Fine-grained and Scalable Approaches for Message Integrity

Joon S. Park and Ganesh Devarajan
The Laboratory for Applied Information Security Technology (LAIST)
School of Information Studies
Syracuse University
Syracuse, New York, USA
{jspark, gdevaraj}@syr.edu

Abstract

When we have multiple users compiling a single message, including shared contents, metadata, policy, and so on, the integrity of the contents created by individual users needs to be maintained in an effective manner. There is an urgent need for new mechanisms in a trusted content-sharing environment to support multiple signers for the same message, which can be dynamically updated with autonomous protection and maintenance mechanisms. In our previous work we identified and compared three different binding mechanisms including monolithic, autonomous, and chained binding mechanisms, using digital signatures. The original work was designed and implemented for digital certificates. In this paper we apply those schemes with extension to the organization-to-organization messaging services that require effective integrity verification and tracking mechanisms at the user-level in the receiving organization in a scalable manner with fine granularity. We implement our ideas in the Java Remote Method Invocation (RMI) platform and discuss the trade-offs of the signature schemes based on our experimental results.

1 Introduction

When data is transferred between different sensitive divisions, domains, or even organizations, reliable control must be taken to maintain the integrity of the messages and files that were passed. In particular, if security levels of users are different, information must not flow from the high level to the low level [1, 4, 8, 10]. In such cases we should verify the security level of the data being transferred from one user to another. To validate data being transferred between different organizations we enforce devices such as Guards, which are also known as Boundary Controllers, or Cross Domain Solutions (CDS). Any critical information that is to be transferred from Organization A to Organization B has to pass through the Guard, which checks, filters, and sanitizes data between the organizations based on the policies. The Guard executes pre-defined security policies that are based on the trust level, guidelines, rules, directives, and instructions. Basically, when we interlink two organizations of different security levels, we should prevent sensitive data from being transmitted to unauthorized organizations or persons. The Guard also maintains logs for all requests and responses that are passed across.

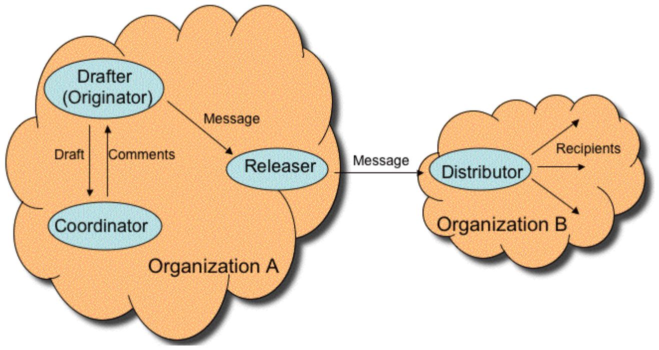


Figure 1. An Example of Organization-to-Organization Messaging Services.

Considering message source and destination, generally, there are two different kinds of services: person-to-person and organization-to-organization. The former is just like ordinary e-mail service, where each sender selects the receiving person. The latter is for organization-level communications (e.g., formal military messaging systems [9, 16]), where the sender does not know who is going to read the message in the receiving organization. When we need security in person-to-person service, we can easily use existing user-level security services such as PGP [19] or PKI [2, 3] tools. However, in the case of organization-to-organization service, such user-level security cannot be used. When sending, the sender does not know whose cryptographic key should be used to encrypt or sign the message, because the receiving individual is not yet determined by the sender. Figure 1 shows a typical example of organization-to-organization messaging services. A user (drafter) can initiate communication by writing or revising a message. Before the message is sent to another organization, the coordinator reviews the message and requires necessary changes in the message based on the organization's policy. If the message is ready to be sent, the releaser of the sending organization sends the message to the distributer in the receiving organization. Once the message is delivered, the message is distributed to the corresponding recipients based on the organization's current policy. In this paper, we focus on the message integrity in the organization-to-organization services, while many other approaches have been introduced for the person-to-person service by other researchers.

2 Related Work

The technical solution for preventing unauthorized release of sensitive information between different domains is referred to as *boundary control*. A boundary controller is a software solution designed to enforce the rules and policies of an organization to control the information flow between the organization and the outside world, as well as between internal units. A high-accuracy boundary controller is essential in supporting the information sharing that is required within an organization where there are groups that need to share some, but not all, information with different levels of security. Technically, a boundary controller is an automated guard whose generic operation is depicted in Figure 2. For future tracking of transactions, it maintains log files for both sending and receiving organizations.

