A Multi Purpose Web-based Contractual Management System

SHARIL TUMIN
University of Bergen
IT Dept
P.O. Box 7800, 5020 Bergen
NORWAY
edpst@it.uib.no

SYLVIA ENCHEVA
Stord/Haugesund University College
Faculty of Technology, Business and Maritime Sciences
Bjørnsonsg. 45, 5528 Haugesund
NORWAY
sbe@hsh.no

Abstract: A Web-based multi purpose contractual management system can provide support to contractual process workflows for different types of contractual process of unilateral, bilateral or multilateral contracts. These processes can be done on-line i.e. Web-based without the need for all actors to be synchronously present with respect to both time and space. Different contractual processes of initialization, negotiation, agreement, signing (witness), and archive are managed within the application securely by analyzing data-flow between actors, ushering actors to perform their duties in a timely manner and employing appropriate cryptographic techniques on every step of the way. The implementation must deliver a management system that provides operational properties of authenticity, privacy, trustworthy, reliability, verifiability, and linkability.

Key–Words: Contract, Contract Management, Applications security, Secure Web-based Application, Secure Workflow Modelling, Applied Cryptography

1 Introduction

Since early time in the history of mankind, men negotiated, agreed and bound to contractual agreements in the matter of properties exchange or promises of future services.

Figure 1: Kushans contract for a sale of land 527 A.D. (front side)

In a collection of Bactrian document from ancient Afghanistan [16] is a piece of Kushan's contractual document, shown in Figure 1, for a sale of land, written down as early as 527 A.D.

One can clearly see that the document was witnessed and sealed by no less then five persons. A copy of the original text was written, tightly rolled, tied with string, and sealed with five bullae on the top of the seen document. The back of the sale document, shown in Figure 2, shows signatures of the witnesses to the respective seals. Presumably, the sealed copy could then be opened in the presence of a judge in case of a future dispute.

Figure 2: Kushans contract for a sale of land 527 A.D. (back side)

Generally speaking, a contract is a legally binding agreement between two or more parties by which rights are acquired by one or more to act or forbearance on the part of the other or others. A person can act on the behalves of a group of people, organization,

cooperation or enterprise. A contract is to be agreed upon. In case of disputes at a later time, a record of the agreement is to be kept and can be used as evident for resolving conflicts. So, a contract is finalized when the content of the contract is agreed and the record of the agreement is accepted trustfully by all concerning parties. Any framework for contract management, digital or otherwise is to promote this trust.

The goal of this paper is to propose an open-ended, multi purpose Web-based Contract Management System (**CMS**) for deployment where contractual processes can be done between actors using Web-based applications as assertive actions and emails as event triggers. Any registered user can initiate a contract procedure. Anyone with an email account anywhere can register to the system. Any of the registered users can be used as a witness. This proposed framework for on-line contract management will provide the support of contract process flow without the need for the different related actors within the process to be synchronously present with respect to both time and space. Virtual meetings are done by message passing via email or Web-based drop-box areas and actions are executed or triggered for assertion by applications in a timely fashion dictated by rule based process flow directives.

A contract can be of a different type depending on the number of participating parties. There can be one, two, three or more interested parties, Figure 3.

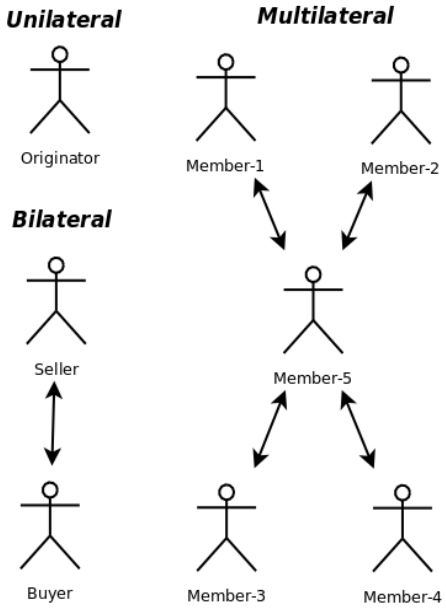

Figure 3: Contract Types

Therefore a contractual process can be typed as the following three types:

1. ***unilateral*** - e.g. a will, a testament,
2. ***bilateral*** - e.g. a buying and selling agreement,
3. ***multilateral*** - e.g. a project team distribution agreement of roles, responsibilities, and resource divisions.

Different types of contractual process define different types objects and attributes, both contract specific and domain specific, and these are *contract-core objects* and *contract-core attributes*, and related to these contract-core are *domain-specific objects* and *domain-specific attributes*. For a particular contract, these objects and attributes together with contract meta data of hashes and digital signatures will be used to produce a finalized contract document in what we call a *contract synopsis*. A *contract synopsis* is represented in XML (Extensible Markup Language).

