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.

replicated message. In this section, a very brief overview of these approaches is presented, just providing the main ideas about how each one works and summarizing their main pros and cons, so it is possible to assess which is the best approach for JXTA. It must also be taken into account that even though the provided application examples are based on each approach, each one has its own additional subtleties.

A. *Unimessage-based approaches*

In this approach, an anonymous path is pre-constructed by the sender before a message is dispatched towards the actual final destination. The message will be relayed through every peer in the path before it reaches its final destination. However, the message is encrypted in such a manner that, at every hop, each relay peer is only able to learn which is the next hop where the message must be sent. No peer, apart from the initial message sender, is able read the whole data path, not even the final destination. Therefore, every time a peer receives the message, it cannot be decided whether it has been received from the original sender or just a former relay. For the same reason, it is not possible to know whether the next hop is the actual final destination.

Path encryption is usually performed by adding successive encryption layers to the original message using public key cryptography. The public key of each relay is applied at each layer, so when a relay decrypts the received message with its private key, the only information found is the identity of the next hop and the encrypted data that must be forwarded. The final destination will be the peer which finds the message (instead of a next hop) after the applying the decryption process. For that reason, this anonymity approach is also called Onion Routing [6].

This is the most popular approach to anonymity in P2P networks, used by systems such as Tor [7] or APFS [8]. This basic idea is also used in other systems, such as Crowds [9] or Shortcut [10], albeit with some modifications. The path is not predetermined by the sender, but probabilistically decided at each hop. In such approaches, however, only sender anonymity is achieved. Any relay peer knows the final destination, since it is transmitted in clear text.

Pros: Medium efficiency, Medium-Low overhead.
Cons: Low reliability.

B. *Split message-based approaches*

This approach is based on a threshold system [11] and it is usually used in file publication systems which focus on sender anonymity. In threshold systems, a secret is split into n parts, which are distributed between the the users. It is enough that t users ($t < n$) collaborate to recover the original secret. This idea is directly applied to files or messages, which take the role of the shared secret, distributed among the peers. The FreeHaven Project [12] is a well known anonymous system which relies on this approach in an straightforward manner.

Still being based on this approach, other systems exist which apply a small twist, such as splitting file request queries, instead of the published file, or encrypting the message and distributing both the key and the cipher as shares [13]. Shares are then continually propagated across the network so a single random peer will be the only one able to recover the query and issue it on behalf of the original creator.

Pros: High reliability.
Cons: Low efficiency, High overhead.

C. *Replicated message-based approaches*

This approach relies on using message broadcast or multicast to send the content to everyone, thus hiding which peer within the network is the actual destination. Messages are encrypted so only the destination will be able to read the content, maintaining its privacy. Hordes [14] is an example in this category.

Pros: High reliability.
Cons: Low efficiency, Very High overhead.

III. Anonymizing JXTA messaging

In order to propose an anonymizing mechanism for JXTA messaging which takes advantage of the middleware's basic capabilities, it is important to review the most important characteristics of its architecture. From this study, it is possible to create an anonymity layer which nicely integrates with JXTA without the need to define additional protocols or core primitives.

A. *JXTA Messaging architecture*

JXTA differs from other P2P middlewares because it introduces the concept of *peer group*, one of the main foundations of its architecture. Usually, P2P environments are conceptualized as a global overlay network without any kind of logical segmentation or segregation as far as resource availability is concerned. However, in JXTA, the global overlay network is segmented into overlapping, hierarchical groups of peers, which offer a context for accessing services. The concept of peer groups is very important, since messages can only be exchanged between peers belonging to the same peer group. Peers may interact through a set of core services that groups offer to its members, the most relevant ones being the *Discovery Service*, the *Membership Service* and the *Pipe Service*.

The **Discovery Service** provides a mechanism to conveniently publish and distribute resources available within the peer group. In JXTA, all resources are announced using a special message named *advertisement*, a metadata record describing it and how it can be accessed, which is sent to all other peer group members. Peers cannot access a resource without previously retrieving its associated advertisement. The most important types of advertisements in JXTA are the following ones:

- *Peer Advertisement*: Describes a peer and the resources and services it provides to a peer group, as well as any available service's special parameters. It also acts as the peer's presence mechanism.

- *Peer Group Advertisement*: Describes a peer group, its specific resources and its offered services' parameters.
- *Pipe Advertisement*: Describes a pipe, the JXTA core mechanism for exchanging messages between two applications or services, providing a simple, unidirectional and asynchronous communication channel.