The Advanced Research Guard for Experimentation (ARGuE [5, 11]), developed by the NAI labs, is one of the working models of boundary controllers. Once a client initiates a request, it is sent to the ARGuE where the request is processed, and if it does not violate any of the policies it is sent to the server on the other end. The processed result is sent back to ARGuE where verification of the security policies takes place and the response is given to the other end. Simultaneously, logs are developed and maintained for each and every transaction that takes place between the organizations.

The Air Force Rome Laboratory (AFRL) developed the Imaginary Support Server Environment (ISSE) Guard [17, 7]. ISSE provides a secure interface in the transaction of softcopy information between two hierarchically varying divisions within local or wide-area networks. The ISSE Guard is built in with Common Guard Interface (CGI). The ISSE Guard supports bi-directional traffic from High to Low or from Low to High security clearance. The ISSE Guard provides the functionality required for securely connecting, validating, downgrading, and transferring information between systems and networks operating at different security clearance levels. It permits secured transaction of digital images, files, and texts between organizations of different security clearance levels by downgrading the information, if transferred from organizations of high security levels to organizations with low security clearance. These are built in with email and file-scanning, sanitizing, downgrading tools that work on XML-based products.

The Cross-Domain Solution (CDS [6]) was developed by the Department of Defense, and the Department of Navy. CDS was developed to send and receive data between different levels of organizations. It is a blend of technologies (guard, components), policies (rules, instructions, and procedures), and threat environments (reliability, trust level, and security clearance) in which it should be deployed. NetSec developed a CDS system that offers high Assurance for data transfer, but its data transfer takes place in one direction alone.

The Message Analysis Downgrade and Dissemination (MADD [17]), developed by AFRL, is a product developed to sanitize or downgrade the message and then pass it on to the user at the other end. This works only for well-formatted messages. It incorporates the message extraction technologies developed by AFRL to enable automated filtering, downgrading, sanitization and dissemination, which are in the USMTF (US Message Text Format).

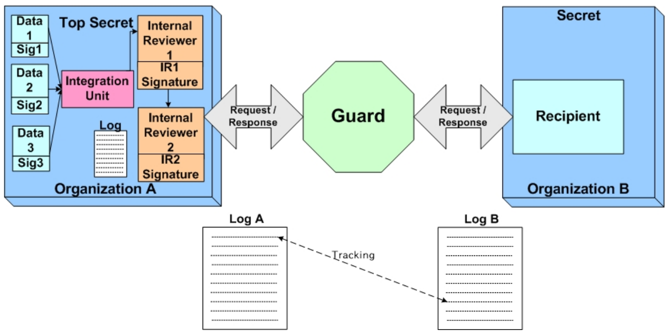


Figure 2. A Generic Operation of Guard.

3 Operational Scenarios

When Alice is sending a message to Bob in a different organization, Alice composes the message and then digitally signs it. This signed packet is sent to the guard, where the contents of the message are verified with current policies that are set for message transfer. In most currently available approaches, if all the contents are allowed for transfer by the policies, then the guard strips out Alice's signature and appends its own signature. This message with the guard's signature is sent to Bob. Subsequently, Bob verifies the guard's signature and retrieves the contents. Typically, Bob does not need to verify that the message was originally signed by Alice, because Bob trusts the guard. In the meantime, the guard creates entries in the log files for future reference purposes.

The operation can be extended with internal reviewers. An internal reviewer (IR) verifies the individual data (message) sent by each user and compiles them into a single data unit before sending it to the guard. Multiple internal reviewers can verify the data before sending it to the guard. Figure 2 illustrates how the data is integrated when we have three users compiling data to be sent to the other organization. The integration unit combines these messages and sends them to IR1. The integration unit strips out the signature of the preceding users and holds just the signature of the last user. This procedure is followed until it reaches the final user. After the message is integrated into one unit it is sent to the IR1. When the data comes out of IR1, the message will have only IR1's signature. Similarly, when the message comes out of IR2, the signature of IR1 will be stripped out and the signature of IR2 will be appended. Once both internal reviewers are done with their verification, the message with the IR2's signature is sent to the guard.

This conventional operation might be the simplest solution when a message is written by a single user and user-level verification is not required at the time of message receiving. However, when multiple users, especially with different security levels, are involved in the contents of the same message before it is transferred to the final receiving organization, it is challenging to provide and verify message integrity in a scalable manner with fine granularity. Furthermore, it is not always possible for the recipient to track the users who are involved in the message creation on the senders' side, although the transactions are logged, especially when the log file of the senders' organization is maintained by a different administration, which is not unusual. To overcome these problems, we propose the following advanced operational scenarios for providing message integrity.

Advanced Operation: Each user who is involved in the message signs his or her portion individually. After the guard verifies the compiled message and individual signatures, it attaches its signature to the message. The final message includes the guard's signature as well as the individual signatures.

