Secure Processes in Applications Workflow Facilitating Systems for On-line Exams

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

SYLVIA ENCHEVA
Stord/Haugesund University College
Department Haugesund
Bjørnsonsg. 45, 5528 Haugesund
NORWAY
sbe@hsh.no

Abstract: By employing basic security applications readily provided by well established cryptographic technologies, workflows in a system for on-line students' assessments and on-line exams can be done securely. By analyzing data flow between actors in the workflow, secure processes can be implemented using Web, database and cryptographic techniques. The goals of the implementation are to provide system users with data privacy, data authenticity and non-repudiation for different phases in the workflow depending on data sensitivity issues and policies.

Key–Words: Student assessments, applications security, multi-tier Web-based application, workflow modeling.

1 Introduction

The purpose of examinations is to assess students' level of learning of knowledge or/and acquired skills after they have been given lessons on specific subjects, whereby the exam scores reflex the degree of assimilation of knowledge and practical understanding in a particular subject under consideration. There are many reasons why educational organizations need to measure what students know and can do. Among others, these evaluations provide critical information needed for ranking students against some national or international standards which students must pass in order to graduate.

In this context, examinations are instruments for collecting information which will be appropriately interpreted for specific decision making processes. Informations that made up the examinations compiled by the teachers, the operational data in the examinations' processes used by the system, the answering data saved after the examinations were completed by the students and the interpreted results, all these and the interfaces between them, then most necessarily need to be handled securely. In order to function as critical measuring entities, these informations need a high degree of confident of validity to all the participating parties in the process chain.

With the current state of computer hardware and software technologies, unprecedented opportunity for on-line examination presents itself by which educational institutions can enjoy the many benefits provided by such systems. Potentially, these benefits offer conveniences and flexibility for the students and teachers, increase of efficiencies and productivities to the administration and security implementations that have the intention to secure data in all phases of on-line examination processes. In these paper we discuss security challenges for securing informations in relation to on-line examinations by providing security model and proposing an implementation framework.

The proposed security implementation makes use of standard techniques provided by the public-key cryptography. These security techniques facilitate the implementation of system procedures that implement encryption and decryption, protection against tampering, provide authentication and lastly, but by no mean least important, ensure non-repudiation.

An educational organization which implements the framework will manage its own public key infrastructure (PKI). A centrally appointed trusted unit, e.g. a security group of a central IT department is responsible in assigning and managing users private-public key pairs. The proposed implementation will not depend and trust an external party for issuing users key pairs and digital certificates. The PKI is managed and owned by the organization and for the intra-domain usage proposal presented in this paper such arrangement is adequate. Cross domains implementations will be possible for a cross domains trust based on mutual agreement and formal contracts. Application of computer based assessment is discussed in [3], [5], and [8].

The rest of the paper is organized as follows. Sec-

tion 2 contains background on cryptographic tools and assessment types. In Section 3 we discuss 1) workflow model, 2) security model, and 3) data model. Section 4 describes system implementation. Section 5 contains the conclusion of this work.

2 Background

2.1 Cryptographic Tools

In relation to security implementation discussed in this paper three important and well known cryptography techniques will be employed. They are 1) public-key encryption implemented in RSA [6] public-key cryptography, 2) cryptographic hash functions implemented by MD5 [11] and SHA [4] hash algorithms and 3) symmetric-key encryption implemented in Blowfish [14].

2.1.1 RSA Public-key Cryptography

The RSA algorithm, was first introduced in 1978, is called after its inventors 1) Ron Rivest, 2) Adi Shamir, and 3) Leonard Adleman while they were working at MIT.

RSA [12] 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. The RSA algorithm is fairly straightforward and simple and is described as follows:

1. find two large prime numbers, **p** and **q** and multiply these together to give **n**
2. select a number **e** which is less than **n** and prime to (**p** - 1)×(**q** - 1), so that **e** and (**p** - 1)×(**q** - 1) have no common factors
3. select another number **d**, where (**e**×**d** - 1) is divisible by (**p** -1)×(**q** - 1)
4. now (**n**, **e**) is the public key and (**n**, **d**) is the private key, where **e** is the public exponent and **d** is the private exponent
5. to encrypt a message **m**, create the ciphertext **c** such that $\mathbf{c} = \mathbf{m}^{\mathbf{e}} \bmod \mathbf{n}$
6. to decrypt the ciphertext **c**, calculate **m** such that $\mathbf{m} = \mathbf{c}^{\mathbf{d}} \bmod \mathbf{n}$

Primes **p** and **q** are no longer needed but cannot be disclosed. It is standard to refer to the bit length of the modulus **n** as the size of the RSA key.

The private exponent **d** can be obtained if factors **p** and **q** could be determined from **n**. The security of RSA cryptographic system depends on the difficulty of factoring large integers. Therefore longer RSA keys provide better security level than shorter ones.