Whenever a peer receives the advertisement, it is indexed, stored in a local cache and assigned an expiration date. When the expiration date is reached, the advertisement is considered stagnant and flushed from the cache. Whenever an existing advertisement is received again, its expiration date is renewed. Advertisements must be periodically retransmitted in order to attain permanency or update parameter changes.

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. In the latest version of JXTA at his time (2.5), three different implementations exist, but only one of them actually provides a method to securely provide such unique identity: the PSE (Personal Security Environment) Membership Service.

The PSE's credentials are based on PKIX [15] certificates. An identity is claimed by being able to properly initialize the keystore which holds the private key for that certificate. Since PSE is based on public key cryptography, its credentials are chosen as a means to provide asymmetric key management for messaging security services. In fact, every security service currently provided by JXTA assumes that PSE is used as the group's Membership Service. Otherwise, it is not possible to use them. In order to distribute a peer's PSE credential to other group members, it is included in a special service parameter entry in its Peer Advertisement.

The **Pipe Service** provides a mechanism to manage JXTA pipes, which are the basic way to access services. Services are made available by individual peers by accepting incoming messages via an *input pipe*. Every input pipe has an associated Pipe Advertisement, which is distributed among group members by the service provider. Whenever another peer wants to send a request to the service, such advertisement must be previously looked up (via the Discovery Service) and retrieved. Only then, an outbound connection, an *output pipe*, can be established via the Pipe Service. A service is considered shut down whenever its Pipe Advertisement becomes unavailable for a set period of time.

JXTA messages are sent through pipe connections. Such messages follow a predefined structure comprised of a set of name/value pairs, organized as an ordered sequence, the most recently added element appearing at the end of the message. As a message passes down each JXTA layer, one or more named elements may be added to the message (for example, control data). As a message is processed back up the stack, each layer will remove these elements.

B. Anonymizing procedure

Form the analysis in sections II and III-A, the chosen method to be adapted in order to anonymize JXTA messaging is onion routing, the main reasons being twofold. First of all, it is the one which keeps a better efficiency while maintaining a high anonymity degree. Keeping a good efficiency is specially relevant since message anonymity usually has a very high impact in system performance, even when compared to other security services. And secondly, it is based in an architecture where nodes are completely autonomous and communications are basically unidirectional, which meshes with the principles of JXTA messaging (pipes). Additionally, it is also worth pointing out that its main disadvantage, low reliability, is not specially detrimental within the JXTA architecture, since JXTA pipes are already non-reliable by design.

The proposal to provide a JXTA network with anonymizing capabilities is based in the deployment of an Anonymity Service, which takes advantage of all of JXTA's core capabilities related to service publication and access. Just like any other standard JXTA service, the proposed Anonymity Service only works within the context of a peer group, meaning that only peers from the same peer group may exchange anonymous messages. As an additional requirement, such group must operate under the PSE Membership Service, juts like all of JXTA's current security capabilities. Thus, it can be guaranteed that all group members have a properly initialized JXTA cryptographic keystore.

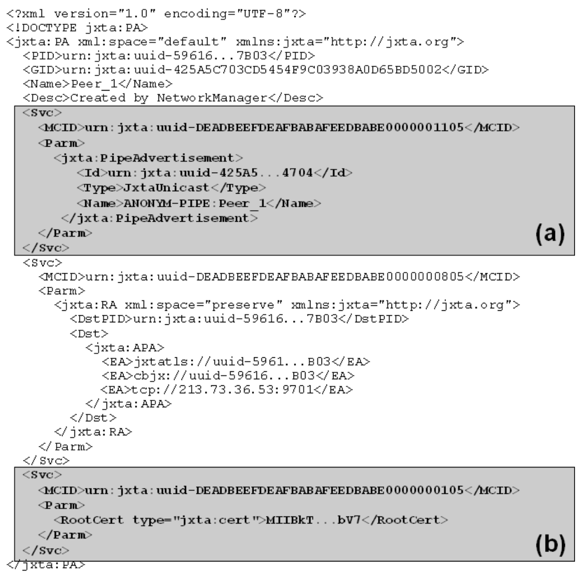
Anonymity Service execution in any peer relies on three distinct procedures: *Publication*, *Message Setup* and *Message Processing*.

Anonymity 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 Advertisement, 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, such advertisement from any group member which is considered online is always readily available.