The proposed operation will provide effective integrity verification and tracking mechanisms at the user level. Furthermore, it increases service granularity. However, it will also introduce the scalability issue in terms of signature maintenance. In the following sections, we describe how we can support the above proposed scenario in a scalable manner with fine granularity .

4 Scalable Digital Signature Schemes with Fine Granularity

When we have multiple users compiling a single message, including shared contents, metadata, policy, and so on, the integrity of the contents created by individual users needs to be maintained in a scalable manner with fine granularity. There is an urgent need for new mechanisms in a trusted content-sharing environment to support multiple signers of the same message, which can be dynamically updated, with autonomous protection and maintenance mechanisms. In our previous work we identified and compared three different binding mechanisms of digital signatures, including monolithic, autonomous, and chained schemes [13, 14, 15]. The original work was designed and implemented for digital certificates. In this paper we apply the digital signature schemes with extension to the organization-to-organization messaging services to support the advanced operation described above that provides fine-grained and scalable user-level integrity to messages.

Of all the signature schemes, the simplest case is when the contents could be signed by one single authority. In this case, we define that the protection and maintenance mechanisms are monolithic. In this case, however, it has difficulty in tracking the integrity at the user level, especially when multiple users with different security levels are involved in the same message. In other words, it is hard to check who provided specific contents to a message and to support fine-grained integrity check unless a log file is maintained with detailed user-level histories. For tracking, the system has to go through the log files generated during the transactions and then trace the perpetrator. However, maintaining a powerful logging database itself, which should be protected in a sensitive system, is not a trivial work. Even if it exists, it usually takes a long time to track the necessary data, and the accuracy of the tracking depends on the logging mechanisms and information. Sometimes, the message-sending organization does not allow other organizations (including the receiving organization) to access its log files. Furthermore, in a traditional scheme, each time a new portion is added to the message, the signer should sign the entire message. This approach is neither efficient nor scalable for a large messaging system. Finally, generating and maintaining log files that can support our advanced operation become complex, especially when messages are merged and separated frequently by users with different security levels. Therefore, we focus on the autonomous and chained schemes with enhancement to support the proposed scenarios in a scalable manner with fine granularity.

In the autonomous scheme, a binder of the original contents and the new contents are digitally signed by an authorized person. The main advantage of the autonomous scheme is that the binders and contents are loosely coupled, which means that, even after the original and new contents are linked by the binder, we can still modify the other portions of the original contents without breaking the link to the new contents as long as the binder and the corresponding signature remain the same. One binder can be used to link different contents and the same contents can be linked by different binders (a many-to-many relationship). The linking mechanism is analogous to the situation where a person can use any ID card to prove that he or she is the owner of his or her credit cards, as long as the names on the credit card and ID card match. This scheme provides fine granularity and scalability because only relative portions of a message are linked to new contents by digital signatures.

Once the entire message is compiled by the autonomous scheme, the message is sent to the Internal Reviewer, who verifies each and every part of the message to confirm that the message does not violate company policies. After this screening process the integrated message is sent to the guard. If the messages do not violate policy settings, then they are allowed to pass through to the other organization. When the user from the other organization requests this message, the guard annexes its own signature by the autonomous scheme just as the users did on the message. If the requesting organization is in a lower security clearance, then the guard downgrades the message to the security clearance level of the requesting domain. However, if there is any portion of the data that is violating policy settings, it is immediately sanitized or downgraded and these kinds of activities are recorded in the log file. After the filtering and scanning process the file is passed on to the requesting organization.

The chained scheme is a particular case of an autonomous scheme where we use the digital signature of the original contents as a binder. In this case the binders and other linked contents are tightly coupled, which means that once the original and new contents are linked by the binder (i.e., the signature of the original contents), we cannot modify any portion of the original contents without breaking the link to the previous signatures. Each signed entity can produce only one unique binder, digital signature, with the same key in the chained scheme. Once the digital signature has been generated, we cannot change any portion in the signed contents without breaking the integrity of the entire original contents, while one signature can be linked to many other new contents (a one-to-many relationship). The linking mechanism is analogous to the situation in which a person should present his or her particular ID (such as a passport) in a foreign country to prove that he or she is the owner of his or her credit cards because they do not understand or trust other IDs such as driver's license cards issued by other countries.

In this scheme, we use the signature as the binding information. In this way the flexibility of adding new contents in the middle of the compiled message is going to be stricter than in the autonomous scheme. However, at the same time, it provides strong integrity verification while it still provides a finer granularity than that of the traditional approaches. Even a slight modification in the header or body part of the message will be reflected in the signature. If we make any modification in the original contents, then the corresponding signature should be regenerated in the message that follows. In this scheme the messages are tightly coupled. When the Internal Reviewer verifies the message before sending it to the guard it can easily determine the amendments made by other users. The guard annexes its own signature by the chained scheme just as the users did on the message. The rest of the functionalities of the guard are similar to those described in the autonomous scheme.