RSA Encryption scheme:

- *Encryption*:
  ciphertext, **c** = RsaPublic(**m**) = $\mathbf{m}^{\mathbf{e}} \bmod \mathbf{n}$
- *Decryption*:
  plaintext, **m** = RsaPrivate(**c**) = $\mathbf{c}^{\mathbf{d}} \bmod \mathbf{n}$
- *Inverse transformation*:
  m = RsaPrivate(RsaPublic(**m**))

RSA Signature scheme:

- *Signing*:
  signature, **s** = RsaPrivate(**m**) = $\mathbf{m}^{\mathbf{d}} \bmod \mathbf{n}$
- *Verification*:
  verify, **v** = RsaPublic(**s**) = $\mathbf{s}^{\mathbf{e}} \bmod \mathbf{n}$
- *Inverse transformation*:
  m = RsaPublic(RsaPrivate(**m**))

More detailed discussion on Public-Key Cryptography Standards (PKCS) #1 v2.1 by RSA Laboratories can be found in RFC3447 [6].

2.1.2 Cryptographic Hash Functions

The purpose of using hash function is of three folds 1) to convert a variable length usually large document to a small size and fixed length hash value, 2) to hide the actual document but relate to it uniquely to its hash value, and 2) any change in the document (however small it might be) will produce a radically different hash. For example the strings '123456' and '123457' will have these SHA256 hashes:

123456 - 8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92
123457 - 54b688a517f7654563a6c64d945a3670880a4c602ec67a065bbebbcd2b22edd5

The use of hashes is an important part of a security procedure of digital signature which will be discussed in detail in Section 3.2. The basic idea is that a cryptographically signed hash can only be cryptographically verified if the original document has not change from its original form, thus provide protection against tampering of original document as well as non-repudiation.

The most widely used cryptographic hash functions are MD5 and SHA-1. The SHA hash functions was designed by the National Security Agency (NSA) of the United States of America and was first introduced in 1993. A successful attack on SHA-1 was reported in 2005. At the end of 2008, it was reported

that the MD5 hash has been broken. Therefore it is advisable to use SHA-2 or SHA-3 to ensure the long-term robustness of an application that employed hash functions as a part of its security implementations.

2.1.3 Blowfish Symmetric-key Encryption

Blowfish belongs to a class of block ciphers first introduced in 1993 by Bruce Schneier as a replacement to all others ciphers of that time to be placed in public domain, thus unrestricted by patents and exports regulations, [16].

Blowfish operates with 8 bytes block size and has a variable key length from 4 bytes and up to 56 bytes. As all others block cipher, the document to be encrypted may need padding to accommodate the block size constrain, and this padding then need to be stripped in decryption process.

Blowfish Encryption scheme:

- *Encryption*:
  ciphertext, $\mathbf{c} = \mathbf{BF}_{enc}(Key, \mathbf{m})$
- *Decryption*:
  plaintext, $\mathbf{m} = \mathbf{BF}_{dec}(Key, \mathbf{c})$
- *Inverse transformation*:
  $\mathbf{m} = \mathbf{BF}_{dec}(Key, (\mathbf{BF}_{enc}(Key, \mathbf{m})))$

It is a well known fact that public-key encryption/decryption is much more slower than symmetric-key encryption/decryption. A well know technique called a digital envelope is commonly used to combine both public-key and symmetric-key to provide both privacy and authentication in message exchange. The digital envelope will be discussed fully in Section 3.2.

2.2 Students Assessment Types

There are subtle differences between accumulating of knowledge and mastering of a skill and obtaining competency. Knowledge is successfully assimilated when it is applied skilfully and competently. In other words, a learning process is a process of acquiring practical knowledge from factual knowledge. The ability to perform skilfully is very much dependable on what one knows but what one knows does not amount to anything if one does not perform.

The task of assessing students is both challenging and complex, [7]. The complexity may be related to the multiplicity of purposes of student assessment tasks. Among others, assessment is used:

(i) to measure students' knowledge level for further study as in entry exams;

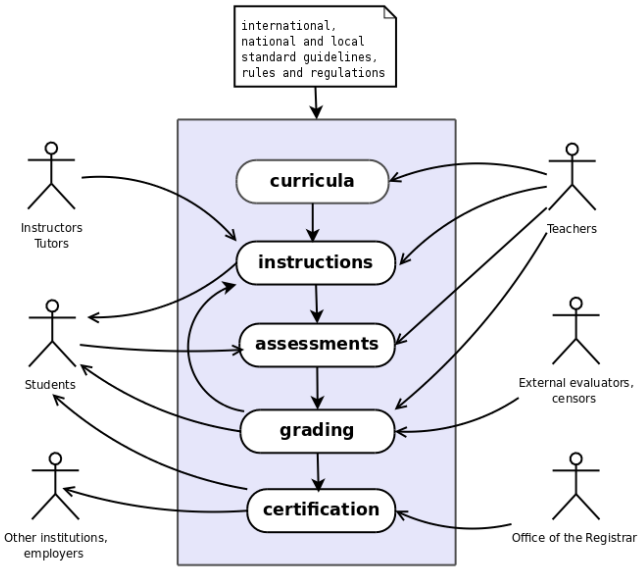

Figure 1: Curriculum to Certification

(ii) to measure students' level of skill and competency;