Different types of contract will have different discreet stages. Stages are arranged in an ordered sequence. The system will track these sequences and inform all concerned parties about the status of a current stage. The currently active actors at a particular stage will be ushered to perform certain imperative actions using Web-based applications.

Most contracts will pass through the stages in the following sequence:

S1 - *Initialization*

S2 - *Negotiation*

S3 - *Agreement*

S4 - *Signing* (witnessing)

S5 - *Archive* (after sealing)

Depending on which type of contractual process shown in Figure 3, the stages of the process will follow the sequence shown in Figure 4.

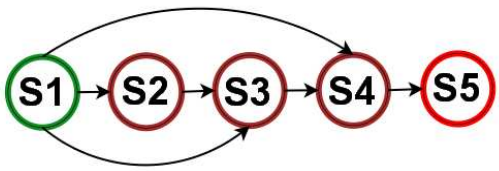

Figure 4: Contractual process workflows

The *unilateral* contractual processes will skip **S2** and **S3**, and will effectively have workflow sequence {**S1**, **S4**, **S5**}. For fixed priced buying and selling, in the *bilateral* contractual processes **S2** will be skipped and will have workflow sequence {**S1**, **S3**, **S4**, **S5**}. The *multilateral* contractual processes are more complex and will have all workflow sequence {**S1**, **S2**, **S3**, **S4**, **S5**}. It can also happen that there are loop-backs from **S3** to **S2**, and will effectively have workflow sequence {**S1**, {**S2**, **S3**}+, **S4**, **S5**}, in such cases.

The contract management system needs to facilitate users in 1) initialization phase (*Initialization*); 2) intermediate phase (*Negotiation* and *Agreement*); and 3) finalize phase (*Signing* and *Archive*).

In the case of *unilateral* contract, the system will provide similar services as given by a notary public officer. The *Negotiation* and *Agreement* are not needed for unilateral contract. One service that is not connected directly to wills or testaments but is similar in its processes is related to notarial documents (signed and witnessed notices) of any kind. This type of contracts are useful for attaching timestamps and witnesses on any type of documents, f.ex. a scientific paper. Both *bilateral* and *multilateral* contracts will have to go through *Negotiation* and *Agreement*.

Essentially, actors in the contractual procedure are having different roles with different responsibilities and duties. These roles are; 1) initiator, 2) provider, 3) consumer, 4) witness and 5) archiver. These actors are actively involved in the different stages mentioned above. We will discussed these actors with their corresponding roles and duties in more detail in Suction 2.1.

The design, development and deployment of a Web-based multi purpose contractual management system (**W/CMS**) must deliver a management system that provides operational properties of:

1. **P1** - *Authentication*
2. **P2** - *Privacy*
3. **P3** - *Trustworthy*
4. **P4** - *Reliable*
5. **P5** - *Verifiable*

These operational properties will be discussed in some detail in Suction 2.2. For the time being it is enough to say that these operational properties define constraints on actors of the system to uphold the security level needed for a contractual management system.

2 Background

2.1 Actors' Roles and Responsibilities

There are five types of human actor identified in the **W/CMS** directly related to the roles they assumed in the system. They are labeled as follows:

$\mathbb{I}$ - *Initiator*, which also will act as intermediary, coordinator, and overseer.

$\mathbb{P}$ - *Provider*, for example seller and benefactor.

$\mathbb{C}$ - *Consumer*, for example client, buyer and beneficiary.

$\mathbb{W}$ - *Witness*, a witness will put signature on a contract.

$\mathbb{A}$ - *Archiver*, an archiver will put a seal on a contract in the fine stage of a contractual process.

Figure 5 shows relationships of actors, objects, actions and states. Actors are collected into three types of groups, 1) system operators group $\mathbb{U}^M$; 2) active users group $\mathbb{U}^A$; and 3) passive users group $\mathbb{U}^P$. An actor belongs to a set of users of the system $\mathbb{U}$.

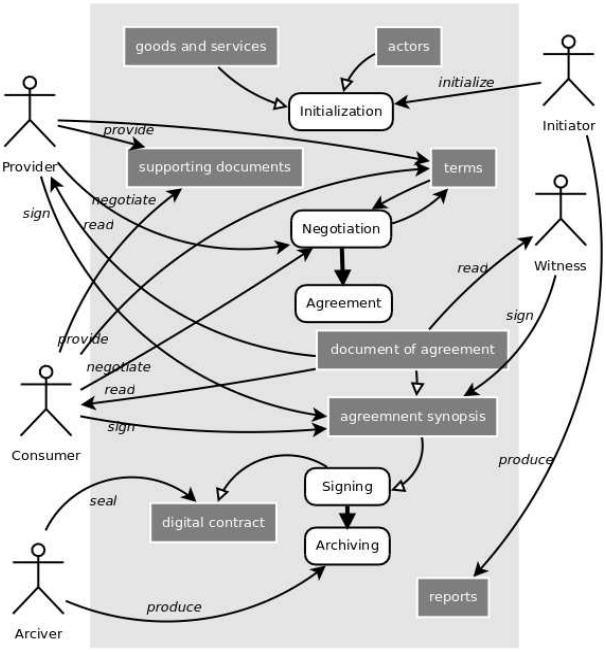

Figure 5: Conceptual CMS use-case

In order to be a user, a person needs to be registered to the **W/CMS**. Each user is given a unique