A sample Peer Advertisement is shown in Figure 1 (some encoded data has been shortened for the sake of readability). The first grey section (a) denotes the Anonymity Service parameters for that peer, which only consist of its Pipe Advertisement. The second grey section (b) denotes the Memberhsip Service parameters. For the case of PSE, that's the peer's public key, encapsulated in a PKIX certificate.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jxta:PA>
<jxta:PA xml:space="default" xmlns:jxta="http://jxta.org">
  <PID>urn:jxta:uuid-59616...7B03</PID>
  <GID>urn:jxta:uuid-425A5C703CD5454F9C03938A0D65BD5002</GID>
  <Name>Peer_1</Name>
  <Desc>Created by NetworkManager</Desc>
  <Svc>
    <MCID>urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000001105</MCID>
    <Parm>
      <jxta:PipeAdvertisement>
         <Id>urn:jxta:uuid-425A5...4704</Id>
         <Type>JxtaUnicast</Type>
         <Name>ANONYM-PIPE:Peer_1</Name>
       </jxta:PipeAdvertisement>
    </Parm>
  </Svc>
  <Svc>
    <MCID>urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000000805</MCID>
    <Parm>
      <jxta:RA xml:space="preserve" xmlns:jxta="http://jxta.org">
        <DstPID>urn:jxta:uuid-59616...7B03</DstPID>
        <Dst>
          <jxta:APA>
            <EA>jxtatls://uuid-5961...B03</EA>
            <EA>cbjx://uuid-59616...B03</EA>
            <EA>tcp://213.73.36.53:9701</EA>
          </jxta:APA>
        </Dst>
      </jxta:RA>
    </Parm>
  </Svc>
  <Svc>
    <MCID>urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000000105</MCID>
    <Parm>
      <RootCert type="jxta:cert">MIIBkT...bV7</RootCert>
    </Parm>
  </Svc>
</jxta:PA>
```

Fig. 1. Peer Advertisement supporting the Anonymity Service, containing (a) Anonymity Service Pipe Advertisement and (b) PSE public key data.

Whenever the Anonymity Service is executed, it is enough that a new service parameter entry is created at the Peer Advertisement. When the service is closed, the entry is removed.

Message Setup:

This procedure is only executed by the peer which actually wants to send the anonymous message, comprising how the the anonymous message is initialized to be sent through a set of anonymizing peer relays.

1) A peer S decides to send a request to any JXTA Service $JXTASvc$, being executed at a destination peer D. Such request is structured as a standard JXTA message, $JXTAMsg$, according to the syntax specification expected by the service.
2) $JXTASvc$'s Pipe Advertisement, $FinalSvcPipe$, is retrieved using the Discovery Service.
3) Up to this point, the previous steps follow JXTA's standard operation, and must be performed whenever a service is accessed, notwithstanding anonymous messaging. Now, it is decided that the message will be sent to $JXTASvc$ anonymously.
4) S generates a bit string, $RndData$, with a randomly chosen length between 560 and 1120 bytes. The reasons for this will be explained in IV-A.
5) S generates an *OnionCore* structure. This structure is a JXTA message composed by the following name-value pairs:
   - `RandomData` $= RndData$
   - `FinalServicePipe` $= FinalSvcPipe$
   - `JXTAMessage` $= JXTAMsg$
6) Using the Discovery Service, S retrieves a set of Peer Advertisements $PAS = Adv_1, \ldots, Adv_n$, where it is true that each Advertisement contains an Anonymity Service parameter entry. The bigger the set, the better anonymity degree it is achieved, but a value of 3 is considered good enough [7]. We will refer to each Advertisement owner as an *Onion Peer*. It is recommended that PAS contains no duplicates, even though not strictly necessary. Nevertheless, it should be true that $\forall Adv_i, Adv_i \neq Adv_{i-1}$.
7) For each Advertisement Adv_i in PAS, from $n \ldots 1$, the following process is iteratively executed. $Onion_i$ is considered the result of each iteration:
   a) This iteration's input, *data*, is chosen.
      i) For the first iteration ($i = n$) the *OnionCore* structure is considered the input.
      ii) For the rest of iterations ($i = n-1, \ldots, 1$), Adv_i's PID field (the peer's unique identifier) is retrieved. Then a *OnionLayer* structure is generated, which will act as input. This structure is a JXTA message composed by the following name-value pairs:
         - `NextHop` $= PID$
         - `OnionMessage` $= Onion_{i+1}$
   b) The peer's 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.
   c) A cryptographic symmetric key SK_i is randomly generated.