(iii) to grade students for professional accreditation or academics certifications;

(iv) to ranks students, relative to one another in competition for study grands and scholarships;

(v) to measure the effectiveness of learning procedures and environments;

(vi) to provide feedback on student learning for both students and staff;

(vii) to provide feedback on teaching for staff;

(viii) to direct students' learning; and

(ix) to define and protect academic standards.

Researchers in educational sciences categorized assessment into four broad categories of 1) placement assessment, 2) formative assessment, 3) summative assessment and 4) performance assessment.

Placement assessment is not directly connected to a particular curriculum but rather to a specific criteria expressed in the competency standards for specific purpose. The placement assessment measures students both on their knowledge and competency measured against some predefined standards.

Formative assessment is directly connected to a particular course or a part of a larger program and is essentially diagnostic in its application [15]. The

main concern here is to provide regular feedbacks to both students and instructors and thus guide students through the most optimal learning path. Formative assessment can be directly coupled to an intelligent tutoring system.

Summative assessment is directly connected to a particular curriculum and its purpose is to establish a formal end exams setting for a particular course. The main objective here is to produce marks or grades which may be used for reports of various types. The marks and grades will then be used to provide students with certificates, diplomas and degrees to be used as measure of academic achievement or professional licenses. In a competitive jobs seeking scenario of a student's future, the results of summative assessment are of paramount importance.

Performance assessment is very much suitable for work-based assessment outside the normal educational institutions. Performance assessment measures how well a person applies knowledge and skills from different domains to solve actual and practical problems. With the fast changing modern technologies no one is actually fully equipped with sufficient knowledge after graduation to competently face the challenges of working live. Skilled workers of today will never be able to stop acquiring new knowledge since they have to cope with everydays' challenges. Live long learning is a real necessity in our modern society.

3 Model

3.1 Workflow Model

It is now hopefully been established that a students' exam is more of a process than just a single event of students' assessment. Four major actors are directly involved in this process, namely 1) teachers, 2) students, 3) censors and 4) administration as shown in Figure 1.

Process workflow necessitates dataflow from one actor to another. Furthermore, dataflow needs storage, transport and coordination, and in this case all these mentioned tasks need to be done securely for obvious reasons. A simplified workflow model is presented in Figure 2.

Each examination is a collection of problems with corresponding correct solutions developed by teachers or exams' designers for a particular purpose and a particular course. Students taking the exams (or assessment) will be presented with questions directly related to the problems and are asked to provide answers. The students answers are compared with the corresponding solutions provided earlier. The scores are calculated from these comparisons. The scores will be used

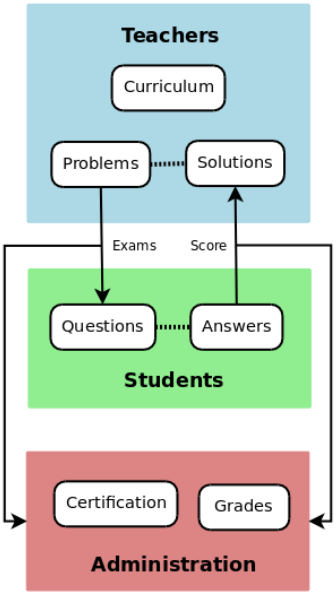

Figure 2: Exams Workflow Model

to provide control data for instructional strategies in case of formative assessment and the same scores can be converted to ranks or grades for other form of assessments.

Based on the model diagram shown in Figure 2 the operational units E_T, E_S, E_R and G_S in the workflow can be identified and defined symbolically as follows:

$$E_T = \{P_i\}, P_i = (p_i, s_i), 1 \leq i \leq n$$

$$E_S = \{Q_j\}, Q_j = (q_j, a_j), 1 \leq j \leq n,$$
$$j = i, \{q_j\} \subseteq \{p_i\}$$

$$E_R = \{R_j\}, R_j = (q_j, r_j), 1 \leq j \leq n,$$
$$r_j = score(a_j, s_j), 0 \leq \sum_j r_j \leq 100\%$$

$$G_S = grade(\{r_j\})$$

The teachers' E_T unit is a collection of distinct pairs of a problem and an associated solution. It is assumed that each problem has a unique solution. This set of pairs constitute all questions where potentially a subset of it can be used for assessments. The set of questions Q in E_S is a subset of P in E_T. Moreover elements of E_T can be shared among teachers across different organizations as shareable learning objects.