Table 1: Contract type vs Number of Witness

Contract type	$\mathbb{P}$ or $\mathbb{C}$	$\mathbb{W}$(min)	$\mathbb{W}$(max)
unilateral	1	1	2
bilateral	2	2	2
multilateral	≥ 3	0	0

username and a password during the registration process. The username and password will be used for authentication during the subsequent interactions with the system.

Each user in the system will be given a pair of keys, one public K_{pub} and the other private K_{prv}. The **W/CMS** will utilize the public-key cryptography for the process of; 1) encryption/decryption; 2) sign/verify; and 3) blind/unblind, more on this in Section 3.1.

Both *initiators* $\mathbb{I}$ and *archivers* $\mathbb{A}$ belong to the **W/CMS** managers groups, $\mathbb{I} \cup \mathbb{A} = \mathbb{U}^M$. They manage the human side of the **W/CMS** and do all manual processes concurrent to the automatic processes defined in the system.

An *initiator* $I_i \in \mathbb{I}, i \in [1, n]$ is naturally a trusted operator in the **W/CMS**. Her work is to initialize all pertinence *contract-core objects*, *domain-specific objects* and their related attributes. The initiator is prompted into action by a request done by an active user of the system.

Within an extend of a contract forming process, a user is either active or passive, and can never be both at the same time, $\mathbb{U}^A \cap \mathbb{U}^P = \emptyset$. Both *providers* $\mathbb{P}$ and *consumers* $\mathbb{C}$ belong to the active user group. While *witnesses* $\mathbb{W}$ belong to the passive user group, $\mathbb{P} \cup \mathbb{C} \cup \mathbb{W} \subseteq \mathbb{U}$. Within a contract forming instance, a provider or a consumer can not also be a witness. All active users $P_i \in \mathbb{P}, i \in [1, m]$ need to provide the **W/CMS** with supporting documents, conduct negotiations and actively agree to the final agreement of the contract. Potential *witnesses*, $W_k \in \mathbb{U}, k \in [1, l], \mathbb{W}_k \neq \mathbb{P}_j$ will be invited to be witnesses and sign a finalized agreement. A witness will sign a particular contract blindly or otherwise, depending on the privacy constraint of the contract. We propose a witness policy plan as shown in Table 1.

For *multilateral* contract, there is no need of an external witnesses, where the active members of the contract themselves serve as witnesses. In this proposal, we make the assumption that the witnessing process incur significant cost to the contract formation processes, in terms of risks, economy and efforts.

2.2 Operational Properties

As mentioned in passing in Section 1, in order to achieve security and quality criteria, thus promote users' trust toward the system, the proposed **W/CMS** must perform with the following operational properties:

P1 - *Authenticity*: Only registered users $U_i \in \mathbb{U}$ are recognized by the system. Users are authenticated prior to granting access to the system. Only eligible users with authorization corresponding to theirs roles are able to access certain parts of the system. Users activities are logged.

P2 - *Privacy*: Access to each *contract synopsis* is controlled by graded privacy level of 1) *public*, 2) *semi-public*, 3) *semi-private*, and 4) *private*. At a *public* level, some or all components of a particular *contract synopsis* are accessible by anyone interested. At a *semi-public* level, some or all components are accessible only by users of the system. At a *semi-private* level, some or all components are accessible to $\mathbb{P} \cup \mathbb{C} \cup \mathbb{W}$. Lastly, at a *private* level, only $\mathbb{P} \cup \mathbb{C}$ are granted access to all components of a particular *contract synopsis*. Each component can be individually marked as either *public* or *private*. The **W/CMS** provides many different combinations of access controls. When there are conflicts in authorization, the *contract synopsis* level takes precedent over the individual components access flags.

P3 - *Trustworthy*: All user, $\mathbb{U}$, are real people and no two or more users belong to a single person. All contracts documents marked with *private* flags are stored in an encrypted forms and only authorized users can access and decrypt them. A detailed history of a particular *contract synopsis* at any stages is given to $\mathbb{U}^A$ at any time. The invariant $\mathbb{U}^A \cap \mathbb{U}^P = \emptyset$ is diligently checked by $\mathbb{A}$ for each $\mathbb{W}$ committed to a particular *contract synopsis*.

P4 - *Reliability*: It is not possible to fake signature on a contract. Each private key K_{prv} is protected by user's password whereby only the owner can make use of them. Private keys are stored securely in the system and no one can misuse them, even persons with system managers permissions, $U_i \in \mathbb{U}^M$. Only valid witness $W_k \in \mathbb{U}, k \in [1, l], \mathbb{W}_k \neq \mathbb{P}_j$ for a particular contract is allowed to sign as a witness.