5 System Development

Based on the advanced scenario and the support mechanisms described in the previous section, we developed a prototype. The prototype was implemented in the Java Remote Method Invocation (RMI) platform with XML digital signatures [18]. The RMI framework provides a platform for developing distributed client-server applications. We employed the Java Cryptographic Extensions (JCE) APIs in our implementation. JCE is a set of packages that provides framework and implementations for encryption, key generation and key agreement. JCE integrated into the Java 2 SDK. The KMS (Key Management Server) is implemented as an RMI server object and the users are mapped as corresponding RMI clients. The entire application was developed using the XML Security Suite in Java developed by the IBM Alpha Works. The other packages used in this project are the Security Package of Java. The Java Security Package developed for the Java 2 Platform, Standard Edition (J2SE), introduced policy-based access control, X.509 v3 [12] implementation of certificate interfaces, and tools for creating and managing security keys and certificates. J2SE 1.4 continued by adding Java Authentication and Authorization Service (JAAS), Java Cryptography Extension (JCE), Java Secure Socket Extension (JSSE), and features for Kerberos communication.

The following Data Flow Diagram characterizes the flow of information in our prototype from one screen to another, Class Diagram illustrates the class dependencies that exists between the classes, and the Activity Diagram shows the operations that are carried out in each module and the information that is passed from one module to other.

5.1 System Overview

In the beginning of the operation a user specifies the task that he or she intends to perform. If the user, Alice, wants to create a new message, she clicks on the *Create New Message* button. The next thing that pops up is the authentication screen where she will have to specify the user name, password and then the role in which she is authorized to access the system. Now the XML guard sends the compose-message screen onto the user's system, which is equipped with the keys required for the different sections of the message, depending on the role that the user logged into the system. Once the message is composed the user encrypts the data using the specified shared secret keys, and then the messages are individually signed using the specified private key of the user. For instance, if the user has composed a message that contains a section Secret and Classified, then she signs the Secret section with her private key for the secret security clearance level and, similarly, uses a Classified private key for the classified security level of the message. After all the parts of the message are encrypted and signed, they are wrapped and sent to the XML guard. The document is analyzed and then processed here. The signatures of the users are verified and must account for any modifications made on the way to the guard. Now, if the same message were to be accessed by any other users within the same domain of the same security clearance level, it would be passed on to that user after proper authentication. The user is capable of appending or modifying data, depending on the message integration scheme adopted for this message. Similarly, after authentication, the user is supplied with an Append message screen with the required keys. Since the user is authorized to view the entire contents of the message, she can view the entire encrypted message and hence has the ability to decrypt the entire message. Now she appends the contents into the fields of the message, depending on the message integration scheme, and then she encrypts the message, followed by signing the message. The message is then sent to the XML guard.

In such case the message has to be sanitized. For instance, if the user who is accessing the Top Secret message has a security clearance of Secret, then the Top Secret message content is stripped out of the message and then versioned and this message is sent to the user who requested the message. This is all done only after authentication of the user. Now the screen sent to the user is based on the user's security clearance, and is supplied with only those keys that are required by her. After the user is done with her appending of the message she encrypts, she signs the message and sends it to the XML guard.

In this manner, messages are integrated while maintaining their integrity. Now suppose the message is to be sent to a user who is on another domain, then she authenticates

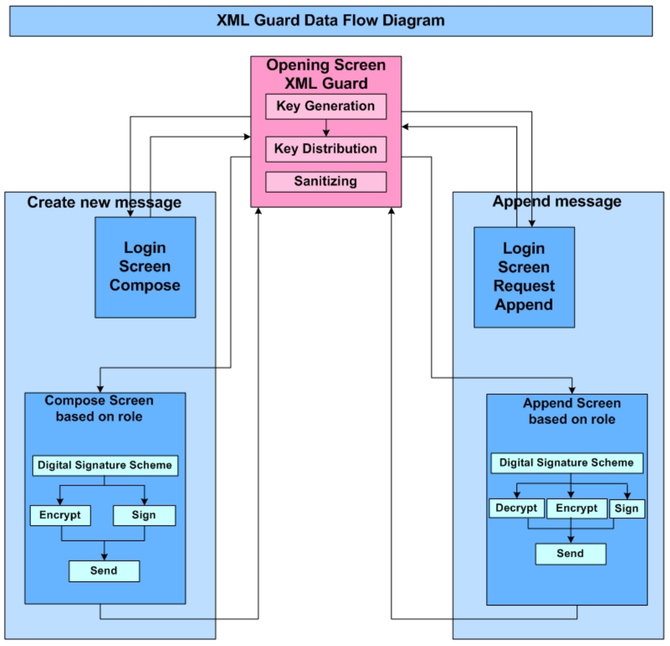


Figure 3. Data Flow Diagram in XML Guard.

