Extending the Liaison Workflow Model and Engine to Support Different Signature Purposes *

Karl R.P.H. Leung♠ Lucas C.K. Hui♡ Ricky W.M. Tang♡
♠Department of Computing and Mathematics,
Hong Kong Institute of Vocational Education (Tsing Yi),
Hong Kong.
email: kleung@computer.org

♡Department of Computer Science and Information Systems,
The University of Hong Kong,
Hong Kong.
emails: {hui, wmtang2}@csis.hku.hk

Keywords: Project Management, Workflow, CSCW, Signature

Abstract

Currently, many software systems are developed in offices geographically distributed in different locations. Furthermore, it is also common for a software system development project contracting to different software houses. These contracted software development projects, very often, are further sub-contracted to some other software houses. These software development modes can be supported and managed by a good distributed workflow systems. Signatures are playing an important role in these software development modes. Most workflow systems, at best, can only support digital signatures. Digital signatures with public key cryptosystem are limited to authentication, integrity, confidentiality and non-repudiation. The wide variety of signature purposes such as authorization or multiple signatures in group decision making are not supported explicitly by most workflow systems. We have studied different kinds of signature in software development and workflow systems. This paper discusses the problems and solutions of incorporating these signatures in distributed workflow engine, in particular, the Liaison Workflow Engine, to support the contemporary modes of software developments.

1 Introduction

With the development of communication networks, the diversity of economy in different countries and other factors, different phases of software developments or different components of a software system may be developed in different locations. Different components of a software system may be contracted to different software houses. The contracted-out components, in turn, may further be sub-contracted to some other software houses. We have reported our studies on supporting these sophisticated modes of software development management by distributed workflow systems, in particular, by *Liaison* workflow model [2, 4].

Software engineers would notice that signatures plays a very important part in these software development modes. All software development documents for communicating between different offices or different software houses have to be signed. These signatures, besides authentication, have other purposes such as authorization and group decision making.

In a public key cryptosystems [11] there are two kinds of keys namely, public key and private key. A message encrypted by a private key can only be decrypted by the corresponding public key, and vice versa. Digital signatures are going to be widely applied in e-commerce [5]. Digital signatures with public key cryptosystem can only support authentication, integrity and non-repudiation. However, the purposes of hand-written signatures are not limits to these. Depending on the context of the paper-based message, hand-written signatures are also used for different purposes, including but not limited to: authorization, generation of time stamp, providing evidences for accountability, and creating witness [6]. These properties are related to the organization model and behaviour, and are usually ignored by research studies in public key cryptography, which treats all types of content in a message as identical. Therefore, applying public key cryptography to solve the signature problem of workflow is insufficient [6]. Sig-

0-7695-0509-0/99 $10.00 © 1999 IEEE

natures can then be handled inappropriately. This lack of support for different purposes of digital signature renders many workflow systems cannot fully support different sophisticated software development modes.

The requirements of different purposes of signatures in workflow were studied [6]. These purposes are classified into single signature and multiple signatures and have been reported in [10] and [9], respectively.

In this paper, we report our studies on implementing different signature purposes in the *Liaison* Workflow Model [2, 4] and *Liaison* Workflow Engine [8]. *Liaison* has a relatively comprehensive model, language and architecture, and is easily extendible to incorporate a Signature Manager which provides signature services.

In Section 2, we review the *Liaison* model, *Liaison* architecture, and different signature purposes. The extension of *Liaison* Workflow Model is discussed in Section 3. The extension of the *Liaison* Workflow Engine are discussed in Section 4. These include data structures and the introduction of Signature Manager into the *Liaison* Workflow Engine Architecture. Realization of different purposes of signature by the Signature Manager is discussed in Section 5. Our design of handling signatures in workflow are discussed in Section 6. We then give our conclusion in Section 7.

2 Background

2.1 *Liaison* Workflow Model and *Liaison* Workflow Engine Architecture

The *Liaison* workflow model [2, 4] is an extension of the reference workflow model proposed by the Workflow Management Coalition [7]. The design of *Liaison* is based on the requirements of workflow systems reported in [1]. The model is also equipped with a workflow specification language *Valmont* [3]. It captures fundamental elements of the workflow paradigm: Organisation Model, Information Model, Process Model, and their relationships. Unlike the reference workflow model, it supports a rich organisation model and sophisticated activity assignment constraints.

