Chapter 4

Security

An office document handling system requires security during production, storage and transport of documents. Therefore, security is an important aspect of the DPW problem. In the present chapter we discuss the security of DPW under a security framework and we propose a secure protocol for production of cases of a DPW.

4.1 The Security Framework

The security of a DPW can be studied in a framework formed by the following standard fundamental security concepts: authentication, access control, authorization, authorization flow, integrity, confidentiality and non-repudiation. All these concepts are well known. We are giving a review of these basic concepts in this framework.

Only a set of legitimate users, called the office workers are allowed to handle documents in a particular office. Therefore, authentication of users is a very important security requirement in an office system.

Definition 4.1 The authentication process is the aggregation of two pro-
cesses: identification and verification. Identification is the process whereby an entity claims a certain identity, while verification is the process whereby the claim is checked. When two communicating entities authenticate each other simultaneously it is referred to as mutual or peer authentication.

There are three primary methods of authentication of an entity:

K. authentication by knowledge: An entity is authenticated by the verification of some secret knowledge, for example, authentication by the knowledge of a password.

O. authentication by ownership: An entity is authenticated by the possession of some devices. Possession of an identity card is an example of this method.

C. authentication by characteristic: An entity can be authenticated by the characteristics unique to the entity. Characteristics like finger print, retinal pattern, DNA patterns etc in case of human being and message digests in case of digital messages can be used as characteristics for authentication.

Let \( A = \{K, O, C\} \) be the set of primary methods of authentication. All the authentication protocols available today use some combinations of these three methods. The set of all possible combination is the power set of \( S \) excluding the null set. The traditional password based method, \( \{K\} \subset A \), is vulnerable because it is easy to grab the password from the network [39]. A substantially better method is the smart card method, which is used in electronic commerce etc. Basically it uses the ownership of a card which contains a secret-key, which in turn is always encrypted by a password and stored in a chip of the card along with other information. It is the combination \( \{K, O\} \subset A \). This
method is also not perfect, because loss of the card and the leakage of the password may lead to forgery. Perfect authentication is an open problem and if a solution for that evolves, it will obviously be of the combination \( \{K, O, C\} \subseteq A \).

Access control and authorization are generally defined in terms of subjects and objects. Subjects are sometimes called the principals.

**Definition 4.2** A subject is an active entity of the system, which accesses objects. These accesses must be controlled to ensure that they match the security requirements.

**Definition 4.3** An object is a passive entity of the system, which contains information to be protected from unauthorized accesses.

If a user is properly authenticated in an office, it does not necessarily mean that the user can access all the objects in the office. Some types of objects may be accessible to only office workers of a certain role. Moreover, a worker of a role may not be authorized to access a particular object even though the role(s) he belongs to may have authorization to access that particular type of objects. How to ensure that access to information resources and hardware resources is available only to the authorized user is the subject matter of access control.

**Definition 4.4** Access control is the act of determining whether a subject is allowed to perform a given operation on an object.

Authorization is typically determined based on three access control policies: Mandatory Access Control (MAC), Discretionary Access Control (DAC) and Role-Based Access Control (RBAC).
• **MAC**: Under MAC, a security level is assigned to each subject reflecting the degree to which it is trusted to not disclose sensitive information and to each object in accordance with the sensitivity of the information that it contains. The set of security levels is partially ordered in a lattice-structured hierarchy so that each level dominates itself and the ones below it. If finer granularity is required, a set of categories can also be assigned to each subject specifying the areas in which it operates and to each object describing the areas to which the contained information pertains. Detail of MAC is available in [32].

• **DAC**: The main idea behind DAC is that of enforcing the rules specified by an access matrix describing the modes in which each subject is allowed to access each object. Since the access matrix is usually sparse, typical implementations adopt either a column-wise representation called Access Control Lists(ACL) or a row-wise representation called Capability Lists(CL). Detail of DAC is available in [32].

• **RBAC**: In RBAC, accesses by a user to an object is determined on the basis of the role(s) that the user plays at the time of access. A role is a representation of a set of responsibilities associated with a particular activity. There are two kinds of authorizations in RBAC:

  - The ACL for each object indicates the modes of access permitted for each role.

  - Each user is authorized to adopt a certain predetermined set of roles.

If a user's responsibility changes then new roles can be assigned. It is not necessary to go through the ACLs of all the objects one by one to
remove entries pertaining to the user in her previous capacity and add entries pertaining to the new role(s).

Definition 4.5 Authorization is defined as a 4-tuple \( A = \{ s, o, pr, [t_+, t_-] \} \), where subject \( s \) is granted access on object \( o \) with privilege \( pr \) at time \( t_+ \) and is revoked at time \( t_- \).