herself into the system and posts a request. The XML guard verifies the authenticity of the user and then the request that she posted. Subsequently, the XML guard signs the message with its own private keys for a different security clearance level based on either autonomous or chained schemes. Once a user, Bob, receives the message in the receiving organization, he can easily verify the signatures of the guard as well as of the users who were involved in the contents of the message.

5.2 System Architecture

Following is the class diagram showing the architecture and information flow in the client's system. The major modules in the implementations are Key Generator, KMS, Digital Signature Schemes, Encryption, Decryption, XML Converter and RMIRegistry.

The KMS initially registers itself with the RMI Registry, a naming service in the Java RMI Framework, to make it available to the world. The User is configured as an RMI Client. The User class module can get a reference to the KMS Server object by querying the RMI Registry. Once it gets a reference to the KMS Server object, then the User class module can access the methods of the KMS just as it can access any other method calls. The User Interface class module has a Graphical User Interface (GUI) that is implemented using Java Swing packages. This class module basically gets all the user inputs and commands through the GUI and passes them to the User class module for further processing and sends the processed result back to be displayed in the GUI. The User class module also has instances to the Encryption and Decryption class modules. Intuitively, the Encryption and Decryption class modules have all the mechanisms for encrypting and decrypting user messages. Messages composed by the user need to be signed using one of the signature schemes. The class relationship that exists between the message and the digital signature scheme is *composition*. While the Digital Signature Scheme class aggregates monolithic, autonomic, and chained message integration schemes, these schemes inherit the User Interface class. Depending on the Message Integration Scheme, the User interface is displayed to the user. When the user opens other users' messages she needs to verify the contents before displaying them. The user class module aggregates decryption, encryption, Verification, XML Converter and the Digital Signature Scheme.

We have different User Interfaces for different signature schemes. The monolithic scheme takes in different messages signed by various users and integrates them into one single message. The user who is integrating the message fills in the header information and then the messages are placed in order and the entire message is signed using her signature. There is a possibility of the message being corrupted on its way to the integrating user or that the integrating user herself might have corrupted the message that was created by another user. In such cases, the signature of the authentic user who originally created the message does not match the existing one. When we have such types of mismatch in signatures, the XML guard filters these messages out and sanitizes them.

The working mechanism of the monolithic message scheme is that we first integrate the signed messages into one complete message. The user who is integrating this message signs it. After the message is signed it is sent to the XML guard. Here, different parts of the messages are verified. The verification of the signature is performed first. We see if the signature created for the contents match the signature created by the user who initially created this message. After the different parts of the messages are verified, we perform the second level of testing. The second level basically verifies the contents of the message. When we are communicating between different security clearances we should make certain that critical information does not pass through the organizations boundary. This is where the contents of integrated messages are verified in the XML guard. After this verification, it is passed on with the guard's signature embedded with the integrator's signature. In this way, the user on the receiving end is certain about the integrity of the information she has received.

There are scenarios where we have more than one person who integrates the work of her group and then forwards it to her superior. Finally, the superior integrates all of the work and passes it on to the client in the other organization.

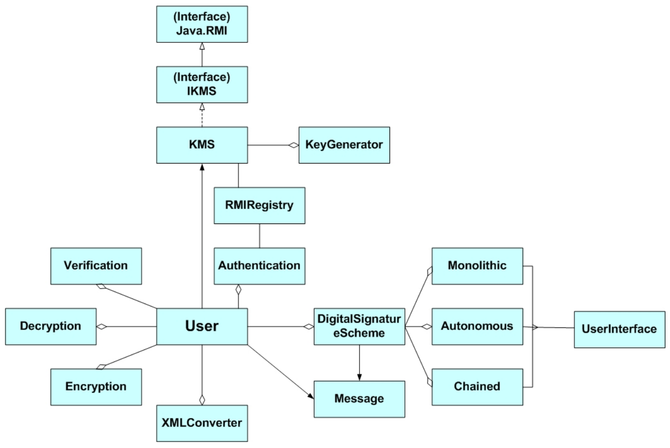


Figure 4. Class Diagram of Signature Schemes.