The *Liaison* Workflow Engine [8] consists of two major components: the build-time engine and the run-time workflow engine. The build-time engine is responsible for parsing workflow specifications in *Valmont* to generate the *internal data*. The run time engine is responsible for managing workflow and task executions, actor assignments and information usage. In *Liaison* workflow engine, these responsibilities are handled by the co-operation of *Clock, Scheduler, Task Manager, Actor Manager, Information Manager, Actor Manager* and *Actor Interface Manager*. This architecture is shown in Figure 1.

The workflow control processes are shown in Figure 2. After selecting tasks to be executed, in step 1, the Scheduler passes the control of these tasks to Task Manager for processing. In step 2, the Task Manager requests Actor Manager to assign actors to process the tasks. If actors can be assigned, the Actor Manager acknowledges the Task Manager with the assignments in step 3. Afterwards, in step 4, Task Manager informs the Information Manager about the actor assignments. The Information Manager pre-

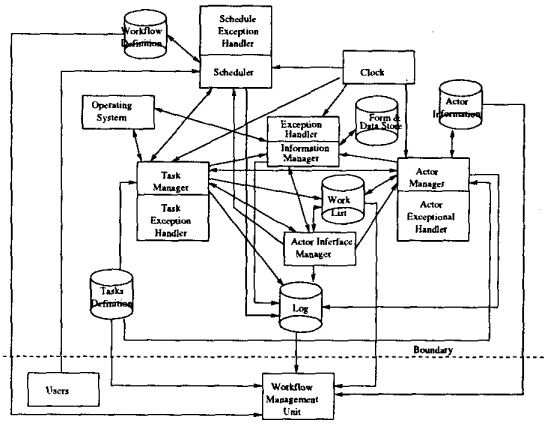

Figure 1. *Liaison* Workflow Engine Architecture

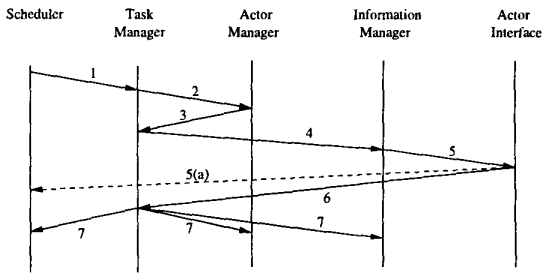

Figure 2. Workflow Engine Process

pare the forms and data, and informs the Actor Interface Manager that the tasks are ready for processing in step 5. Actors are then informed by the Actor Interface Manager that they can start processing the tasks. In step 5(a), if permitted by the task definitions, actors can fork new tasks during the processing by informing the Scheduler through the Actor Interface Manager. On completion of tasks, actors inform the Task Manager through the Actor Interface Manager in step 6. In the last step 7, the Task Manager informs the Scheduler, Information Manager and Actor Manager on completion of tasks.

2.2 Different Purposes of Signatures

Different documents such as system specifications are used in different phases of software development. These documents can be regarded as forms. For example, in a system specification, besides the contents of the specification, it also contains housekeeping information such as the author. These contents can be represented in a form by different attributes as different fields in the form. Signatures are required in many software development workflow processes. For example, the author of a system specification has to sign the specification. Here is a brief description of these purposes. More details can be found in [10].

Authentication This is the most popular usage of a signature. A person signs a form to prove the origin of this form. Other people who are able to verify the digital signature are able to find out who is the creator of this form.

Authorization Authorization is used in approval of invoking some tasks or allocation of resources.

Resources can be classified into two types: materials and facilities. Material resources include the information resources such as data and the physical materials such as pre-printed forms in the processing of a software development workflow. Facilities refers to the equipments supporting the processing of the workflow such using the production system for testing.

Very often the fields of the form are associated with resources. Successful verification of the authorization signature will trigger different processes to allow the intended recipient to use the resource, and updating the accounting information.

Proxy Signatures Very often in workflow processes, a person, called a proxy, is signing a document on behalf of another person. In the context of workflow systems, this means that a proxy signature is signed on the content of the form, and the signature should be accompanied by some evidence that the proxy had obtained authorization of his signing power.

Time stamping Signatures are also used to verify the time of creation of a form. When a signature is used for this purpose, both the details of the form and the current time and date has to be signed.