   d) $Enc_i(data)$ is created by encrypting the input data with a secure symmetric key algorithm, using SK_i as the secret key.
   e) $Enc_i(key)$ is created by encrypting SK_i with a key wrapping algorithm, such as the one defined in [16], using PK_i as the key.
   f) An *OnionMessage* structure is generated. This structure is a JXTA message composed by the following name-value pairs:
      - `OnionData` $= Enc_i(data)$
      - `WrappedKey` $= Enc_i(key)$
   g) The *OnionMessage* structure becomes this iteration's result ($Onion_i$).
8) S looks up the Anonymity Service definition entry in Adv_1 and retrieves the contained Pipe Advertisement.
9) S opens an output pipe using such advertisement, and directly sends $Onion_1$.

Message Processing:

This procedure describes how an anonymous message is processed whenever it is received by any peer which has deployed the Anonymity Service. A summary of the Message Processing procedure, showing the different structures used during the onion routing process, is presented in Figure 2.

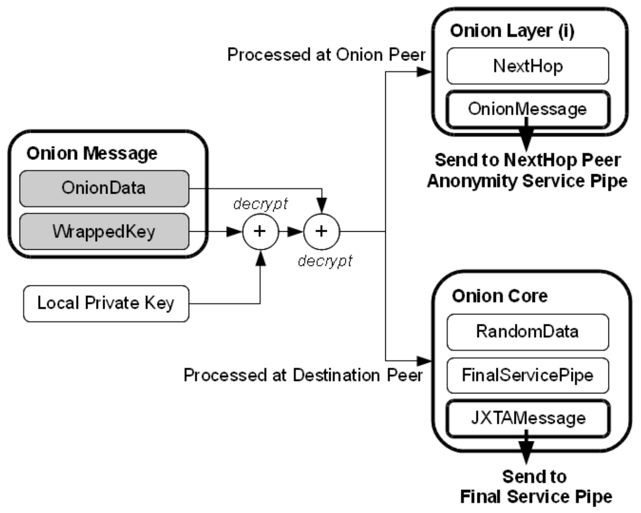


Fig. 2. Onion message processing. Encrypted message fields are denoted in grey.

1) A peer executing the Anonymity Service receives an incoming *OnionMessage* through its input pipe.
2) The content of the `WrappedKey` field is decrypted using the peer's local private key. It is guaranteed that such key exists, since it was initialized when the peer joined the group via de PSE Membership Service. The result is SK_i.
3) The content of the `OnionData` field is decrypted using SK_i. Then, two different things may happen:
   a) The decrypted data results in an *OnionLayer* structure. This occurs when the message is processed by an *Onion Peer*.
      i) Its `NextHop` field is extracted.
      ii) The Discovery Service is used to locate a Peer Advertisement, Adv, containing a PID field equal to `NextHop`.
      iii) The Anonymity Service definition entry is looked up in Adv. The contained Pipe Advertisement is retrieved.
      iv) The peer waits a random amount of time $RndTime$.
      v) An output pipe is established, using the Pipe Advertisement. *OnionLayer*'s `OnionData` field content (an *OnionMessage* structure) is sent to the pipe.
   b) The decrypted data results in an *OnionCore* structure. This is only true for the final destination, D.
      i) The original JXTA message, $JXTAMSg$, is retrieved from the `JXTAMessage` field.
      ii) The actual final service Pipe Advertisement, $FinalSvcPipe$, is retrieved from the `FinalServicePipe` field.
      iii) The `RandomData` field is discarded. No processing is required.
      iv) A local open pipe connection is established to $JXTASvc$ using its Pipe Advertisement. It is then used to send $JXTAMSg$. It must be noted that, from $JXTASvc$'s standpoint, the fact that the message has undergone and anonymizing process has been completely invisible.

An overview of the message onion routing procedure is detailed in Figure 3. In the figure, to clearly show how data transformations comes about, message processing is considered to take place at three distinct layers. The actual data to be sent is created at the Application layer. That data conversion to a JXTA Message transmission using service pipes happens at the JXTA Basic Service Layer. The Anonymity Layer invisibly takes care of the whole anonymizing process at a lower layer.

IV. Anonymity Service evaluation

In this section the proposed anonymity layer is evaluated from two different standpoints. First of all, from a security point of view, analyzing how it is able to counter typical attacks to anonymity. Secondly, from a performance angle, assessing the resulting overhead over standard messaging.

A. Security evaluation

Attacks used to compromise anonymity in P2P networks can be categorized into those with a local attacker or a global one. The former is considered a peer group member, only able to analyze messages it relays. In contrast, the latter is able to monitor the whole network, being able to intercept any message and locate the originator and destination of any intermediate message exchange. The proposed protocol in this paper only focuses in avoiding local attacks, the most common ones, since measures to counter global attacks have a tremendous impact on performance.

The most important existing local attacks are:

- **Key retrieval:** Any time a peer requests any public key, in order to create onion layers, it becomes evident that it is about to send an anonymous message. In this proposal, key retrieval is performed along Peer Advertisement propagation, a standard JXTA procedure routinely performed by peers. Thus, it cannot be used to pinpoint suspect peers.