P5 - *Verifiability*: Individual person involved with a particular contract is able to independently verify any signature within the contact of the contract.

All activities pertinence to the contract forming process were log, every piece of items will be connected with timestamped events in the *events history* of a particular contract. At the closing phase of the contract both the *contract synopsis* and *events history* will be signed and sealed by the archiver 𝔸 and stored in a save storage area or an off-line media for example an encrypted CD (compact disk).

2.3 Design Principles

Effective design connects acting to thinking which in turn connects implementation to formulation. To be effective a design needs to be guided by some design principles. These principles provide the designers and the implementers with a base line on which the product as a whole is aiming to achieve when it is put into production. Thus the system that facilitate the contract management, must above all, be *usable*, *adaptable*, and *manageable*. The system users interfaces and process workflows must conform to users needs and expectations, logically laid out and simple to understand. There should not be any ambiguities. Regulations regarding different aspects of contract making will change with time, so the system needs to adapt to the current regulatory demands. To be agile and responding to changes, the system and its subsystems need to be managed with manageable interfaces. Therefore, in this paper we propose an open-ended Web-based contract management system guided by these five design principles:

Q1 - *Usability*

Q2 - *Adaptability*

Q3 - *Scalability*

Q4 - *Interoperability*

Q5 - *Maintainability*

It goes without saying that these five design principles must include and implement the five operational properties mentioned earlier in Section 2.2.

From the conceptual use-case diagram of Figure 5 and the three phases mentioned in Section 1, the working model of the **W/CMS** can be decomposed into three separate subsystems as shown in Figure 6, which represents a simplified version of process workflows schematic of the whole system.

The design principles will help developers, installers and maintainers respectively to develop, deploy and maintain the propose **W/CMS**. The success of any Web-based system, very much depending on how closely these design principles are

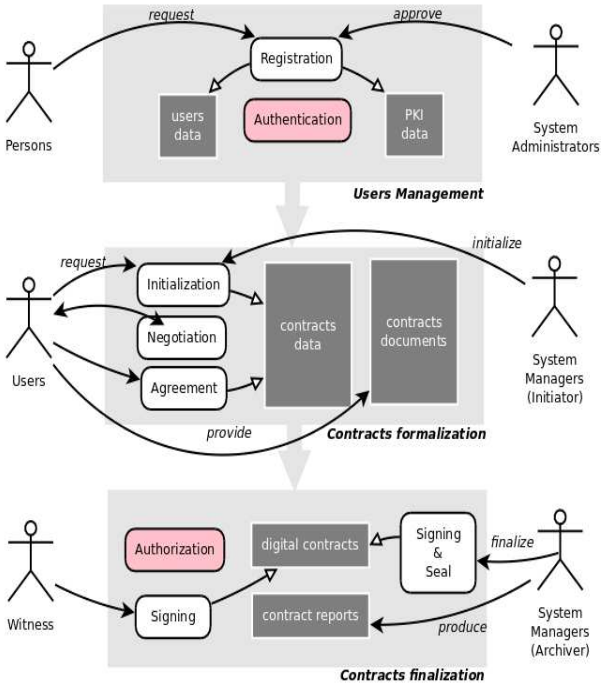

Figure 6: **W/CMS** Subsystems use-case

followed under modelling and design phase of the development. Here we provide a short summary of issues which we meant to be pertinence to the five design principles:

Q1 - *Usability*: Users are the primary reason why a system ever get built, deployed and maintained. Any system under consideration must above all put users requirements as its prime motive for existence. In a Web-based application, users to the system will interact with it using Web browsers. The look and feel of Web pages can be coded using HTML (HyperText Markup Language), CSS (Cascading Style Sheets) and JavaScript. It is a fact that different Web browsers interpret all HTML, CSS and JavaScript codes differently. To be useful Web pages must behave consistently regardless of what type Web browsers are being used. Another important issue related to *usability* is error and exception management. Both users and system errors should be handled properly with end users in mind. All fatal and non-fatal failures due to database, inter-communication, and system exceptions should be traps and handled meaningfully.

Q2 - *Adaptability*: Legislature changes all the time. The same is true to technology. The system

in production, often needs to be redeveloped and redeployed in response to necessary changes. There are choices to be made from the very beginning to which tools and architectural models on which the production system will be based on, in order to support *adaptability* principle. Subsystems are made to be open-ended and in conformation to widely used standards. To achieve a high level of adaptability, the system should be modular both in development and deployment, by employing independent and loosely couple operational units.