Accountability Signatures for this purpose are to certify the responsibility or accountability of certain actions. The form details should describe actions or procedures which require responsibility such as signing a user acceptance test report. Also information about the signer's position within the organization is needed,

There are two levels of accountability.

Level 1 — weak There exists an actor responsible for the consequences of the form. A signature signed by a valid actor of the organization would be sufficient for this purpose.

Level 2 — strong In additional to the requirement in level 1, form identity is checked against the actor responsibility list. This is to ensure that the actor has the right to be responsible for the form.

Being informed The signatures of this purpose means that the signers declare that the contents of the forms are known to them. Furthermore, only eligible actors should be informed of the contents of the forms.

Witness A witness signature is used to certify the happening of an event. There are two cases.

transparent The signer knows the contents of the form. The signature should be signed on information about the details of this event, and the details of participants.

opaque The signer does not know the contents of the form which is signed by the involved parties. The signer only signs on the signatures of the involved parties.

2.3 Multiple Signatures

In case of group decision making, signatures of more than one actors are required. It will be difficult to maintain the purposes of the multiple signatures if signing and validation of signatures are treated like separate single signatures. Furthermore, information from the organization model is needed to process multiple signature forms correctly. Therefore multiple signatures operations have to be considered separately[9].

2.3.1 Sequential Multiple Signature

In software development, some phase such as using the production system for testing, may require the signature of the project manager and then the signature of the manager of the production department. The form is passed to each of the actors one after another. We called this *sequential multiple signature*.

There can be different designs for different company policies:

Independent Sequential Multiple Signature: the order of approval is immaterial.

Dependent Sequential Multiple Signature: the actors are required to be signed in a specific order.

2.3.2 Parallel Multiple Signature

If copying of the form in a group decision making is allowed, copies can be sent to actors for signing. The signers are then sign the content of the form only. We call this mechanism of distributing form copies *fork*. Note that duplication of forms is more convenient in electronic media than in paper-based world.

Also there is a need of a mechanism to collect the signatures and thus continue the processing of the form. We can this mechanism *join*. According to the operations of sign and verify, fork and join can be further classified as follows.

fork-all In the previous example, if the approval of production system testing is not dependent on the sequence of signing, a copy of the form can be sent to the project manager, the manager of the production department.

fork-some If in the same example, in additional to the signatures of the two managers, it also requires the signatures of any two of the four operation staffs of production department, forms have to be sent to the two managers and at least two of the operation staff. This means that the number of copies can range from four to six.

join-all In the collection of the signed forms, the join-all case requires that all forms that are sent out in parallel should be collected.

join-some In the join-some case, there is no need to wait for all the signatures, but only those which make the conditions satisfied. In the example mentioned in the 'fork-some' case, the production system testing may go ahead after the responsible staff has received the signed form from the project manager,

the manager of the production department and any two of the operation staff. Furthermore, this is a case of voting where only a number of signatures are sought.

3 Extensions in *Liaison* Workflow Model

In order to handle signatures of different purposes properly, we first extend the *Liaison* workflow model with appropriate facilities such that all the required information for handling different purposes of signatures can be captured and modelled in the workflow specifications. Otherwise, it would be difficult for the automated workflow systems to determine the purposes of the signatures and carry out appropriate processing.

Liaison is a comprehensive workflow model. Most of the information for handling different purposes of signatures, such as organisation structures, have been captured by the model. The only extension required by the *Liaison* workflow model is the extension in data types in Information Model.

In *Liaison* the Information Model section consists of two subsection: Data Model and Form Model. The Data Model section define the fields and the data types in the forms to be used in the workflow system. The Form Model specify the fields included in the forms and the presentation format of the forms as well.

3.1 Data Model

Some new data types are required to be introduced into the Data Model of *Valmont*. In order to capture and support different signature purposes. These data types are discussed and expressed in DeMarco's data dictionary notation as follows.

signature This is a composite type for holding the information for signature operation. It consists of *signatureContentList*, *purposeList*, optionally *association* and *decision*.

$$signature = signatureContentList + purposeList + (association) + decision$$

signatureContentList The *signatureContentList* is used to hold all the signature contents which are kept by the *signature_content*. Fields of *signature_content* data type are used to hold digital signatures, identity of the signers and identity of keys which are used to create the digital signatures.