The students' E_S is a collection of distinct pairs of questions and answers. The answers given can be either 1) empty, 2) wrong, 3) partially correct or 4) correct. The function $score$ compares the answer to the corresponding solution of a particular question either manually of automatically to calculate results of E_R units. The values of each set of question and resulting pairs will be used to $grade$ students G_S.

The process handling for coordination, storage and transport of E_T, E_S, E_R and G_S need to implement security mechanisms 1) to prevent unauthorized and untimely access, 2) to protect against information tempering, and 3) to resolve dispute with non-repudiation techniques.

3.2 Security Model

In principle, there is no difference between a message exchanged over distance or a message exchanged over time. A massage sent from point p_a to point p_b has to be protected from eavesdropping and tampering. The same security problem is also immutably true fora message stored over a time period. A message stored at time t_0 must be protected for privacy, integrity and trustworthiness when it is read at a later time t_1.

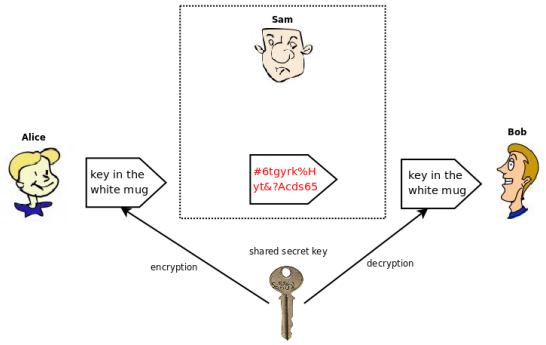

Figure 3: Symmetric-key Encryption/Decryption

On the first look, a symmetric-key (or a secret-key) encryption seems adequate for message exchange between two communicating parties sharing a secret key. In Figure 3 Alice is sending a private message to Bob using a secret key that they both know. Larking in the background unknown to them stands Sam ever ready to break the encrypted message. Now, Sam has many ways to intrude into the private communication between Alice and Bob. The easiest way is to steal the secret key.

Once the secret key shared by Alice and Bob was stolen by one mean or another, the communication channel between them is no longer secure. The worst

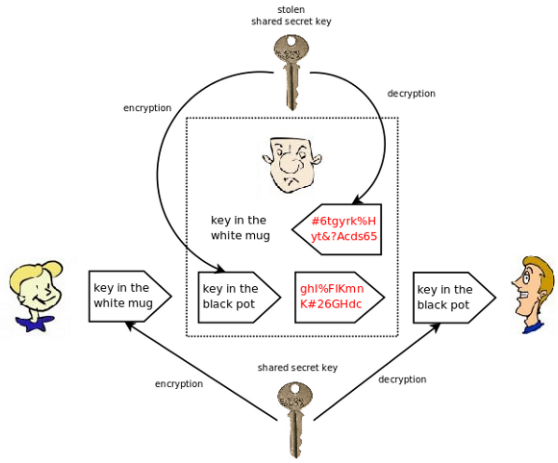

Figure 4: Stolen Secret Key Problem

part of it all is that the infringement is transparent to both of them. Sam can not only read their secret message but even change the original message without Alice and Bob knowing as shown in Figure 4.

The solution to this problem is to employ a public-key encryption scheme provided by RSA that supports both encryption/decryption and sign/verify security protocols, Figure 5. RSA scheme operates on a message asymmetrically using one part of a pair of related keys to encrypt and decrypt a message, likewise for signing and verification.

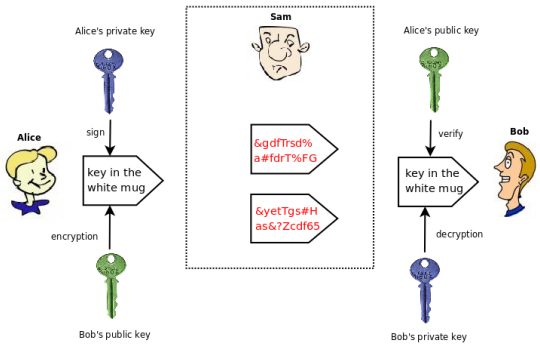

Figure 5: Public-key Encryption/Decryption

Each communicating party will use theirs keys asymmetrically when exchanging messages. Alice will use Bob's public key to encrypt a message from her to Bob. Alice will also sign her message using her own private key. Any ones' public keys were made public. Anybody can send en encrypted message to

Bob using Bob's public key, including Sam. However, Bob is only interested in Alice's message, therefore only accepts a message signed by her. The way Bob does this is to use Alice's public key to verify Alice's signature. The communication between Alice and Bob will remain secure as long as their respective private keys are keep secret to themselves. If Alice's private key was lost to Sam then Sam can fake a message to Bob as though the message came from Alice. If Bob's private key was lost to Sam then Sam can read all messages addressed to Bob.

