Chapter 15
High Assurance Asynchronous Messaging Methods

William R. Simpson and Kevin Foltz

15.1 Introduction

This paper is based in part on a paper published by WCECS [1]. Asynchronous messaging describes communication that takes place between one or more applications or systems, in which the sender does not receive feedback from the receiver during transmission of a message. This is in contrast to synchronous communication, in which the sender of a message waits for acknowledgement or a response from the receiver before completing the transmission.

There is no assumption about which layers asynchronous and synchronous communication take place in or how these relate to each other. It is possible to implement synchronous communication using an asynchronous messaging service or an asynchronous messaging service using synchronous communication channels. In practice, asynchronous messaging often uses an underlying synchronous channel.

A common asynchronous messaging design involves one system placing a message in a message queue and continuing its processing. At the completion of message transmission, the sender does not know when or whether the receiver received it. The message queuing system is responsible for delivering the message to the recipient. Some systems use two or more queues or intermediaries.

W.R. Simpson (✉) · K. Foltz
Institute for Defense Analyses, 4850 Mark Center Drive, Alexandria, VA 22311, USA
e-mail: rsimpson@ida.org

K. Foltz
e-mail: kfoltz@ida.org

S.-I. Ao et al. (eds.), *Transactions on Engineering Technologies*,
DOI 10.1007/978-981-10-2717-8_15

15.1.1 Some Advantages of Asynchronous Communication

Asynchronous messaging solves the problem of intermittent connectivity. If the receiving equipment fails or is unavailable, the message remains in a message queue and is delivered after the failure is corrected. This is especially useful for transmission of large data files, in which failures are more likely and retransmissions more costly.

An asynchronous messaging system with built-in intelligence may transform the content and/or format of the message automatically to conform to the receiving system's requirements or needed protocol but still successfully deliver the message to the recipient. This intelligence is used to provide a higher level of understanding of the content, which allows translation into other formats and protocols. Complicated transformations are better suited to asynchronous communication than synchronous communication because they may increase latency and cause connectivity problems or other underlying protocol failures for synchronous systems.

15.1.2 Some Disadvantages of Asynchronous Communication

The disadvantages of asynchronous messaging include the additional component of a message broker or transfer agent to ensure the message is received. This may affect both performance and reliability. Another disadvantage is the response time, which may be inconvenient and not consistent with normal dialog communication.

15.2 High Assurance Issues

There are several high assurance security principles that must be maintained for asynchronous communication.

1. Know the players—in synchronous (two way communication) this is done by enforcing bi-lateral end-to-end authentication. For synchronous communication Enterprise Level Security the certificate is PKI.
2. Maintain Confidentiality—in synchronous Communications this entails end-to-end encryption using Transport Layer Security (TLS).
3. Enforce Access and Privilege—in synchronous (two way communication) this is done by the use of an authorization credential. For synchronous communication Enterprise Level Security the certificate is SAML.
4. Maintain Integrity—know that you received exactly what was sent—know that content has not been modified—For synchronous communication and enterprise level end-to-end TLS encryption the use of message authentication codes is enforced. Packages are signed and signatures are verified and validated. Credentials of signers are verified and validated.

15.3 Prior Work

A proliferation of standards [2–12] for asynchronous messaging has caused interoperability problems, with each major vendor having its own implementations, interface, and management tools. Java EE systems are not interoperable, and Microsoft's MSMQ (Microsoft Message Queuing) does not support Java EE. Many of these are reviewed and compared in [13]. A few of the numerous standard protocols used for asynchronous communication as defined in the Internet Assigned Numbers Authority (IANA) protocol registries [14] are in the Table 15.1.