$$signatureContentList = 1\{signature\_content\}$$
$$signature\_content = actor\_identity + key\_identity + workflow\_signature$$

purposeList A signature may serve more than one purpose. For example, a signature for witness usually also has the purpose of accountability. This *purposeList* holds all the purposes of a signature. The data type *purpose* states different kind of purposes of signature supported by *Liaison*.

$$purposeList = 1\{purpose\}$$
$$purpose = [Authentication \mid Authorization \mid Time\_stamping \mid Accountability\_weak \mid Accountability\_strong \mid Proxy \mid Witness\_transparent \mid Witness\_opaque \mid Being\_informed]$$

associations This data type states the associate resources which may be required for the purposes of the signature.

decision This *decision* field hold the information about the signers and the mode of signature. It is a composite type which consists of at least one *group*. Each of the *group* is also a composite type which hold the information about the anonymity of the signatures in the group, the signers and mode of the group of signature. The *anonymity* data type holds the information whether the identities of the signers can be disclosed. The values of this type are *Open* and *Anonymous*. If it is only a single signature, there will be only one signer in the *signerList* and no *mode* field. The *mode* shows whether the group of signatures can have multiple copies, indicated by *Sequential* and *Parallel*, respectively. If it is of *Sequential* mode, it will further illustrate that whether it is of *Dependent* or *Independent* mode. In case *Parallel* mode, the fork amount and join policy will be specified. Join policy states whether the *Totality* of join, i.e. whether the multiple signatures are of *Mandatory* or *Voting* type. Both *ForkAmount* and *JoinAmount* are integers. This integers shows the minimum number of copies to be sent out for signature in fork or the minimum number of copies of signatures are required to be received in join. Hence the values of these two policies must be greater than zero and no greater than the number of signers in the *signerList*. Furthermore, if the value of the *ForkPolicy*, and respectively *JoinPolicy*, is equal to the number of signers in the *signerList*, this means **fork-all**, and respectively, **join-all**. All *signer* in the *signerList* are those actors who may sign the form. Signers can be expressed by means of positions or specific actors. These positions or actors must have been registered in the Organisation Model. Furthermore, there are at least one *signer* in the *signerList*.

$$decision = 1\{group\}$$
$$group = anonymity + signerList + (mode)$$
$$anonymity = [Open|Anonymous]$$
$$mode = [Sequential|Parallel]$$
$$Sequential = [Dependent|Independent]$$
$$Parallel = ForkAmount + JoinPolicy$$
$$ForkAmount = integer$$
$$JoinPolicy = Totality + JoinAmount$$
$$Totality = [Mandatory|Voting]$$

$$
\begin{aligned}
JoinAmount &= integer \\
signerList &= 1\{signer\} \\
signer &= [position \mid actor]
\end{aligned}
$$

4 Extensions in *Liaison* Workflow Engine

The *Liaison* Workflow Engine needs appropriate extensions to cope with the extension in Information Modelling. These extensions include some internal data structures and a Signature Manager (SM). The internal data are used by the workflow engine in the processing of the workflow. The SM is used to handle signature processing. These extensions are discussed as follows.

4.1 Extension in Internal Data Structure

The internal data structure required to be extended in the *Liaison* Workflow Engine is the *eligibility_list*. Every restricted resource and restricted task has a *eligibility_list*. These eligibility lists hold the tasks which are eligible to use the resources or invoke the task. All these restricted access resources and tasks must check the eligibility of the requesting task with the *eligibility_list* before it is being used or invoked by the corresponding task.

4.2 Signature Manager

The introduction of Signature Manager (SM) in the *Liaison* Workflow Engine is the major extension to handle signatures of different purposes. The features provided by SM, the co-operation of SM with other components of the *Liaison* Workflow Engine are discussed in this section.

Public key cryptography is used for the action of digital signature signing and verification. So extension on the *Liaison* Workflow Engine is required to manage the public and private key involved. Tasks include the typical public key cryptography functions, like: key generation, key distribution, key backup and recovery, key replacement, key revocation, and key termination. A high cohesion and loose coupling design is to introduce a software component Signature Manager (SM) to provide the public key management services. In addition, the SM should also provide appropriate signature services for different workflow purposes.

In order to facilitate authentications over distributed and mobile workflow systems, all actors are provided with identities which are signed by the private key of the SM. This SM signed identity is used for the SM and actors to authenticate the signer which signs the signature. This mechanism provides a basis for more reliable authentication. It is because the identity can only be decrypted with the SM's public key. If the signer's identity obtained from the decryption is different from the said signer, the said signer is an un-trustable actor and the signature is an invalid one.