In these situations it is best that we use the autonomous or chained message integration scheme. In the autonomous scheme we implemented, a user first authenticates herself into the system where she wants to upload or download a file. During authentication she specifies her security clearance as per the level of her security clearance that her message composition screen pops up. In that, itself, the author of the message has the privilege to construct a message that might be lower in security clearance than what she currently is logged into. For example, say user Alice logs into the system with a Top Secret security clearance but wants to construct a message to be sent to a Secret level domain. She can lower her security clearance to Secret and then construct a message to that effect. Once the user has created the message, it is encrypted with the user's shared secret key with the XML guard. The user has different shared secret keys for each and every security clearance that are of her level and the levels below her. For instance, supposing she logs into the system as a Classified user, then she has the Classified and Unclassified shared secret keys with the XML guard. This message is now signed using the user's private key so that any other user who opens this message can verify the authenticity or the correctness of the data. Now, suppose another user, say Chris wants to add some information to the current message, then he authenticates himself into the system and asks the XML guard for this file. Chris could use a part of the header or the body of the message that was created by Alice as binder information to integrate his work into it. Once this is done, Chris encrypts his appended message and signs the file using her private key. This message is deposited in the file repository that is set up in the XML guard. Similarly, more people can append their contents to the base file that Alice created and branch out, hence creating a branched message in the autonomous scheme.

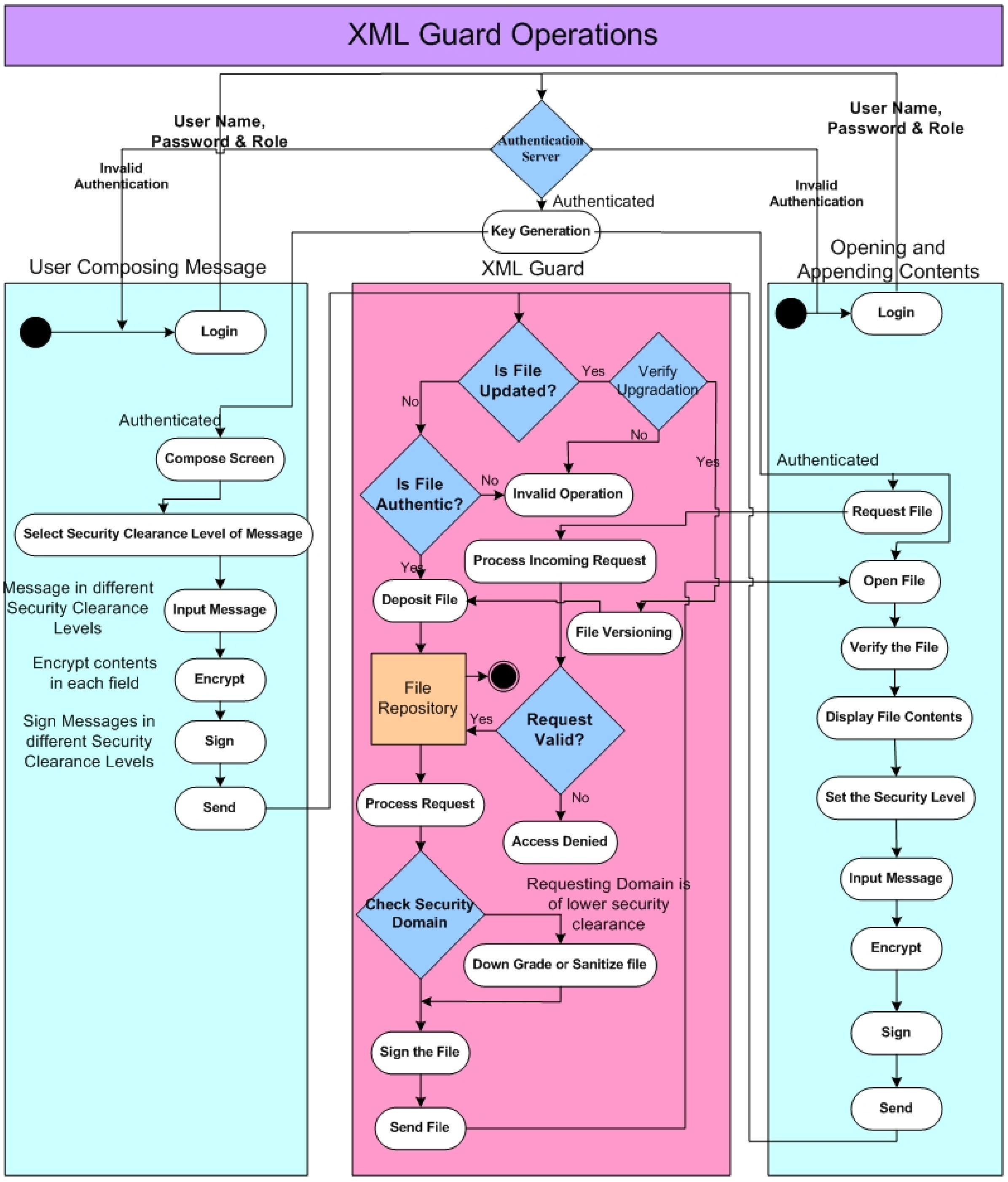


Figure 5. Activity Diagram of XML Guard Operation.