Table 15.1 Messaging ports

Port	TCP/UDP	Messaging Protocol and Description	Status
18	TCP and UDP	The **Message Send Protocol (MSP)**, more precisely referred to as Message Send Protocol 2, is an application layer protocol used to send a short message between nodes on a network. Defined in RFC 1312	Official
110	TCP	**Post Office Protocol v3 (POP3)** is an email retrieval protocol	Official
119	TCP	The **Network News Transfer Protocol (NNTP)** is an application protocol used for transporting Usenet news articles (*netnews*) between news servers and for reading and posting articles by end user client applications. Defined in RFC 3977	Official
143	TCP	**Internet Message Access Protocol (IMAP)** is a protocol for e-mail retrieval and storage as an alternative to POP. IMAP, unlike POP, specifically allows multiple clients simultaneously connected to the same mailbox and through flags stored on the server; different clients accessing the same mailbox at the same or different times can detect state changes made by other clients. Defined in RFC 3501	Official
161	UDP	**Simple Network Management Protocol (SNMP)** is an "Internet-standard protocol for managing devices on IP networks." Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks, and more. Defined in RFC 3411-3418	Official
218	TCP and UDP	**Message Posting Protocol (MPP)** is a network protocol used for posting messages from a computer to a mail service host	Official
319	UDP	Event Messages for The **Precision Time Protocol (PTP)** is a protocol used to synchronize clocks throughout a computer network. On a local area network, it achieves clock accuracy in the sub-microsecond range, making it suitable for measurement and control systems. Defined in **IEEE 1588-2008**	Official
587	TCP	**Simple Mail Transfer Protocol (SMTP)**, as specified in RFC 6409	Official

(continued)

Table 15.1 (continued)

Port	TCP/UDP	Messaging Protocol and Description	Status
1801	TCP and UDP	**Microsoft Message Queuing** or **MSMQ** is a message queue implementation developed by Microsoft and deployed in its Windows Server operating systems	Official
1863	TCP	**MSNP (Microsoft Notification Protocol)**, used by the Microsoft Messenger service and a number of Instant Messaging clients	Official
1935	TCP	Adobe Systems Macromedia Flash Real Time Messaging Protocol (RTMP) "plain" protocol	Official
2195	TCP	Apple Push Notification service Link	Unofficial
2948	TCP and UDP	**Multimedia Messaging Service (MMS)** is a standard way to send messages that include multimedia content to and from mobile phones	Official
4486	TCP & UDP	**Integrated Client Message Service (ICMS). Defined in RFC 6335**	Official
5010	TCP	**IBM WebSphere MQ Workflow**	Official

15.3.1 Java Standard Messaging Protocol

Java Messaging System (JMS) is a message-oriented middleware API for communication between Java clients. It is part of the Java Platform Enterprise Edition. It supports point-to-point communication as well as publish-subscribe.

15.3.2 De-Facto Standard Microsoft Message Queuing

Microsoft Message Queuing (MSMQ) allows applications running on separate servers/processes to communicate in a failsafe manner. A queue is a temporary storage location from which messages can be sent and received reliably, as and when conditions permit. This enables communication across networks and between computers running Windows, which may not always be connected. By contrast, sockets and other network protocols require permanent direct connections.

15.3.3 Open Source Messaging Protocols

In addition to Java and Microsoft, different open source solutions exist [2–12]. RabbitMQ is an open source messaging solution that runs on multiple platforms and multiple languages. It implements Advanced Message Queueing Protocol (AMQP), in which messages are queued on a central node before being sent to

clients. It is easy to deploy, but having all traffic pass through a single central node can hinder scalability.

ZeroMQ is another cross-platform, cross-language messaging solution that can use different carrier protocols to send messages. It can support publish-subscribe, push-pull, and router-dealer communication patterns. It can be more difficult to set up, but it provides more control and granularity at the lower levels to tune performance.

ActiveMQ is a compromise between the ease of use of Rabbit MQ and the performance of ZeroMQ. All three support multiple platforms and have client APIs for C++, Java,.Net, Python, and others. They also have documentation and active community support. There are many other implementations, including Sparrow, Starling, Kestrel, Beanstalkd, Amazon Simple Queue Service (SQS), Kafka, Eagle MQ, and IronMQ.