In the following two sections, we discuss the general signature signing and verifying procedures. The realization of different signature purposes are discussed in Section 5.

4.2.1 General Procedures of Signing Signatures

The signing of a signature involves five major steps which are shown in Figure 3. In step one, signatures are initiated by an actor

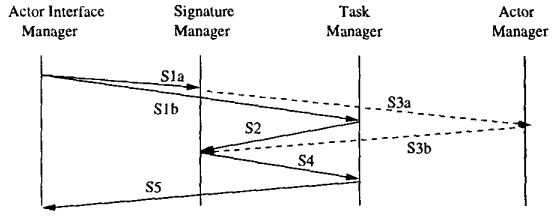

Figure 3. Signing a Signature

from the Actor Interface through Actor Interface Manager. The actors sign the forms with their private keys. These signed forms are then sent to SM directly, i.e. Step S1a. A request for signature is also sent to the Task Manager, i.e. Step S1b. This request includes the information of the task identity, the form identity and the actor identity. The Task Manager then validates whether the actor requesting for signature service is the actor assigned to handle the task, and record the state of the task as "signature requested". The result of this validation together with the identities are then sent to the SM in Step S2.

After receiving the validation results from Task Manager, if the actor is the assigned actor for executing the task, the SM validates the signature with the actor identity. The SM may requires more information about the actor such as whether the actor is eligible to request for signing the form. The SM will then request for these information from the Actor Manager in Step S3a and the results are sent back to the SM in Step S3b. If all criteria for signing the form are valid, the SM will carry out the procedures according to the purposes of the signature stated in *purposeList*. These procedures are described in Section 5.

After all the procedures associated with the purposes of the signature has completed and are succeeded, the SM will sign over the form with its private key. The SM passes the signed records to the Task Manager in Step S4. The Task Manager will record the completion of the signing process, changes the state of the task to "signature completed". Then it will send the signature to the Actor Interface Manager which request for the signature in Step S5. The signing process is then completed.

In Step S2 and S3, if any one of the validations failed, the SM will raise an exception with the type of exception. Corresponding exception handling procedures will be invoked according to the workflow specifications.

4.2.2 General Procedures of Verifying Signatures

The verification of a signature involves five major steps which are shown in Figure 4. When an actor wants to verify a signature on a form, it will request through the Actor Interface Manager. The form involved is sent to the SM in Step V1a. The Task Manager is also informed that a request for signature verification has invoked in Step V1b. The Task Manager will change the state of the task to "Signature Verification Requested".

After receiving a form for signature verification, the SM decrypt the form with its public key. Then the signature of the signer(s) and information about the signature, including identity of the claimed signer(s), can be obtained. The SM then obtain the signer and key identities from the signature_content. The SM

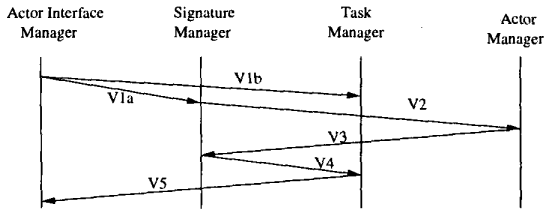

Figure 4. Verifying a Signature

can verify the signature of the claimed signer with its public key. If the signature is valid, the SM sends the signer information, the task identity and the signing time to the Actor Manager in Step V2. The Actor Manager then checks whether the actor is responsible for the task at the specified time. The result is sent to the SM in Step V3.

If the Actor Manager returns a valid message, the SM will carry out procedures according to purposes in the *purposeList*. After successful completion of these procedures, result of the signature verification will be sent to the Task Manager in Step V4. The Task Manager will record the completion of the signature verification and passes the result to the Actor Interface Manager in Step V5.

In Step V2 and V3, if the signature is invalid or the actor information do not match the information carried in the signature, exceptions will be raised by the SM. Corresponding exception handling procedures will be invoked according to the workflow specifications.

5 Realizing Different Signature Purposes

In this section, we discuss the procedures of handling signatures of different purposes with the extended *Liaison* Workflow Engine. These discussions are refinements of the steps in signing and verification of signatures (Section 4.2).