The time interval \([t_+, t_-]\) signifies that the subject \( s \) has privilege \( pr \) to access object \( o \) only during this time interval. In a workflow the authorization is non-monotonic in nature. A suitable authorization model for workflows must ensure that authorization is granted only when the task starts and revoked as soon as the task finishes. Otherwise a subject may possess authorization for time periods longer that required, which may compromise security. Therefore, authorization flow need to be tightly coupled to the workflow in order to ensure subjects possess authorization only when required.

Apart from authorization, the system must ensure to the users that the content of an object has not been tampered during transmission or while being stored. When an office object is stored or transported over a network from one point to the other, the confidentiality of the information contained in the object may need to be maintained. That means the information contained in the object be accessible only to the authorized subject. The type of access includes printing, displaying and revealing the existence of an object. In common practice, confidentiality of digital information is maintained by cryptographic encryption.

Definition 4.6 Confidentiality of an object is concerned with the protection of disclosure of information to unauthorized users.

In the production process of a document three main events are involved authoring a document, sending a document to the next reviewer, and receiving
a document from the previous reviewer of the document. These events have binary possibilities: whether an entity has authored or has not authored a document, whether a document has been sent or has not been sent, whether a document has been received or has not been received, whether a document has been sent / received at a given time or has not been sent / received at that time. If two possibilities of an event cannot be distinguished, a party related to the event could make the denials.

**Definition 4.7** Repudiation is defined as denial by one of the entities involved in a communication of having participated in all or part of the communication. Non-repudiation is concerned with preventing such a denial.

Digital signatures address the non-repudiation of authorship of a document, whereas non-repudiation of sending and receiving are addressed by non-repudiation protocols using evidence of sending (NRS) and of receiving (NRR) with a certified time stamp. It is shown in [44] that to address the non-repudiation of sending or of receiving a document at a given time, involvement of an trusted third party (TTP) is mandatory. The TTP must be in-line. In the protocol for production of MPMSD also an in-line is mandatory. This will be discussed in 4.2.

**Definition 4.8** A trusted third party (TTP) is a security authority or its agent, trusted by other entities with respect to security related activities [22].

From a communication viewpoint, three categories of TTPs can be distinguished based on the relative location to, and interaction with, the communicating parties. The three categories are in-line, on-line and off-line.

**Definition 4.9** A TTP is in-line if it is the intermediary, serving as the real-time means of communication between two entities A and B.
Definition 4.10 A TTP is on-line if it is involved in real-time during each protocol instance (communicating with A or B or both), but A and B communicate directly rather than through the TTP.

Definition 4.11 An TTP is off-line if it is not involved in the protocol in real-time, but prepares information a priori, which is available to A or B or both is and used during protocol execution.

Definition 4.12 A security policy is a set of rules that define the security subjects, security objects and relationship among them. Security policies define the principles on which access is granted or denied (without limiting legitimate access).

The definition of security policies lead to the explicit formulation of security strategies, thus giving security its rightful relevance instead of the fragmentary and approximate consideration it is often given [32]. In the perspective of security we can consider an office as a single trust domain.

Definition 4.13 A trust domain is a logical administrative structure within which a single, consistent security policy holds. Put another way a trust domain is a collection of both subjects and objects governed by single administration and a single security policy.

The scope of our discussion is limited to a single trust domain. Inter domain security is beyond the scope of this discussion.

There are standard user authentication schemes, like Smart cards. During run-time a process represents the user. Process authentication is discussed in [21]. Standard digital signature schemes address the fundamental issues, like proof of origin, content integrity (sec 1.8.2). The issue of confidentiality can also be addressed by standard encryption schemes. There are standard
non-repudiation protocols [1, 2, 3], which address the repudiation issues. But for persistent documents, the issue of signature replacement and the content integrity issue, where even the original author of a part of a MPMSD is not allowed to modify the content after it is dispatched to the next reviewer, are not addressed by standard digital signature schemes.

4.2 A Protocol for Secure Production of Cases

Based on the security framework discussed, we propose in the present section a protocol for secure production of MPMSDs, which are the cases of a DPW. The protocol addresses the issues particular to MPMSD. The protocol is based on a central arbitration mechanism. A preliminary version of the protocol appeared in [37].