15.3.4 Emerging Standard Advanced Message Queuing Protocol

Advanced Message Queuing Protocol (AMQP) is an open standard application layer protocol for message-oriented middleware [15]. It is an emerging technology addressing the standardization problem. Implementations are interoperable. It includes flexible routing and common message paradigms like publish/subscribe, point-to-point, request-response, and fan-out.

The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability, and security. AMQP mandates the behavior of the messaging provider and client to the extent that implementations from different vendors are truly interoperable, in the same way as SMTP, HTTP, FTP, and others have created interoperable systems.

15.4 Asynchronous Messaging Security

Asynchronous messaging can provide authentication of the sender and receiver identities and the integrity and confidentiality of the message content if the holder of the queue is trusted. One key challenge in asynchronous messaging systems is that a third party is often involved in the transaction, which may or may not be trusted to speak for the sending or receiving entities or view or modify content in transit. As a result, security models often require a trusted third party, which restricts deployment options. In contrast, synchronous web traffic relies on routers and other infrastructure to deliver messages, but the use of TLS provides end-to-end security without the need to trust these intermediate nodes.

15.4.1 Security for Server Brokered Invocation

Server brokered invocation uses web server middleware to manage message queues. The sender and receiver both communicate directly through secure synchronous channels to the server to send and receive messages. This model is shown in Fig. 15.1. Asynchronous message security must be from sender to receiver, not just from sender to server and server to receiver. The latter fails to provide end-to-end authentication, integrity, and confidentiality, which are required for a high assurance environment.

In order for the parties involved in the transaction to provide accountability, integrity, and confidentiality, the service requester must authenticate itself to the receiver, encrypt the message so only the service provider can receive this message, and provide verifiable integrity checks on the full message content. The service provider must confirm that the message is from a known identity, decrypt the content with a valid key, and verify the integrity checks before that entity can take action on the message.

This is accomplished by invoking two cryptographic techniques. The first is the use of a digital signature by the sender. When the message signature is verified, the service provider knows the identity of the sender and that the content has not been altered by another entity after it was signed. The second is the encryption of the message using the public key of the service provider. This requires that the requester know the public key of the target. A response to the requester must similarly be signed and encrypted using the public key of the requester.

The use of asymmetric encryption is paired with more efficient symmetric encryption, where content is encrypted with a random symmetric key, which is itself encrypted using the receiver's public key. Additional security can be provided by message expiration deadlines within queues and central auditing of all messages sent and received.

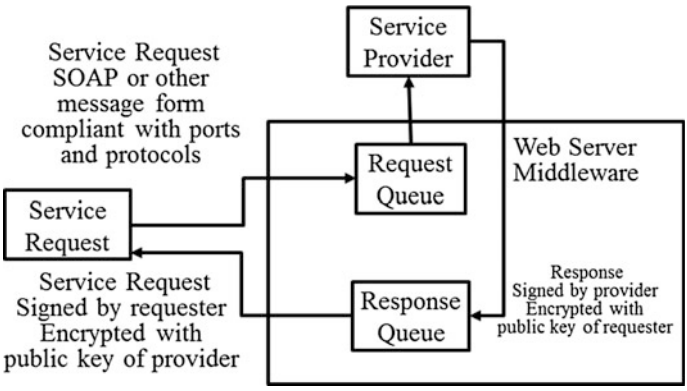

Fig. 15.1 Security considerations for server brokered invocation

15.4.2 Security for Publish Subscribe Systems

In a Publish Subscribe System (PSS) the queue server acts as an intermediary between sender and receiver to manage many-to-many instead of just many-to-one communications. Senders and receivers communicate with the PSS through a secure synchronous channel. The PSS collects messages and makes them available to entities based on subscriptions. This model is shown in Fig. 15.2.