Authentication For a signature of authentication purpose, SM has to check whether the actor is eligible to be the signer of the form. This includes whether the signer is the actor assigned to handle the task and the signer's position is eligible to sign the form. These information are obtained from the Task Manager and the Actor Manager in Step S2 and S3.

To verify a signature, SM will check actor assignment information with the Actor Manager in Step V2 to ensure that the actor was assigned to handle the task at the time of signing.

Authorization When a signature is used for authorization, some tasks or resources will be involved. These tasks or resources are stated in the associations of the *signature* data. In the Step S2 of the signing process, information about the right of the signer of invoking the tasks or accessing to these resources is obtained from the Actor Manager. If the signer is the assigned actor to carry out the task and has the right to invoke the tasks or access the resources, the signature will be added to the corresponding eligibility list by the SM.

When restricted tasks or resources are being invoked or accessed by a task, The signature in the eligibility lists will be looked up and the signer will be authenticated with the procedures of authentication.

Proxy Signature When an actor delegates a right to another actor, it requests a signature on a proxy form in Step S1. In this proxy form, besides the description of the right to be delegated, it should also include the identity of the actor and the identity of the proxy, i.e. the actor which is delegated the right. In Step S2, the SM should check whether the actor which create the proxy form is eligible to process the form which it delegate the right of processing to the proxy actor. These information is obtained from the Actor Manager through Step S3.

In verifying the proxy signature, the SM should verify the identity of the actor which exercise the proxy form against the proxy identity as stated in the proxy form. Furthermore, it should also check whether the actor which delegates the right to the proxy actor is eligible to process the form. These information were obtained through Step V2.

Time Stamping When a signature is used as a time stamp, in addition to the attributes to be signed, the date and time of signing the signature should be included in the signature. Hence when verifying the signature for time stamp purposes, the date and time in the signature should be extracted and compared with the date and time in the form contents after Step V3.

Accountability When a signature is used as accountability, it means that some actors are responsible for the consequence of the form. In signing a form for the purpose of accountability, in case of weak accountability, the SM only requires to check if the actor is a valid actor of the organisation in Step S3. In case of strong accountability, the SM also requires to check if the actor is eligible to be accountable for the form in Step S3.

In Step V3 of verifying a signature, if a signature is used for weak accountability, the SM only need to ensure that the signature is signed by a valid actor of the organisation. In case of strong accountability, the SM will also requires to ensure that the signer is eligible to be accountable for the task at the time of signing the form.

Being Informed This kind of signature means that the signers is notified about the details of the form. Furthermore, only the eligible actors should be informed of the information carried in the form. This is checked in Step S3.

Witness The purpose of witness in paper-based workflow is to provide non-repudiation support. When the forms are signed by public key cryptosystem together with the signature of SM signing over the form, non-repudiation services is provided. Hence, in normal situation, the purpose of witness is no longer required.

Sequential Multiple Signatures In handling sequential multiple signature, signing and validation of signatures, for most of the situations, are going hand-in-hand. Before signing a signature, the SM has to validate whether the previous actors are eligible actors to sign the form. These include whether the actors who signed the form are

- the right actors who are assigned to process the tasks and
- are eligible to process the tasks.

These information are obtained through Step S3. If any of the signatures are invalid, this means that the form has been processed by some invalid actors. Information may have disclosed to some inappropriate actors or some inappropriate processing might have been carried out. Hence exceptions should be raised.

In case of dependent sequential multiple signature, the sequence of the actors signing the form has to be checked with the content of the *signature*. If the sequence of signing the form is incorrect, the signing processing should be rolled back and re-do the signing again.

According to the dependency requirement, signatures can be signed differently.

Independent Sequential Multiple Signature In the independent sequential multiple signature case, since the sequence of signing are immaterial, the signers can simply sign on the context of the form without signing on the signatures of the previous signers in Step S1.

In the validation of this case, it only has to check

1. the validity of each of the signatures and
2. all the form contexts obtained from the digital signatures are consistent.

Dependent Sequential Multiple Signature In case of dependent sequential multiple signature, the sequence of signing the forms is important. This can be solved easily by applying the Signature On Signature mechanism in Step S1. This means that the signers has to sign only on the signature of the previous signers to form the new digital signature. More details of Signature On Signature are discussed in Section 6.