The security implementations makes use of some well known cryptographic applications of 1) digital envelop and 2) digital signature [13]. These security applications will employ a combination of 1) public-key encryption, 2) cryptographic hash functions, and 3) symmetric-key encryption techniques.

Digital envelope:

$$
\begin{aligned}
&pack \\
&\quad text_{enc} = blowfish_{enc}(S_{clear}, text_{clear}) \\
&\quad S_{enc} = rsa_{enc}(K_{pub\_B}, S_{clear}) \\
&\quad envelop = \{S_{enc}, text_{enc}\} \\
&unpack \\
&\quad \{S_{enc}, text_{enc}\} = envelop \\
&\quad S_{clear} = rsa_{dec}(K_{prv\_B}, S_{enc}) \\
&\quad text_{clear} = blowfish_{dec}(S_{clear}, text_{enc})
\end{aligned}
$$

Digital signature:

$$
\begin{aligned}
&signing \\
&\quad ts = timestamp \\
&\quad hash = sha2(ts_A + text_{clear}) \\
&\quad signature_A = rsa_{sign}(K_{prv\_A}, hash) \\
&\quad sign_{data} = \{text_{clear}, ts_A, signature_A\} \\
&verification \\
&\quad \{text_{clear}, ts_A, signature_A\} = sign_{data} \\
&\quad hash = sha2(ts_A + text_{clear}) \\
&\quad verify = rsa_{verify}(K_{pub\_A}, hash, signature_A)
\end{aligned}
$$

3.3 Data Model

Managing an enterprise PKI is not a trivial matter. The process of making key pairs is expensive in terms of processing resources and time. Figure 6 shows the operational key management for each user, however a similar table in a limited access database needs to be provided to store users' RSA key-pairs. The private keys will be protected by an administrative global secret key.

Using this scheme, it is now possible to manufacture all RSA key-pairs in advance for each user. The users will be given their keys and protect their private keys using their own secret keys upon user request. There will be no problem to set a new secret key onto a user private key should the user lost her current secret key.

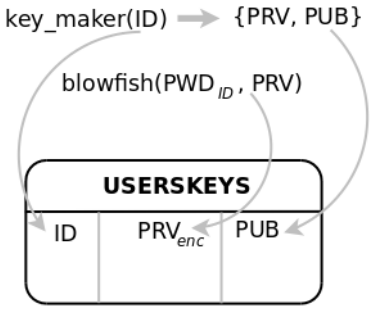

Figure 6: Keys making function

The essential data tables presented in Figure 6 and Figure 7 are the necessary part of the whole data model to represent the security elements. To be sufficient other operational data tables are used in the implementation. These data tables will contain control information on the security elements, such as validation and invalidation dates on *SECURESTORE* when in particular the entries in this table contain elements for exams.

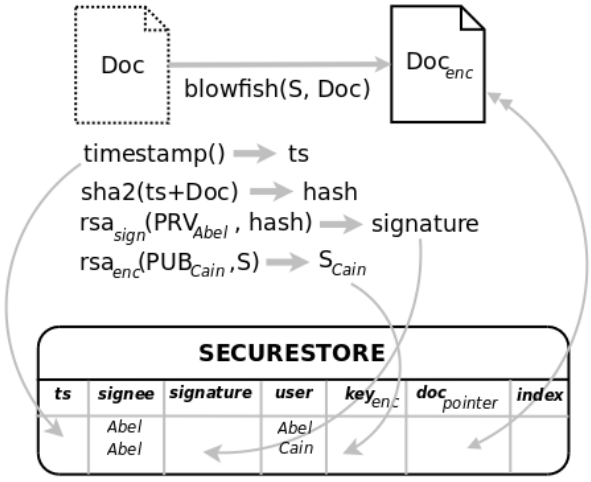

Figure 7: Secure Storage

It is appropriate to mention at this point the special table entry where the *signee* and the *user* attributes are equal. This situation indicates that the user can assign secure access to any new user of the document. The *signee* is the owner or the originator of the document while the *user* is the reader of the document. The secret key is not known to anyone directly,

so while any valid *user* can decrypt the document with the help of her private RSA key, only the *signee* can distribute new access rights via a special system application.

4 System

The system is implemented as a Web-based system, Figure 8. The users interact with an Apache [1], http server. The programmable environment and the middlewares are written in Phython, [10]. The backend database is implemented using PostgreSQL, [9].

It is essential that the system has the support of an enterprise identity management system (IDM). The IDM provides all the necessary and trustworthy users' informations on particular users identities and roles. The system will be relying on the enterprise IDM for the source of users authentication and authorization.