Q3 - *Scalability*: As the system become widely known and has to cater to more users than the initial stage, the scalability will become an issue that need to be addresses. How well the system can be scaled up is very much dependent on the underlying architecture. For instance, multiple servers can be deployed to support increasing demands. If a system can be decomposed into independent subsystems, as the **W/CMS** can be logically and practically split into three parts as shown in Figure 6, then three different servers can be employed to deploy the collective functionalities of the whole system. Certain functions can also be served from multiple servers, when a completely centralized control is not needed during day to day operations. Late data synchronisations can be done as a part of standard routines.

Q4 - *Interoperability*: A system does not exist on an island, entire of itself; it needs to co-exist with its operational environment. It needs to accommodate existing systems and future systems within its parent organization and any collaborating organizations. Its ecosystem could be far reaching and was not perceived at initial instalment, however the success of survival can very well depend on how well it is able to communicate with other independent systems. Most often, the possibilities of introducing middle-ware is the only practical way to tackle the problem of *interoperability*. In order to do this developer-maintainer needs the ability and interest to understand externals systems. Standardization can help a lot.

Q5 - *Maintainability*: Web pages and forms, operational codes and codes library, and data model need to be maintained in response to all the demands incurred by all the above. Choice of tools and technologies are important. Object oriented programming techniques and open source technologies will provide maintainer of a system with a better level of *maintainability* than unstructured coding and legacy technologies. Simple design based on well understood and stable framework will greatly contribute to system *maintainability*. To reduce operational surprises due to bugs, development and testing processes must be done in parallel. Whenever possible a sandbox environment to the production system should be provided.

2.4 Contract Making Process Flow

Different human actors and their interactions with the system are clearly labelled in Figure 6. The three main subsystems are:

1. *Users Management*
2. *Contracts Formalization*
3. *Contracts Finalization*

They are modeled to be distributed, cooperative and loosely coupled independent subsystems. To promote higher security level, these subsystems are designed to be operational and administrative independent of each other. System management responsibilities will be given to three non-collaborative teams. These managers will not share administrative secrets and systems' data.

The *users management* subsystem manages user registration. A new user will be given a unique username and a self-chosen password, and a pair of RSA public and private keys. The private key is protected by the user password using a symmetric-key encryption. The meta data and private key are also symmetric-key encrypted by the subsystem and send over to the user for safe keeping. The user needs to provide this piece of encrypted text block whenever she changes her password later on.

The *contracts formalization* subsystem is the bulk of **W/CMS**. Each contract is initialized when the request is approved by system managers. Each contract will be given a standardized individual database. All contract data and contract documents will be stored in the database. The negotiation will be conducted among persons related to the contract in an environment similar to a blog. All events are logged to this blog by the system software, system managers, coordinator and users. The production of the *contract synopsis* is done when an agreement is reached by all parties. All parties will be asked to sign the *contract synopsis*.

The *contracts finalization* subsystem constitutes the finalizing processes for the closing phase of a contract. Witnesses will be called by the archiver to verify the *contract synopsis* and all participating parties' signatures. The witnesses will then put their signatures

on to the *contract synopsis*. The archiver will then verify the *contract synopsis*, all participating parties' signatures and all witnesses' signatures, before closing the blog and sealing the contract by putting her signature to the whole *contract synopsis* and signatures. The signing hierarchy is shown in Figure 7. The sealed contract package and the blog are encrypted and then securely stored.

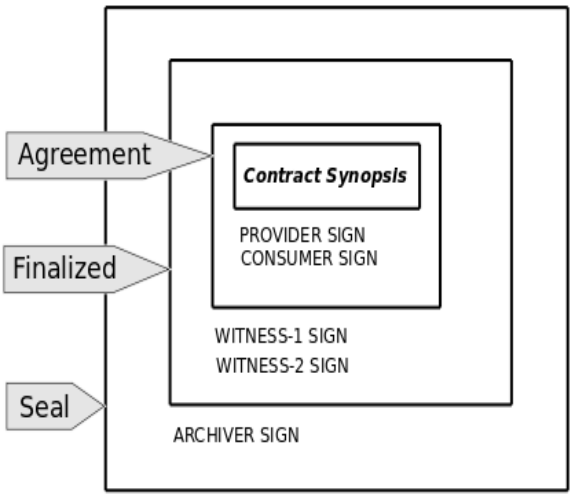

Figure 7: Contract Signatures Hierarchy

3 Supporting Tools

3.1 Cryptographic Tools

The proposed system employees three simple, well known and widely used cryptographic tools. They are:

1. symmetric-key encryption implemented in Blowfish [13],
2. public-key encryption implemented in RSA [4] public-key cryptography,
3. cryptographic hash functions implemented by SHA [3] hash algorithms.

Blowfish symmetric-key encryption uses a single secret key. This key is used for both encryption and decryption.