In the verification of the signature, from the *signature_content*, the identity of the previous signer and key_identity can be obtained. Then the signature can be verified by decrypting the signature with the public key of the previous signer. After decrypting the signature, the form content and the signature of the "signer before this signer" will be obtained. Then the signature of this signer can be validated in a similar manner. This verification process is repeated until all the signatures are verified. Consequently, Step V2 would be required to be repeated accordingly.

Parallel Multiple Signature In parallel multiple signature, the information about the potential signers of are kept in the *signature*.

The signing process of parallel multiple signature is handled as simple signing process because there are multiple copies of the same form and each signature is signed over the form individually.

In case of join, i.e. in the verification process, according to the information in *JoinPolicy*, join-all or join-some can be determined easily. The SM has to validate all the signatures over each of the form one by one. If there are any invalid signatures, the validation is regarded as invalid. Whether exceptions are raised when there are invalid signatures will depend on the purposes of the multiple signature. These purposes can be classified into **mandatory** and **voting** type.

Mandatory In mandatory type, all signatures are required in the workflow. Any invalid signature will invalidate the process. Exceptions should then be raised for further action.

Voting In voting process, each actor signs its copy of form. The signing process is handled as a simple signature purposes. In the validation process, since invalid signatures will only invalidate the votes, invalid signatures will not cause exceptions. Two counters have to be used for counting the valid and invalid votes. Furthermore, there should be a field in the form for the SM to indicate whether the form is a valid vote. The validated votes, together with the counting, are then returned to the task requesting for signature validation. through the Task Manager in Step V4 and V5.

Anonymous The value of the *anonymity* field is *anonymous* means that the identities of the signers in the group cannot be disclosed. Anonymity is easier to be implemented in electronic workflow systems than in paper workflow systems. There are standard cryptographic techniques to implement voting schemes [11]. SM only needs to verify the signatures in the usual way. Afterwards, the SM only requires to send the verification result, which is either valid or invalid, to the corresponding tasks without providing any information about the actors which sign the form in Step V4 and V5.

If the value of the *anonymity* is *open*, this means that the identities of the signers are to be disclosed. No extra work is required.

6 Discussion

Some interesting issues in our data structure designed for handling different signatures purposes are discussed in this section.

Workflow Signature as Fields The objectives of most digital signatures are data integrity, authentication, non-repudiation. The signatures usually are consisted of two parts, the original message and ciphertext obtained by encrypting the original message. Authentication services is provided by the property of public key cryptography. Data integrity services is provided by comparing the original message with the plaintext obtained by decrypting the ciphertext. By signing and archiving by some trusted third parties, non-repudiation is achieved.

In our design of workflow signature, since forms are used in workflow, signatures need *only* consists of the ciphertext. Furthermore signatures can *naturally* be treated as fields in the form, just like what it is doing in paper-based workflow systems. Even in case of independent multiple signatures, each signer need only sign the contents of the original form and put their signatures in the *signatureContentList*. By making use of public key cryptosystem, authentication services is obtained. Data integrity services is provided by decrypting the content of the *workflow_signature* and compare it with the contents of the form. Non-repudiation services is provided by the SM and the Task Manager.

Signature On Signature In paper-based workflow, dependent multiple signatures have to be enforced by rules. So the sequence of signing the signatures can be broken easily. On the other hand, many digital signature schemes are designed with only one

signature in mind. There are two main approaches in handling multiple signatures. The first approach is to use complicate mathematical schemes, such as the modified RSA scheme [11] to handle multiple signatures. These schemes are usually very complicate in structure, and its application is not easily extendible to solve related problems. Therefore it is relatively difficult to incorporate them into existing workflow systems. The second approach is to apply single digital signatures more than once with some predefined sequences. This approach can be demonstrated by the previous example in Sequential Multiple Signature.

1. Amy, the project manager, signs the Production System Testing Form (PST Form) with her *private key*. The workflow signature, i.e. the ciphertext, is then put inside the *signatureContentList*. This signed form is then sent to the Bob, the production manager.
2. After Bob receiving the signed PST Form, Bob can verify the workflow signature with Ann's public key. If he agrees with Amy, he can then sign on the workflow signature of Amy and then put his signature in the *signatureContentList*.

This approach has the advantage that no extra mathematical schemes are used. All cryptographic functions used are identical to those used in single signature schemes. Furthermore, in multiple signatures, only one copy of the original form is kept. As a result, this approach is more favorable to be used to incorporate multiple signatures in an electronic workflow system.