The system needs to manage its own public key infrastructure (PKI). Each system user will be given a pair of RSA private and public keys. The user's private key is protected with a password using symmetric-key encryption. A document securely published in the system is protected with a randomly generated system key and a symmetric-key encrypted by the system. Any user allegeable to read the encrypted document will be given the encryption via a key $rsa_{dec}(K_{prv}, rsa_{enc}(K_{pub}, key))$ using her RSA key-pair. The secret key thus obtained will be used to decrypt the document. Moreover the published document is signed by the owner using $rsa_{sign}(K_{prv}, sha2(ts + Doc))$ to ensure the genuineness of the document through the process of verification using the owner RSA key-pair. These security mechanisms are presented in Figure 6 and Figure 7 showing relationships between cryptographic functions and data model elements.

4.1 Security Code Implementations

Firstly the utility module `bf.py` is defined. This module which provides Blowfish functions will be useful in 1) securing private keys, and 2) processing documents digital envelopes. The module implementation is making use of the `Crypto` module for its cryptographic functions and `binascii` module for binary to ascii conversion. It is worth noting that `binascii.b2a_base64` output string is about 50% less in size then `binascii.b2a_hex` output string. If no key is given the `BLOWFISH` class will create one randomly, during its initialization.

```
# bf.py
# usage: from bf import BLOWFISH
from Crypto.Util import randpool
from Crypto.Cipher import Blowfish as BF
```

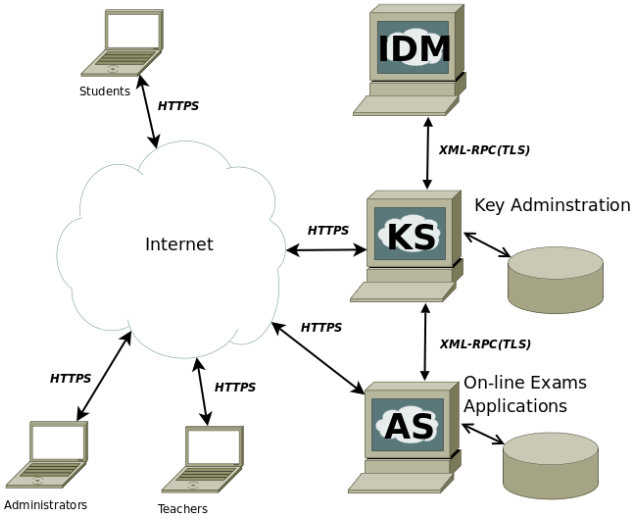

Figure 8: System Architecture

```
import binascii

class BLOWFISH:
   def __init__(self, key=''):
     if key == '':
        ran = randpool.RandomPool()
        ran.randomize()
        self.k = ran.get_bytes(56)
     else:
        # The key must be 56 char long
        while len(key) < 56: key += key
        self.k = binascii.a2b_base64(key)

     # blowfish cipher engine
     self.blowfish = BF.new(self.k)

   def key(self):
     return binascii.b2a_base64(self.k)

   def encrypt(self, txt):
     p = len(txt) % 8
     # padding string
     pad = '%d' % (8-p) + '*' * (8-p-1)
     return binascii.b2a_base64\
     (self.blowfish.encrypt(pad+txt))

   def decrypt(self, txt):
     dec = self.blowfish.decrypt\
     (binascii.a2b_base64(txt))
     return dec[int(dec[0]):]
```

Second utility module is `rsa_keys.py` module. Using the `bf.py` previously defined, this module is used to create RSA key-pairs. The private key part of the key-pair will be protected using blowfish encryption. Python standard library `cPickle` is use to convert the internal data-structure of the RSA keys. `cPickle.dumps` converts internal data-structure to external representation while `cPickle.loads` re-

verses the process. The utilities functions defined in this module are need for creating, saving and using RSA key-pairs.

The `prv_key_pwd()` function is used when a particular user's private key is protected using her password, while the `prv_key()` is used when `BLOWFISH` uses a random key.

```
# rsa_keys.py
# usage: from rsa_keys import *
from Crypto.PublicKey import RSA
from Crypto.Util import randpool
from bf import BLOWFISH

import cPickle
import binascii

rpool = randpool.RandomPool()

def get_keys(secret=''):
    if secret:
       bf = BLOWFISH(secret)
       bf_key = 'secret'
    else:
       bf = BLOWFISH()
       bf_key = bf.key()
    # Generate keys
    privkeyA = RSA.generate(1024, rpool.get_bytes)
    pubkeyA = privkeyA.publickey()
    # dump keys
    cb64_prvk = bf.encrypt\
                (cPickle.dumps(privkeyA))
    cb64_pubk = binascii.b2a_base64\
                (cPickle.dumps(pubkeyA))
    return (bf_key, cb64_prvk, cb64_pubk)

def prv_key(bf_key, cb64_prvk):
    bf = BLOWFISH(bf_key)
    prvk = cPickle.loads(bf.decrypt(cb64_prvk))
    return prvk

def prv_key_pwd(pwd, cb64_prvk):
    bf = BLOWFISH(pwd)
    prvk = cPickle.loads\
           (bf.decrypt(cb64_prvk))
    return prvk

def pub_key(cb64_pubk):
    pubk = cPickle.loads\
           (binascii.a2b_base64(cb64_pubk))
    return pubk
```