Blowfish Encryption scheme:-
Encryption β_{enc}:
ciphertext, $\mathbf{c} = \beta_{enc}(K_{sec}, \mathbf{m})$
Decryption β_{dec}:
plaintext, $\mathbf{m} = \beta_{dec}(K_{sec}, \mathbf{c})$
Inverse transformation:
$\mathbf{m} = \beta_{dec}(K_{sec}, (\beta_{enc}(K_{sec}, \mathbf{m})))$

Symmetric-key cryptography is not practical for secure communication between many persons due to key distribution and key management problems. For private communications between two persons, one secret key is enough, but for more than two, say four persons, six keys are needed and everyone must have three keys each. In fact for n persons, the total number of keys needed are $n!/2 \times (n-2)!$ and each need to keep $n-1$ keys.

The utilization of public-key cryptography will solve these problems since each person needs only to have a pair of keys, namely the public and private keys. The public key K_{pub} is readable by all interested parties while the private key K_{prv} is kept secret and only known to the owner.

A public-key cryptographic system depends heavily on computational complexity theory and number theory. RSA is the most well known and widely used cryptographic system in today's digital world. RSA supports non-symmetric-key cryptographic schemes for 1) encryption/decryption; 2) sign/verify; and 3) blind/unblind.

RSA Encryption scheme:-
Encryption - ξ_{enc}
ciphertext, $\mathbf{c} = \xi_{enc}(K_{pub}, \mathbf{m})$
Decryption - ξ_{dec}
plaintext, $\mathbf{m} = \xi_{dec}(K_{prv}, \mathbf{c})$
Inverse transformation:
$\mathbf{m} = \xi_{dec}(K_{prv}, (\xi_{enc}(K_{pub}, \mathbf{m})))$

RSA Signature scheme:-
Signing - ζ_{sig}
signature, $\mathbf{s} = \zeta_{sig}(K_{prv}, \mathbf{m})$
Verification - ζ_{ver}
verify, $\mathbf{v} = \zeta_{ver}(K_{pub}, \mathbf{s})$
Inverse transformation:
$\mathbf{m} = \zeta_{ver}(K_{pub},(\zeta_{sig}(K_{prv}, \mathbf{m})))$

The blinding factor ψ is only known to the message owner. Blind signature is useful when anonymity is important [2]. Using blind/unblind, a signee can sign the message without knowing what it contains.

RSA Blind Signature scheme:-
Blind - λ_{blind}
blind, $\mathbf{b} = \lambda_{blind}(K_{pub}, \psi, \mathbf{m})$
Signing - ζ_{sig}
signature, $\mathbf{bs} = \zeta_{sig}(K_{prv}, \mathbf{b})$

Unblind - $\lambda_{unblind}$
unblind, $\mathbf{s} = \lambda_{unblind}(K_{pub}, \psi, \mathbf{bs})$
Verification - ζ_{ver}
verify, $\mathbf{v} = \zeta_{ver}(K_{pub}, \mathbf{s})$

A more detailed discussion on Public-Key Cryptography Standards (PKCS) #1 v2.1 by RSA Laboratories is presented in RFC3447 [4]. Other intersting works in that area are shown in [5], [6], [7], [11], [17], [18], and [19]. Their application in systems supporting eGoverment models is presented in [8], [9], and [15].

3.2 Software Tools

The system can be implemented as a multi-tiers Web-based application, built using free and open source software. The users using their Web browsers will interact with an Apache [1] Web server using HTTPS (Hypertext Transfer Protocol Secure) communication protocol. The programmable environment and the middlewares are written in Phython [10]. The back-end database can be implemented using SQLite [14].

The Python programming environment is used as integrating middleware between the Web server front-end and the database back-end. Some of the important Python packages used to implement the system were:

1. mod_python - live-programmable module to Apache,
2. Crypto - cryptographic libraries,
3. xmlrpclib - XML-based RPC (Remote Procedure Call),
4. tlslite - SSL v3 (Secure Sockets Layer) and TLS v1 (Transport Layer Security) libraries,
5. standard Python libraries for examples - SocketServer, BaseHTTPServer, sqlite3, base64 and binascii.

4 System

4.1 System Components

As mentioned earlier in Section 2, the **W/CMS** can be broken down into three logically separate subsystems as shown in Figure 6. Modelled from these decompositions, the proposed system will provide these main Web user interfaces as shown in Table 2. Then, building on top of these Web interfaces, the system will support these main operational functionalities as shown in Table 3. Utility functions are coupled directly to Web interfaces. The codes that implement these Web interfaces and the utility functions are grouped into five groups:

1. Web-based interfaces - **W**$\star$
2. user management - **U**$\star$
3. session and access controller - **S**$\star$
4. contract management utilities - **C**$\star$
5. report, archive and cleanup - **R**$\star$

Table 2: Web interfaces decomposition of **W/CMS** subsystems

Subsystems	Web user interfaces
Users Management	**W1** `registration` **W2** `login` **W3** `changePwd`
Contracts Formalization	**W2** `login` (*) **W4** `contractReq` **W5** `contractNego` **W6** `contractAgre`
Contracts Finalization	**W2** `login` (*) **W7** `contractWitn` **W8** `contractClose` **W9** `reportReq`

All user interface functions are labelled with **W**. The function **W2** (`login`) is shared between all three subsystems. Behind each **W**$\star$ are utilities functions. Whenever an interface is used by a user, these utilities functions will get invoked, for example;

W2 $\rightarrow$ (**S1**, **S2**)

(**W6**, **W8**) $\rightarrow$ **C3**

These are just a top-level view of the proposed system. There are much more functions in the **W/CMS** than the ones listed in Table 2 and Table 3.