The PSS is an active entity and registered in the Enterprise Service Directory. Active entities act on their own behalf and are not a proxy. To preserve the end-to-end accountability chain for messages, the original publisher signs the message. However, unlike server-brokered invocation, no single public key can be used for all potential receivers. One solution to address this is for the PSS to encrypt the content to the receivers. The sender's signature remains intact, preserving integrity, but end-to-end confidentiality is not guaranteed.

A PSS may use the web server broker as shown in Fig. 15.3. The web server broker is used only for notification messages, so it does not require security like the main channel. The transmission of the actual message is still done through the secure synchronous channel. The storage queue must be encrypted using the PSS's public key. This is piecemeal confidentiality, because the sender encrypts to the PSS, and the PSS encrypts to the receiver. This relies on trust of the PSS.

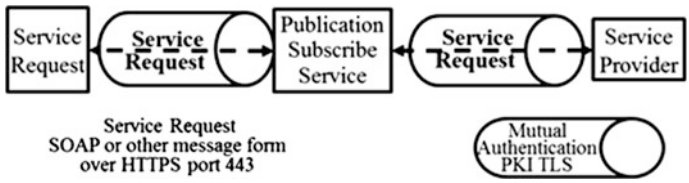

Fig. 15.2 Publish-subscribe push model

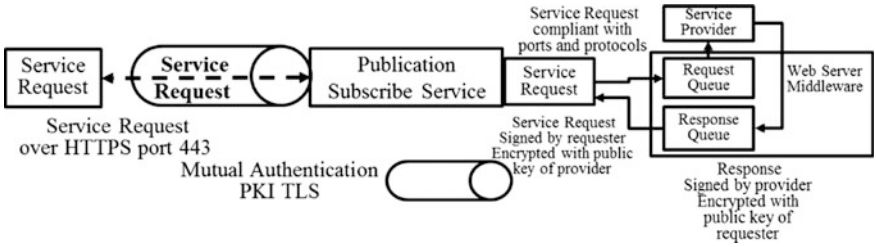

Fig. 15.3 Publish-subscribe pull model

15.5 Pss Rock and Jewel

The following is an approach developed to maintain high security assurances with the use of an untrusted PSS. In this formulation, the sender and receiver maintain end-to-end security because the PSS is unable to impersonate either endpoint or view or modify the content. The key concepts are the use of "rocks" and "jewels" to provide security guarantees. The "rocks" are encrypted content blocks, and the "jewels" are the decryption keys for these rocks, encrypted using public keys for the intended recipients.

15.5.1 Claims for Targeted Content (PSS)

After authentication through TLS v1.2 or later versions and authorization based on SAML claims, the sender accesses PSS services. The PSS will offer either publish or retrieve based on the values in the SAML content claim. If there are no SAML content claims, the subscriber will only receive basic services based on identity.

Publishing of content for a targeted list, as used by software publishers, is based upon registered delivery. The targeted list requires the following steps:

0. Publisher does a bi-lateral authentication and establishes a TLS 1.2 session with SAML authorizations for session establishment with the PSS. The PSS identifies him as a publisher. He may also be a subscriber, or he may be modifying a previous publish or he may be retrieving messages, so the PSS ascertains the reason for his session.
1. Content to be published will be digitally signed by the publisher.
2. The publisher will generate an AES-256 encryption key and encrypt the content.
3. Encrypted Content is placed in a queue based on an access claim and list name. The publisher will keep such lists. The PSS will assist in developing claims.
4. Access is based on a list of targets and claims. A target may be an individual subscriber or a group queue. The publisher may establish a new queue based on claims and the list for retrieval. This new queue requires an identity and a claims establishment for retrieval (see 3 above). Additional content may be published as needed.
5. Expiration time of targeted content is determined by the publisher or the messaging system.
6. The PSS will provide PKI certificates for each of the targets for the content (if the publisher needs them and they are already registered in the PSS). The publisher should check all certificates on the list for currency and revocation. If invalid certificates are discovered, the list should be pruned.
7. The publisher will prepare encrypted key sets (jewels) by wrapping the AES encryption key in each target's public key.