5.3 XML Guard Operation

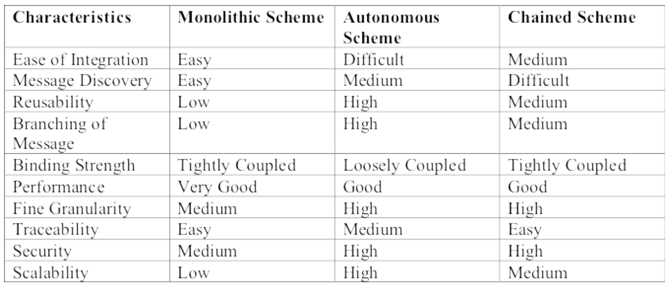
Figure 5 shows the working of the autonomous scheme of message binding. Only authenticated users are supplied with the keys for that session. If the user requests the composing of a new message, the screen opens up, depending on the role and the security clearance of the user who has logged in. The author has the privilege of downgrading security clearance to compose a message for the domains of lower security clearance. After that, she encrypts, signs, and then sends the message to the XML guard where the file is deposited in the File Repository. Now, supposing another user wants to open this message and modify a few fields, all that she needs to do is log into the system, then post a request to the XML guard. Here, the request is verified and the security clearance of the author trying to access the file is crosschecked with the security clearance of the message. If the security clearance of the message is higher than the security clearance of the requesting node, then the message is sanitized (the message sections which are not to be revealed for the requesting domain are all stripped out) and then passed to the requesting user. If the security clearance of the message is lower than the security clearance of the requesting user, then no operation is performed and the entire message is passed on to the requesting user. Similarly, if the security clearance of the message and the requesting user is the same, then there is no operation performed. Now the modified file is encrypted and signed and then sent to the XML guard. In the XML guard the modified file contents are verified and then deposited in the file repository.

We can also adopt the chained message integration scheme where the signature of the previous user is used as binder information. Consider the same scenario where Alice is creating a message. Now the message has different parts, such as header and body, and the body has different parts which classify the different security clearance that Alice is authorized to operate in. This message is deposited in the file repository. Now, suppose Chris wants to append some related contents to the file that Alice has created. He will post a request to the XML guard and the guard checks to find out if the file has to be sanitized or not. After this verification the file is sent to Chris, who appends his bit of information to the message, signs the message, and sends it back to the XML guard. In this manner we can bind many messages from different users together in a chained format.

The final integrated message is sent to the organization or the user who requested this message. Before sending the message across the other domain the XML guard signs the integrated message based on either the autonomous or chained schemes and passes it to the requesting user or organization.

6 Discussion

Although the monolithic scheme is one of the simplest ways to integrate contents into a message, it does not provide as much fine granularity and flexibility as the autonomous and chained schemes do. The autonomous scheme has an edge over the rest of the schemes in terms of scalability and reusability.



Characteristics	Monolithic Scheme	Autonomous Scheme	Chained Scheme
Ease of Integration	Easy	Difficult	Medium
Message Discovery	Easy	Medium	Difficult
Reusability	Low	High	Medium
Branching of Message	Low	High	Medium
Binding Strength	Tightly Coupled	Loosely Coupled	Tightly Coupled
Performance	Very Good	Good	Good
Fine Granularity	Medium	High	High
Traceability	Easy	Medium	Easy
Security	Medium	High	High
Scalability	Low	High	Medium

Table 1. Characteristics of the digital signature schemes.