Note that by migrating multiple signature from a paper-based workflow system to an electronic workflow system, extra functionalities are created. In the above example, if it is a paper-based workflow system, there is no way to ensure that Bob signs the form *after* Amy has signed it. The appearance of the signatures on the paper form will be identical regardless of the signing sequence. However, in electronic workflow systems, we can ensure the signing sequence. We call this mechanism Signature On Signature.

Although Signature On Signature is a function not available in paper-based systems, including this function does not require much extra work in the design of the SM. In supporting Signature On Signature, information including mode of the signature and sequence of the signature are required. The extension of data structure in the Information Model of *Liaison*, discussed in Section 3 are specially designed to handle this approach.

7 Conclusion

As the use of open networking facilities such as the Internet is becoming more and more popular, there is an increasing demand to change paper-based workflow systems to electronic workflow systems. There is an urgent need to understand the problems of implementing digital signatures in electronic workflow systems, and to provide solutions for the problem of using signatures for different purposes. In papers [10] and [9], different purposes of signature are discussed.

The main contribution of this paper is to demonstrate how to realize the different purposes of signatures in a workflow architecture. In this paper, the the *Liaison* Workflow Model [2, 4] is used as example. The *Liaison* Workflow Engine [8] is extended to incorporate a Signature Manager which provides signature services. Also, how to design extra data structures to handle signatures are discussed.

By empowering the workflow system with signature handling facilities, we moved a step forward to achieve the goal of managing sophisticated software development processes by a distributed workflow environment.

References

[1] Daniel K.C. Chan and Karl R.P.H. Leung. A Workflow Vista of the Software Process. In *Proc of the 8th International Workshop on database and Expert Systems Applications (DEXA)*, pages 62–67, Toulouse, France, September 1997. IEEE Computer Society Press.

[2] Daniel K.C. Chan and Karl R.P.H. Leung. Software Development as a Workflow Process. In *Proc of the Joint 1997 Asia-Pacific Software Engineering Conference and International Computer Science Conference (Joint 4th APSEC'97 & ICSC'97)*, pages 282–291, Hong Kong SAR, China, December 1997. IEEE Computer Society Press.

[3] Daniel K.C. Chan and Karl R.P.H. Leung. Valmont: a Language for Workflow Programming. In *Proc of the 31st Hawaii International Conference on System Science (HICSS'98)*, volume Vol 7, pages 744 – 753, Hawaii, USA, January 1998. IEEE Computer Society Press.

[4] D.K.C. Chan, K.P.H. Leung, C.Y. Ying, and K.C.C. Chan. Liaison: A Workflow Model for Novel Applications. In *Proc of the Fifth Asia-Pacific Software Engineering Conference (APSEC'98)*, pages 144–153, Taipei, Taiwan, R.O.C., December 1998. IEEE Computer Society Press.

[5] Warwick Ford and Michael S. Baum. *Secure Electronic Commerce*. Prentice-Hall International, 1997. ISBN 0-13-476342-4.

[6] Lucas C.K. Hui and Karl R.P.H. Leung. The Need of Signatures Handling in Electronic Workflow Systems. In *to appear in the Proc. of the 5th International Conference on Information Systems Analysis and Synthesis (ISAS'99)*, Orlando, Florida, USA, July – August 1999.

[7] Peter Lawrence, editor. *Workflow Handbook 1997*. John Wiley & Sons, Baffins Lane, Chichester, West sussex PO19 1UD, England, 1997. ISBN 0-471-96947-8.

[8] Karl R.P.H. Leung and Jojo M.L. Chung. Liaison Workflow Engine Architecture. In *Proc of the 32nd Hawaii International Conference on System Science (HICSS'99)*, Hawaii, USA, January 1999. IEEE Computer Society Press. Published in CD-ROM, ISBN 0-7695-0001-3.

[9] Karl R.P.H. Leung and Lucas C.K. Hui. Multiple Signature Handling in Workflow Systems. 1999. in preparation.

[10] Karl R.P.H. Leung and Lucas C.K. Hui. Signature Management in Workflow Systems. In *to appear in the Proc. of the 23rd Annual International Computer Software and Applications Conference (COMPSAC'99)*, Phoenix, Arizona, USA, 27-27, October 1999.

[11] B. Schneier. *Applied Cryptography*. Wiley, New York, 2 ed edition, 1995.