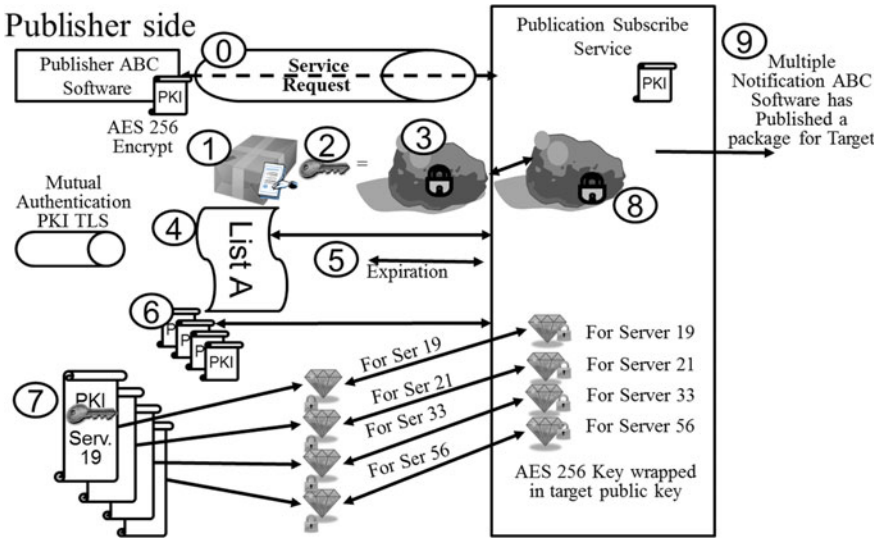

Fig. 15.4 Publishing of targeted content

8. The publisher will publish the encrypted material (rocks) and the encrypted key sets (jewels) for the targets. The PSS will link these to the encrypted material and the target(s).
9. The PSS will provide notification, if desired, to the subscriber list. The PSS will assist with message selection and target details, or the publisher may script his own.
10. The publisher closes the session.

Note: the target must be on the list and have authorization to view content. The steps are shown in Fig. 15.4.

15.5.2 Retrieving Content for Known Claimants

Retrieval of targeted content may be achieved without the targeted identities contacting the publisher. The following steps are followed:

0. Subscriber does a full bi-lateral authentication using TLS 1.2 with SAML authorizations for session establishment with the PSS. The claims identify him as a subscriber. He may also be a publisher, so the PSS ascertains the reason for his session.
1. The PSS offers subscriber content available for the claims in queues for which the claimant has an encrypted key available, and the subscriber chooses and retrieves the encrypted content (rock).

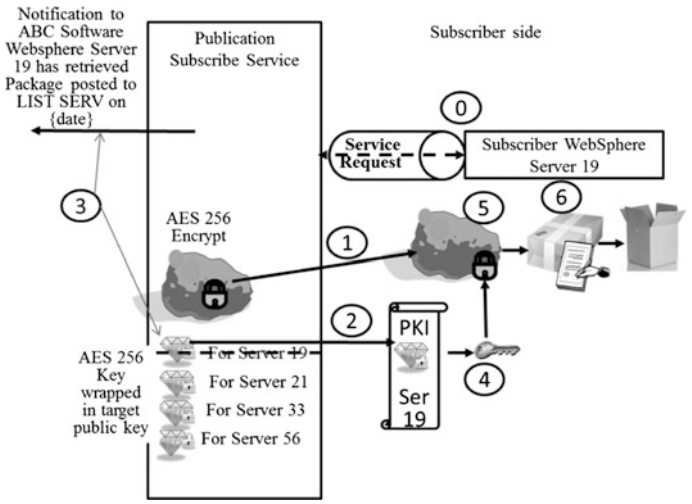

Fig. 15.5 Subscriber retrieval(s) from a known target

2. The PSS provides the encrypted key package (jewel).
3. The PSS notifies the publisher. When expiration time occurs, the server deletes the packages and notifies the publisher which packages were not delivered. The publisher may republish to that list if desired.
4. The subscriber decrypts the content encryption key (jewel) with his private key and accesses the content (rock) decryption key.
5. The subscriber decrypts the content.
6. The subscriber verifies and validates signature.
7. The subscriber closes the session or retrieves additional content.