- **Packet size analysis:** As an anonymous packet travels across the netwrok, its size decreases. Since the size of the anonymity layers and and the final Pipe Advertisement is known, the last relay can easily spot whether the next hop is actually the final destination. The included `RandomData` avoids this attack, by inserting unknown fake size equivalent to extra hops (each Onion Layer amounts to 560 bytes).
- **Packet timing:** Analyzing message reception dates, it may also be possible to link messages from a same source peer which reuses paths, when network latency

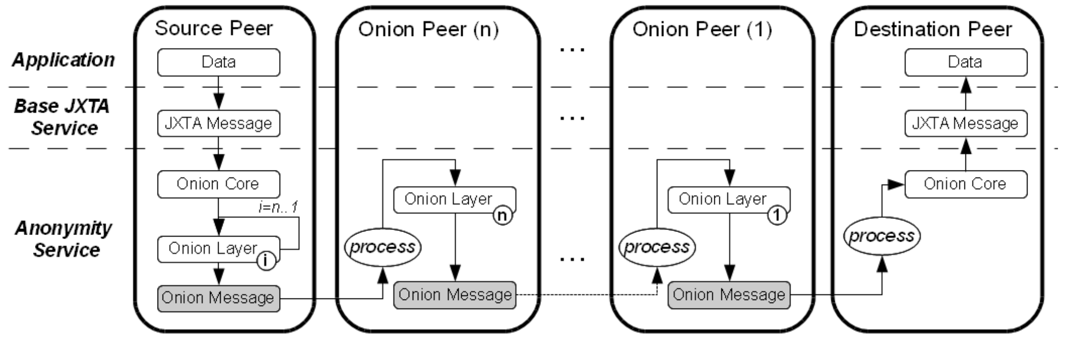


Fig. 3. Onion routing procedure.

can be consistently estimated. Waiting a random amount of time before relaying a message, in step 3.iv of message processing, disrupts this kind of analysis.

B. Performance evaluation

The incurred overhead in messaging as a result of applying the anonymity layer may greatly vary depending on the actual message content and the amount of random data blocks generated in step 4 of the message setup process. Furthermore, overhead decreases at each hop, as each layer is processed. However, it is possible to measure a theoretical worst cas scenario, considering an empty message (no data content).

A single hop Onion Message incurrs in a 203-378% overhead over an empty message (640 bytes). This is also the overhead range of a message at its last hop. Each additional hop increases the overhead in about 87.5%. Thus, a standandard 3 layer message results in a 378-553% maximum overhead boundary. The values are quite high, but decrease for messages with actual content. For example, a 1Kb data message has an overhead range of 163-151% at its starting point.

It can be concluded that, even though onion routing systems are classified as “medium ovehead”, from a standard protocol standpoint, it is quite high nevertheless.

V. Conclusions

A proposal for an anonymity layer in JXTA has been presented. Apart from the fact that JXTA currently does not provide anonymous messaging, the main contributions of the chosen approach are twofold.

First of all, it fully realizes JXTA’s capabilities, 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.

Secondly, the anonymity layer is almost invisible to the actual service being accessed. From the service provider’s standpoint, the original message has been normally received through its published input pipe. No additional processing is required from that services’ standpoint. Therefore, the anonymity layer may be applied to any standard JXTA service.

Further research goes toward extending the anonymity layer to support more complex JXTA connection types, such as bidirectional pipes, which basically requires presetting an onion return path and moving from a stateless service to a stateful one. Finally, it is also worth studying how to apply mechanisms that thwart global attackers.

References

[1] Kocher P. C. Freier A. O., Karlton P., “The SSL Protocol v3.0”, 1996, http://wp.netscape.com/eng/ssl3/draft302.txt.

[2] 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.

[3] Wallace J. D., “Nameless in cyberspace: Anonymity on the internet”, 1999, http://www.cato.org/pubs/briefs/bp-054es.html.

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

[5] 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.

[6] 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.

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

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

[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] Zhang X. Xiao L., Xu Z., “Low-cost and reliable mutual anonymity protocols for peer-to-peer networks”, *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 9, pp. 829–840, 2003.

[11] 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.

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

[13] Liu Y. HAn J., “Rumor riding: Anonymizing unstructured peer-to-peer networks.”, *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, pp. To appear, 2010.

[14] Shields C. Levine B.N., “Hordes: A multicast based protocol for anonymity”, *Journal of Computer Security*, vol. 10, no. 3, pp. 58–70, 2002.

[15] CCITT, “The directory authentication framework. recommendation”, 1988.

[16] J. Staddon B. Kaliski, “PKCS1: RSA Cryptography Specifications. Version 2.0”, 1998.