Third, the `digital_sign_envlp.py` utility module directly implements digital envelope and digital signature scheme as mentioned Section 3.2.

```
from Crypto.Hash import SHA256
from bf import BLOWFISH
import binascii
import time

def pack(secret, pub, text):
    bf = BLOWFISH(secret)
    enc_text = bf.encrypt(text)
    enc_secret = binascii.b2a_base64\
                 (pub.encrypt(secret, '')[0])
    return(enc_secret, enc_text)

def unpack(prv, enc_secret, enc_text):
    secret = prv.decrypt\
             ((binascii.a2b_base64(enc_secret),))
    bf = BLOWFISH(secret)
    text = bf.decrypt(enc_text)
    return text

def sign(prv, text):
    ts = str(time.time())
    g = SHA256.new(ts+':'+text)
    hg = g.hexdigest()
    signc = str(prv.sign(hg, "")[0])
    return (ts, signc)

def verify(pub, ts, text, signc):
    g = SHA256.new(ts+':'+text)
    hg = g.hexdigest()
    sign = (long(signc), )
    ok = pub.verify(hg, sign)
    return ok
```

The three utility modules can now be used to implement security code as can seen in this simple example.

```
from rsa_keys import *
from digital_sign_envlp import *

pwd = 'This is my secret password'
# create RSA key-pair; save p and q
b, p, q = get_keys(pwd)

# reload private and public keys
prv = prv_key_pwd(pwd, p)
pub = pub_key(q)

# the text
o_text = '''
This is the secret text we want to protect
'''

# digital envelop packing
sb, st = pack('sim11sam', pub, o_text)

# now the reverse
p_text = unpack(prv, sb, st)
print p_text

# digital sigature
ts, sig = sign(prv, o_text)

ver = verify(pub, ts, o_text, sig)
print ver
```

4.2 System Implementation

The system prototype implementing the basic functions was tested on an Ubuntu 9.04 Linux Kernel 2.6.28-11-generic with Intel Pentium Dual Core T3200 @ 2.00 Ghz and 4 Gbyte memory. The whole system was written in Python. Apache programmable framework was done using *mod_python* module while database middleware was implemented with the help of *pyPgSQL* Python module.

Figure 8 shows three independent servers.

1. **IDM** - Enterprise identity management system. The **IDM** is not a part of the system, but having an identity management system is crucial. The **IDM** provides all the necessary and trustworthy users' informations for users authentication and authorization.

2. **KS** - Key Administration server. All users defined in the **IDM** can create and administer RSA key-pairs. Both private and public keys are stored in **KS**. A particular user's private key is Blowfish encrypted using her user password.

3. **AS** - Application server. Users interact with the on-line assessments and on-line exams through **AS**. Any user using the services of **AS** must be registered on **KS** and has already created RSA key-pair.

Users interact with both **KS** and **AS** using their Web browsers over HTTPS (Hypertext Transfer Protocol Secure). The system does not require the clients to run client-side program. It is not a requirement that either Java or JavaScript is enable on the users' browsers. It is required that Web browsers support Web cookies, Web forms and URL redirects.

The three servers are loosely connected to each other using XML-RPC (XML-based remote procedure call) over TLS (Transport Layer Security). The communication programs are written based on 1) xmlrpclib - XML (Extensible Markup Language) RPC (Remote Procedure Call), 2) tlslite - SSL v3 (Secure Sockets Layer) and TLS v1 (Transport Layer Security) libraries, and 3) standard Python libraries for examples - SocketServer, BaseHTTPServer, base64 and binascii.

4.3 Test Results

The result of a speed test for generating RSA key-pair with different key size is as follow, where key is in bits, time is in seconds and the maximum size of text that can be encrypted in bytes, see Table 1.

Table 1: RSA Key Generating Speed Test

key	time	text	remarks
1024	0.2174	128	between 2006 and 2010
2048	0.3501	256	sufficient until 2030
3072	1.4122	384	useful beyond 2030
4096	7.4454	512	

The mileage will be different since the generating of RSA key-pair involves in finding a pair of suitable large prime numbers. Even though the calculations were done using Crypto Python module, the results show reasonable speed, less then 0.5 second for a key with 2048 bits size. Generating RSA key-pairs for 10000 users will take around 84 hours. Both the blowfish secret maximum key size and the SHA2 hash are of 56 bytes in length. Since RSA with 2048 bits can successfully operate on text with length up to 256 bytes, 56 bytes key and hash will not be a problem. The hash values were calculated using *hashlib* Python module and again the results show reasonable speed for practical purposes.