Note: the target must be on the list and have a content claim. The steps are shown in Fig. 15.5.

15.5.3 Retrieving Content for Unknown Claimants

Unknown claimants cannot retrieve the content until registering with the content provider. The steps in that process are described below:

0. The subscriber does a full bi-lateral authentication TLS 1.2 with SAML authorizations for session establishment with the PSS. The authentication identifies him as a subscriber. He may also be a publisher, so the PSS ascertains the reason for his session.
1. The PSS checks the content claims available and the subscriber chooses and retrieves the content for which full packages exist.

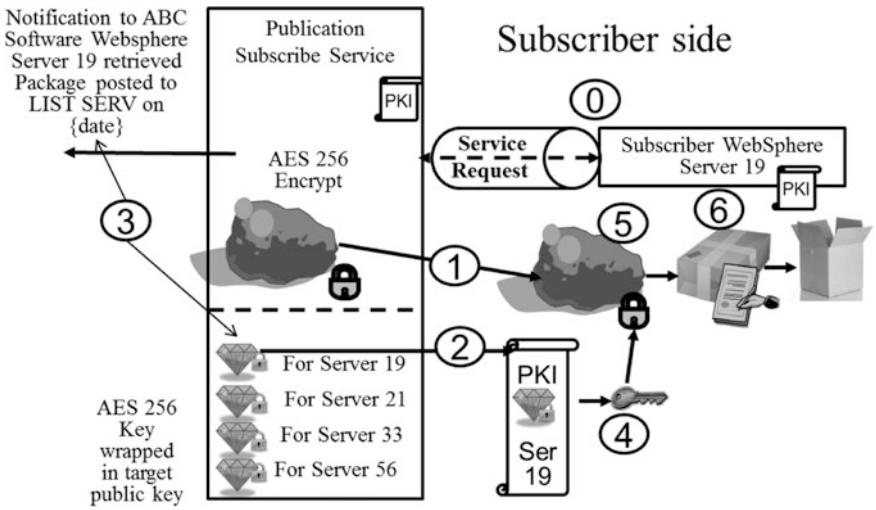

Fig. 15.6 Subscriber retrieval(s) from an unknown target

2. For the unknown list, the encrypted key package is not available. The PSS replies "the publisher has no record of your membership. I need to contact the publisher. I will send you a notice if the publisher agrees."
3. The PSS stores a message for the publisher and notifies him that he has a message.
4. The PSS and subscriber await publisher action.
5. The subscriber closes the session or retrieves additional content.

Note: the target has a content claim, but is not on the list. The steps are shown in the Fig. 15.6.

15.5.4 Adjusting Publishing Targets (Untrusted PSS)

Publishers must add receivers to the distribution list before they can be provided messages. The steps in that process are described below:

0. The publisher does a full bi-lateral authentication through TLS 1.2 with SAML authorizations for session establishment with the PSS. The authorization process identifies him as a publisher. He may also be a subscriber, or he may be modifying a previous publish or he may be retrieving messages, so the PSS ascertains the reason for his session.
1. Retrieve messages. These are retrieved one by one with action taken (or not) and deletion of the message.