Table 1 summarizes the results of our experiments. To start with the ease of message integration, in the monolithic scheme only one user compiles the entire message and hence the message integration is very easy, whereas in the autonomous and chained schemes we have multiple users integrating the message and they need to specify the binder information for the integration. In the chained scheme we use just the signature of the previous message, whereas in the autonomous scheme we could use any part of the header or message as binder information. In terms of message discovery, which implies the ability to spot a particular message written by a specific author, the paths in the monolithic scheme are short and it is easy to determine a particular message. In the autonomous scheme, the paths are based on a shallow model with many branches and hence it is relatively easy to determine a message, whereas in the chained scheme the integrated messages run in a long stream and thus we need to verify the user details of each and every message in order to reach the actual message for which we performed the search. Branching of messages is very low in the monolithic scheme, very high in the autonomous, and moderate in the chained scheme. The reason is that in the monolithic scheme a single user inputs all the contents and hence not much of the branching. In the autonomous, branching can be done with different parts of the message as binder information and hence it is highly branched, and in the chained scheme we can use only the signature of the previous message as binder information. It is not branched, but at the same time it is a long sequence of messages. The reusability characterizes the parts of the message in different sections by providing some bit of modification to the message. In the monolithic scheme, even if a small portion of the message is modified it clearly affects the rest of the message. In the autonomous scheme, even if one-binder information values are altered, the rest of the message can still be kept without changing them. At the same time, in the chained scheme, if a message is modified all the messages beneath it will be affected and the messages above it stay undisturbed. Binding strengths of the monolithic and chained are tightly coupled, which means that even if one portion of the message is modified the integrity of the whole message following it is lost as well. Performance-wise the monolithic scheme is very good because they handle only a small number of messages. In the chained and autonomous schemes we have good performance, based on the operations that we have undertaken. In terms of control granularity, we can easily verify the contents put forth by each user in the autonomous and the chained scheme, whereas in the monolithic scheme, the entire message is the minimum control unit. In terms of scalability the autonomous scheme is the best, the monolithic is the worst, and the chained scheme is in between them, as we described before.

7 Conclusions

An XML guard performs operations such as sanitizing or downgrading the documents before being transferred between domains. To ensure a high level of message integrity, we have applied three new message integration schemes: monolithic, autonomous, and chained. In this paper we applied those schemes with extension to the organization-to-organization messaging services that require effective integrity verification and tracking mechanisms at the user-level in the receiving organization in a scalable manner with fine granularity. We implemented our ideas in the Java Remote Method Invocation (RMI) platform and discussed the trade-offs of the signature schemes, based on our experimental results.

References

[1] D. Bell and L. Lapadula. Secure computer systems: Mathematical foundations. Technical report, The MITRE Corporation, Bedford, MA, March 1973. MTR-2547.

[2] K. P. Bosworth and N. Tedeschi. Public key infrastructures - the next generation. *BT Technology Journal*, 19(3):44–59, 2001.

[3] S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu. Internet x.509 public key infrastructure certificate policy and certification practices framework. Technical report, Network Working Group, , United States, 2003. RFC3647.

[4] D. E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, 1976.

[5] J. Epstein. Architecture and concepts of the argue guard. In *the 15th Annual Computer Security Applications Conference (ACSAC)*, page 45, Washington, DC, USA, 1999. IEEE Computer Society.

[6] Galois. *Cross-domain solutions*, 2006. http://www.galois.com/xdomain.php.

[7] GlobalSecurity.org. *Information Support Server Environment (ISSE) Guard*, 2006. http://www.globalsecurity.org/intell/systems/isse-guard.htm.

[8] C. E. Landwehr. Formal models for computer security. *ACM Comput. Survey*, 13(3):247–278, 1981.

[9] C. E. Landwehr, C. L. Heitmeyer, and J. McLean. A security model for military message systems. *ACM Transactions on Computer Systems (TOCS)*, 2(3):198–222, 1984.

[10] L. J. LaPadula and D. E. Bell. Mitre technical report 2547, volume ii. *Journal of Computer Security*, 4(2-3):239–263, 1996.

[11] E. Monteith. Genoa tie, advanced boundary controller experiment. In *the 17th Annual Computer Security Applications Conference (ACSAC)*, page 74, Washington, DC, USA, 2001. IEEE Computer Society.

[12] M. Myers, C. Adams, D. Solo, and D. Kemp. Internet x.509 certificate request message format. Technical report, Network Working Group, United States, 1999. RFC2511.

[13] J. S. Park. Towards secure collaboration on the semantic web. *ACM Computers and Society*, 32(6), June 2003.

[14] J. S. Park, P. Chandramohan, A. Zak, and J. Giordano. Fine-grained, scalable, and secure key management scheme for trusted military message systems. In *Military Communications Conference (MILCOM)*, Monterey, CA, October 31 - November 3, 2004.

[15] J. S. Park and R. Sandhu. Binding identities and attributes using digitally signed certificates. In *the 16th Annual Conference on Computer Security Application (ACSAC)*, New Orleans, Louisiana, December 11-15, 2000.

[16] R. W. Shirey. The defense message system. *ACM SIGCOMM Computer Communication Review*, 20(5):48–55, 1990.

[17] the Air Force. *ISSE Guard*, October 2004. http://www.if.afrl.af.mil/tech/programs/isse/.

[18] W3C. *XML-Signature Syntax and Processing*, February 2002. http://www.w3.org/TR/xmldsig-core/.

[19] P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.