Table 3: Functional decomposition of **W/CMS** subsystems

Web user interfaces	Utility functions
W1 registration	**U1** userReg
	U2 keyGen
W2 login	**S1** userSession
	S2 authorization
W3 changePwd	**U3** keyReset
W2 login (*)	**S1** userSession
	S2 authorization
W4 contractReq	**C1** contractInit
W5 contractNego	**C2** blogUtil
W6 contractAgre	**C3** signingUtil
	C4 finalize
W2 login (*)	**S1** userSession (*)
	S2 authorization (*)
W7 contractWitn	**C5** verifyingUtil
W8 contractClose	**C3** signingUtil (*)
W9 reportReq	**R1** report
	R2 archive
	R3 cleanUp

4.2 System Operational Descriptions

Under the registration interface **W1** a person will be asked to provide email address. The system will initialize a user in the database and send an activation code to the previously given email address. Using this activation code the person will be able to provide the system with her choice of username and password. This will then trigger **U1** which will register user data into the database. The **U1** will in turn trigger **U2** which will produce a pair of public K_{pub} and private K_{prv} keys for the newly registered user. The private key is encrypted using Blowfish encryption with the user's password as the secret key, $K^{bf}_{prv} = \beta_{enc}(pwd, K_{prv})$. The public key K_{pub} and encrypted private key K^{bf}_{prv} are then securely stored in the system database. The system will then create a *user package*, which is an encrypted message package containing user data and the keys, and send this message via email to the user for safe keeping.

A registered user can change her password at **W3** by providing the system with her email address and activation code. If the system recognizes the given data the system will then request the user to submit the old password and a new password, together with the user's *user package*. The system will then execute **U3** which will decrypt the *user package* and extract user data and private key. If the user data is valid then the system will decrypt the K^{bf}_{prv} using the old password and re-encrypt the private key using the new password and store them into the database.

To make use of the facilities provided by the **W/CMS**, users must first authenticate themselves at **W2**. The two utility functions associated with **W2** are **S1** and **S2**. The **S1** manages users sessions of all currently authenticated users. Each *user-session* contains all necessary information about a particular current session, in particular the user current authorization and permissions. *User-session* is directly related to session Web-cookie.

The **W4** provides Web interface to a *contract requester* and an initiator $\mathbb{I}$ for submitting request and approving for a new contract, respectively. Whenever the request is approved, the initiator $\mathbb{I}$ must check that the provided initial contract data is complete and correct. Related to **W4** is **C1**. The purpose of **C1** is

1. to initialize a new database with all the contract objects and attributes, all participating actors $\mathbb{U}^A$ and $\mathbb{U}^P$, and

2. to create all the necessary scaffoldings for the new contract management work.

The contract negotiation will be done in an Web environment similar to a *blog* at **W5**. The initiator $\mathbb{I}$ will function as coordinator and a mediator. All $\mathbb{U}^A$ will be given access to contact database and environment. When agreement is reached by all $\mathbb{U}^A$, the $\mathbb{I}$ will coordinate the signing and the finalization of the contract using the **W6**. Related to **W6** is **C3**, the signing utility function. When the *contract synopsis* is produced, an event is triggered on **C4**.

The **C4** will close the *contract blog* and alert the $\mathbb{A}$ and if necessary the witnesses $\mathbb{W}$. By using the interface **W6** the $\mathbb{W}$ will function as witness to the *contract synopsis* by first verifying the signatures using **C5** and then sign using **C3**. Likewise the archive $\mathbb{A}$ will put her seal by signing using **C3** to the already witnessed *contract synopsis*. All important data particular to the contract will be archived by $\mathbb{A}$ by invoking **R2** and their traces are deleted using **R3**. A standard report is produced by **R1** and send via email to all $\mathbb{U}^A$ as a receipt.

5 Conclusion

A significant part of this paper is devoted to conceptual discussions on Web-based Contract Management

system. We identified three possible types of contracts. With two step user-case diagrams we proposed a system model on loosely coupled system of three main subsystems. We provide a table showing functional components of these subsystems. These functions are further decomposed into utility functions that are logically grouped. All these functions are implemented as modules in our prototype implementation of the system.

A combination of cryptographic hash functions (SHA), symmetric-key (Blowfish) and and public-key (RSA) cryptographic tools provided by `hashlib` and `Crypto` modules in Python made it possible to easily implement hash calculation, encryption/decryption, sign/verify and even blind/unblind. There are several different cryptographic tools provided by Python cryptographic libraries.