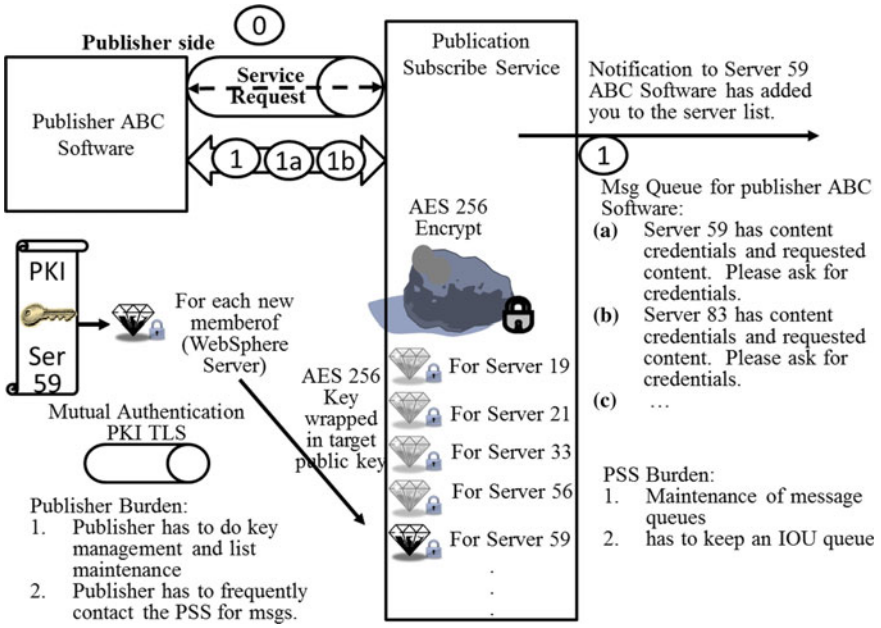

Fig. 15.7 Publisher message retrieval and subsequent actions

- The publisher asks for credentials of previously unknown claimants he wishes to add to his lists.
  - The publisher may add claimants to the publisher's list
  - The publisher computes jewels.
  - The publisher posts jewels.
  - The PSS notifies the subscriber that he has content available. This makes the entity a known target and Sect. 15.5.2 applies.
- PSS provides messages to requester.

2. The publisher closes the session.

The steps are shown in Fig. 15.7.

15.5.5 Distribution of Burdens

Several burdens are incurred in this high security mode. The publisher has to do key management and list maintenance. The publisher has to frequently contact the PSS for messages for publishers. The PSS must maintain message queues for publishers.

The PSS has to keep a linked wrapped key package by target with published content. The PSS is responsible for additional notifications that are sent out. The unknown claimant may have a delay in receiving content to which he has claims.

15.6 Summary

We have reviewed the basic approaches to asynchronous communication in computing environments. We have also described high assurance approaches to the process. The proliferation of standards in this area has created a problem with high assurance. In many instances the high assurance elements require additional steps in the asynchronous process, but they provide a way to proceed when some intermediaries are untrusted. This work is part of a body of work for high assurance enterprise computing using web services. Elements of this work are described in [16–29].

References

1. Foltz K, Simpson WR (2015) Maintaining high assurance in asynchronous messaging. In: Proceedings World Congress on Engineering and Computer Science 2015, WCECS2015. Lecture Notes in Engineering and Computer Science. San Francisco, USA, pp 187–192
2. World Wide Web Consortium (W3C): XML Encryption Syntax and processing. http://www.w3.org/
3. Organization for the Advancement of Structured Information Standards (OASIS) open set of Standards: Web Services (WS) standards and Security Assertion Markup Language (SAML) standards. https://www.oasis-open.org/standards
4. National Institute of Standards, Gaithersburg, MD: Encryption related standards. http://www.nist.gov/srm/
5. PKCS#1: RSA Cryptography Standard: ASN Module for PKCS#1 v2.1, June 14, 2002
6. MSMQ—http://msdn.microsoft.com/en-us/library/ms711472(v=vs.85).aspx
7. RabbitMQ—http://www.rabbitmq.com/
8. Apache ActiveMQ—http://activemq.apache.org/
9. Apache Qpid—http://qpid.apache.org/
10. JBoss HornetQ—http://www.jboss.org/hornetq
11. ZeroMQ—http://www.zeromq.org/
12. WebSphereWMQ–http://www-01.ibm.com/software/integration/wmq/
13. Message Broker Comparison– http://lifecorporatedev.blogspot.com/2012/07/recently-i-have-been-given-task-to-find.html
14. Internet Engineering Task Force (IETF) Standards: IANA and Protocol standards including The Transport Layer Security (TLS). https://www.ietf.org/
15. Advanced Message Queuing Protocol (AMQP)—https://www.amqp.org/
16. Simpson WR, Chandersekaran C, Trice A (2008) A persona-based framework for flexible delegation and least privilege. In: Electronic Digest of the 2008 System and Software Technology Conference, Las Vegas, Nevada, May 2008