The results of speed tests for SHA2 function on different binary file sizes can be seen in Table 2.

Table 2: SHA2 Speed Test

test	size (bytes) (s)	time (seconds)	remarks
1	8666855	0.181262969971	s
2	34667420	0.428112983704	$s \times 4$
3	69334840	0.879555940628	$s \times 8$
4	277339360	1.71250987053	$s \times 32$

The results of speed tests for Blowfish encryption on different binary file sizes can be seen in Table 3. The encoded file will be $4/3$ bigger than the original document size due to base64 encoding.

Table 3: Blowfish encryption Speed Test

test	size (bytes) (s)	time (seconds)	remarks
1	8666855	0.288280963898	s
2	17333710	0.576570034027	$s \times 2$
3	34667420	1.16794395447	$s \times 4$
4	69334840	2.31661200523	$s \times 8$
5	138669680	4.60946893692	$s \times 16$

The results of speed tests for Blowfish decryption on different binary file sizes can be seen in Table 4. It can be observed that there is no significant speed difference between encryption and decryption using Blowfish. The time taken to encrypt and decrypt is linearly proportional to the document size.

Table 4: Blowfish decryption Speed Test

test	size (bytes) (s)	time (seconds)	remarks
1	11555809	0.295892953873	s
2	23111617	0.581923007965	$s \times 2$
3	46223233	1.17632222176	$s \times 4$
4	92446465	2.38166999817	$s \times 8$
5	184892921	4.86344981194	$s \times 16$

The result of speed test for RSA public-key (1024 bits) encryption and RSA private-key decryption with

120 Bytes block size on different binary file sizes can be seen in Table 5.

Table 5: RSA encryption/decryption Speed Test

test	encryption time (seconds)	decryption time (seconds)	remarks $s = 8666855$
1	9.52177882195	58.8852601051	s
2	19.0537331104	117.467578173	$s \times 2$
3	38.1037080288	234.975291967	$s \times 4$
4	76.222935915	470.42832303	$s \times 8$
5	152.550204039	940.82120204	$s \times 16$

The time taken to encrypt and decrypt is linearly proportional to the document size. However, it can be observed that the RSA private-key decryption takes approximately 6 times longer than RSA public-key encryption.

Blowfish encryption is 33 times faster than RSA public-key encryption. While, Blowfish decryption is 193 times faster than RSA private-key decryption. For practical reasons, using RSA encryption/decryption technique straight on a large document is not advisable.

5 Conclusion

A good security is a matter of implementing simple and understandable secure procedures within a workflow rather then complicated security protocols within a standard security framework. Secure procedures for on-line exams or student assessments can be implemented using existing ICT and cryptographic technologies.

Free and open source softwares and frameworks can be readily used to implement the supported security mechanisms to some degree of sophistication. Python facilitates implementation due to the many ready made modules in its library base. The speeds of sub-applications were found to be practically applicable.

With a simple and cheap implementation and at the same time robust and sophisticated, a Web-based application can be deployed that implements security mechanisms to prevent unauthorized and untimely access, to protect against tempering and can resolve dispute to an on-line exams workflow.

References:

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

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

[3] C. Boboila, G. V., Iordache, and M. S. Boboila, An Online System for Testing and Evaluation, *WSEAS Transactions on Advances in Engineering Education*, 1, 5, 2009, pp. 20–28

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

[5] X. Jin, Research on E-business Intelligent Examination System. *WSEAS Transactions on Information Science and Applications,* 1, 6, 2009, pp. 21–30

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

[7] Kerri-Lee Harris, Guide for Reviewing Assessment. *Centre for the Study of Higher Education, University of Melbourne*, 2005

[8] M. Pipan, T. Arh, B. J. Blazic, Evaluation Cycle Management - Model for Selection of the most Applicable Learning Management System, *WSEAS Transactions on Advances in Engineering Education*, 3, 5, 2008, pp. 129–136

[9] PostgreSQL, The world's most advanced open source database. *http://www.postgresql.org/* (last accessed 2009)

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

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

[12] R. Rivest, A. Shamir & L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21 (2), 1978, pp. 120-126

[13] RSA Laboratories, Simple Applications of Cryptography. *http://www.rsa.com/rsalabs/node.asp?id=2180* 2009 (last accessed)

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

[15] S. Walker and I. Reece, Teaching, Training & Learning: A Practical Guide. *Athenaeum Press, Gateshead*, 1997

[16] Wikipedia, Blowfish (cipher). *http://en.wikipedia.org/wiki/Blowfish_(cipher)* 2009 (last accessed)