For the purpose of providing one database per contract, the SQLite database is more than adequate. Connectivity between Web front-end to SQLite back-end database using sqlite3 is robust and easy to utilize. Blob data type is very useful for storing binary documents. Each SQLite database is implemented as regular file in the operating system, so the whole database can be encrypted as any regular file. Other database systems may be used instead.

All the software tools used to implement are open-source and free. The model provides an implementation structure which is open-ended in the sense that different tools can be freely mixed-and-matched.

How the proposed system measures up against the 1) Usability, 2) Adaptability, 3) Scalability, 4) Interoperability, and 5) Maintainability criteria mention in Section 2.3?

The answers are definitively, 'yes' - it measures up well, on all counts. It is usable since the system is Web-based. Web-based interfaces are well known to nowadays average computer users. It is adaptable since the system is made up of open software components. Any new functionality can be incorporated easily. Subsystems can be implemented on different servers to provide scalability.

By using standard - HTTP, XML, XML-RPC, the system can inter-operate with other systems. Since the system is build around subsystems and modular components, it will promote maintainability under both development and deployment.

We can not help but to admire the Kushans people for their ingenuity. With simple and primitive technologies they have shown us the basic principle of contract making. What we did is nothing more than what had been done fifteen centuries ago albeit our advances in computers and communication technology.

References:

[1] Apache, *The Apache Software Foundation, HTTP server*, `http://www.apache.org/`, 2009 (last accessed).

[2] David C., *Blind signatures for untraceable payments*, Advances in Cryptology - Crypto '82, Springer-Verlag pp. 199-203, 1983.

[3] FIB PUB 180-3, *SECURE HASH STANDARD (SHS)*, Federal Information Processing Standards Publication, 2008.

[4] Jonsson J., Kaliski B., *Public-Key Cryptography Standards (PKCS) #1*, RSA Cryptography Specifications Version 2.1 RSA Laboratories, 2003.

[5] Halevi, S., Krawczyk, H., Strengthening Digital Signatures via Randomized Hashing, Advances in Cryptology - Crypto 2006, Vol. 4117/2006, Springer Berlin, 2006, pp. 41-59.

[6] Kelsey, J., Schneier, B., *Second Preimages on n-Bit Hash Functions for Much Less than 2n Work*, Advances in Cryptology - Eurocrypt 2005, Vol. 3494, Springer Berlin, 2005, pp. 474-490.

[7] Kovari, B., Albert, I., and Charaf, H., *A general approach to off-line signature verification*, WSEAS Transactions on Computers Vol.7, No. 10, 2008, pp. 1648-1657.

[8] Lin J., Yu J., and Hsu C., *An Ontology-Based Architecture for Consumer Support Systems*, WSEAS Transactions on Information Science and Applications, Vol. 7 (2), pp. 153-165, 2010.

[9] Malek J. A., *Informative Global Community Development Index of Intelligent City*, WSEAS Transactions on Information Science and Applications, Vol. 7 (1), pp. 114-121, 2010.

[10] Python Programming Language, `http://www.python.org/`, 2009 (last accessed).

[11] Rahman S.M.M., Masum S.M., Khan M.S.I., Alam M.S., and Hasan M.I., *A New Message Digest Function for Message Authentication*, WSEAS Transactions on Computers, Vol. 3 (5), 2004, pp. 1466 -1469.

[12] Rivest R., *The MD5 Message-Digest Algorithm*, MIT Laboratory for Computer Science and RSA Data Security, Inc, 1992.

[13] Schneier B., *The Blowfish Encryption Algorithm.* `http://www.schneier.com/blowfish.html`, 2009 (last accessed).

[14] SQLite, *A self-contained, serverless, zero-configuration, transactional SQL database engine*, `http://www.sqlite.org/`, 2009 (last accessed).

[15] Simonova S. , Kopackova H., *eDocument in eGovernment*, WSEAS Transactions on Information Science and Applications, Vol. 7 (1), pp. 92-101, 2010.

[16] Sims-Williams N., *Bactrian Documents from Ancient Afghanistan*, http://www.gengo.l.u-tokyo.ac.jp/~hkum/bactrian.html, 2010 (last accessed).

[17] Stinson, D., *Some observations on the theory of cryptographic hash functions*, Designs, Codes and Cryptography, Vol. 38 2006, pp. 259 - 277.

[18] Tsang-Yean L. and Huey-Ming, L., *Encryption and decryption algorithm of data transmission in network security*, WSEAS Transactions on Information Science and Applications. Vol. 3 (12), 2006, pp. 2557-2562.

[19] Tuba M. and Stanarevic N., *Relation between Successfulness of Birthday Attack on Digital Signature and Hash Function Irregularity*, WSEAS Transactions on Information Science and Applications, Vol. 7 (2), 2010 pp. 186-195.