17. Simpson WR, Chandersekaran C, Trice A (2008) Cross-domain solutions in an era of information sharing. In: The 1st International Multi-Conference on Engineering and Technological Innovation: IMET2008, vol I, Orlando, FL, June 2008, pp 313–318
18. Chandersekaran C, Simpson WR (2008) The case for bi-lateral end-to-end strong authentication. In: World Wide Web Consortium (W3C) Workshop on Security Models for Device APIs. London, England, December 2008, pp 4
19. Simpson WR, Chandersekaran C (2009) Information sharing and federation. In: The 2nd International Multi-Conference on Engineering and Technological Innovation: IMETI2009, vol I, Orlando, FL, July 2009, pp 300–305
20. Chandersekaran C, Simpson WR, (2010) A SAML framework for delegation, attribution and least privilege. In: The 3rd International Multi-Conference on Engineering and Technological Innovation: IMETI2010, vol 2, Orlando, FL, July 2010, pp 303–308
21. Simpson WR, Chandersekaran C (2010) Use Case Based Access Control. In: The 3rd International Multi-Conference on Engineering and Technological Innovation: IMETI2010, vol 2, Orlando, FL, July 2010, pp 297–302
22. Chandersekaran C, Simpson WR (2011) A model for delegation based on authentication and authorization. In: The First International Conference on Computer Science and Information Technology (CCSIT-2011). Lecture Notes in Computer Science. Springer Verlag Berlin-Heildleberg, pp 20
23. Simpson WR, Chandersekaran C (2011) An agent based monitoring system for web services. In: The 16th International Command and Control Research and Technology Symposium: CCT2011, vol II, Orlando, FL, April 2011, pp 84–89
24. Simpson WR, Chandersekaran C (2011) An agent-based web-services monitoring system. Int J Comput Technol Appl (IJCTA), 2(9):675–685
25. Simpson WR, Chandersekaran C, Wagner R, (2011) High Assurance Challenges for Cloud Computing. In: Proceedings World Congress on Engineering and Computer Science 2011, Lecture Notes in Engineering and Computer Science, vol I, San Francisco, October 2011, pp 61–66
26. Chandersekaran C, Simpson WR (2012) Claims-Based Enterprise-Wide Access Control. In: Proceedings World Congress on Engineering 2012, The 2012 International Conference of Information Security and Internet Engineering. Lecture Notes in Engineering and Computer Science, vol I, London, July 2012, pp 524–529
27. Simpson WR, Chandersekaran C (2012) Assured content delivery in the enterprise. In: Proceedings World Congress on Engineering 2012, The 2012 International Conference of Information Security and Internet Engineering. Lecture Notes in Engineering and Computer Science, vol I, London, July 2012, pp. 555–560
28. Simpson WR, Chandersekaran C (2012) Enterprise high assurance scale-up. In: Proceedings World Congress on Engineering and Computer Science 2012. Lecture Notes in Engineering and Computer Science, vol 1, San Francisco, October 2012, pp. 54–59
29. Chandersekaran C, Simpson WR (2012) A uniform claims-based access control for the enterprise. Int J Sci Comput 6(2):1–23. ISSN:0973-578X