4.2.1 The Central Arbiter

To make a protocol as general as possible, researchers attempt to avoid the use of an arbiter. However, to provide non-repudiation with time information, an in-line TTP is necessary. To provide non-repudiation of a digital signature, time of the signature is essential as the key used in the signature may become public at a later time. In our environment, documents are persistent and so non-repudiation of a digital signature is essential. So an in-line TTP will be required. Further, to prevent the reuse of parts (Sec. 1.8.3, item 2), an in-line TTP will evidently be required as immediate detection of such reuse will be necessary to prevent the taking of wrong decisions. When B and D collude to bypass C, C will not be aware of this till much later if no in-line TTP is present. Since an in-line TTP is mandatory for addressing issues like repudiation of sending and receiving documents as evident from [44], we can
address other issues on MPMSD as well, with an arbiter as an in-line TTP. Therefore, the production of cases in the DPW system is based on a central arbitration mechanism. It is basically a client/server computing paradigm, where the arbiter is the server. All the cases flowing from one client (reviewer) to another are routed through the arbiter. In case of dispute, the decision of the arbiter, based on the stored evidences, is final and binding.

A light weight TTP is usually favoured as this reduces the overhead of communication. However, in our case, the TTP has to perform more tasks. With our assumption of an organization as a single trust domain, within which the documents flow, an 'in-house' TTP can be implemented and this can therefore be made heavyweight.

4.2.2 Signed Session Keys

In most of the office solutions, public-key based digital signatures are used. Commonly, the RSA [30] algorithm is used for encryption. The computational complexity of the algorithm is high. Therefore, digital signatures based on RSA are slow. In an e-office a reviewer may have to sign many documents per day. This may lead to performance degradation of an e-office system. In our scheme, we have a novel idea of using the signed session keys for the digital signatures of the reviewers on a MPMSD. Actually this idea makes our protocols efficient. During a session, each reviewer has a session key, and a signed copy of the same, signed by the reviewer and certified and time-stamped by the arbiter, is available to only the reviewer and the arbiter. Now since the session key is known only to the reviewer and the arbiter and the arbiter is trusted and will not cheat as per our assumption, any message encrypted with this session key during the session can be treated as the digital signature of the reviewer on the message. In most practical imple-
mentations, public-key cryptography is used to secure and distribute session keys. Those session keys are used with symmetric algorithms to secure message traffic. Session key establishments are dynamic in nature, whereby the key established by a fixed pair of entities vary on subsequent executions. A key establishment protocol involves generation, transport and confirmation of keys. Authentication of communicating entities is also typically needed in a secure session key setup and is achieved by a signed session key. There are many protocols for signed session key establishment. X.509 [22] recommendations define 'strong two-way' and 'strong three-way' protocols. Both provides mutual entity authentication with optional key transport. Here 'strong' distinguishes these from simple password based methods and two-way and three-way refers to protocols with two and three message exchanges. In all these protocols for key establishment with entity authentication, session keys are either signed by the initiating entity and then encrypted with the public key of the recipient, or the encryption of the session key is done first and then the encrypted session key is signed before transport. Any standard signed session key establishment protocol can be used. For the ease of discussion we present here a protocol, in the line of the X.509 recommendation, integrating entity authentication, key transport and key confirmation together.

**Notations**

Before discussing the protocol, we present the notations here. The notations are also used in subsequent protocols almost unchanged.

\( w \) : a Document Production Workflow

\( A_1, A_2, A_3, \ldots, A_n \) : reviewers of \( w \)

\( N \) : a neutral arbiter, which is an in-line TTP.

\( CA \) : an off-line certifying authority, which generates digital certificates of
entities like $A$ and $N$, certifying the association of entity $X$ with its public key.

$k_{i,N}$ : shared session key between the $i^{th}$ reviewer $A_i$ and $N$.

$s_X$ and $p_X$: secret key and public-key of a principal $X$; $X$ may be a reviewer $A_i$ or $N$ or CA.

$\{m\}_k$ : message $m$ is encrypted with the key $k$.

$X \to Y : m$ : principal $X$ sends a message $m$ to principal $Y$ and $Y$ receives it intact.

$Cert_X = \{CA, X, p_X, t_v\}_{s_CA}$ : Certificate of $X$, where $t_v$ is the expiry time of the certificate.

$r_1, r_2, r_3$ etc : randomly generated nonces.

t$_1, t_2, t_3$ etc. : timestamps to test the freshness of messages

The following flags are used in the protocols indicating the intended purpose of the messages transferred in the protocol steps-

$f_{frsk}$ : flag for session key request

$f_{ssk}$ : flag for shared session key

$f_{csk}$ : flag for confirmation of session key.

PROTOCOL 1 : Signed Session Key

Summary:

A reviewer $A_i$ requests $N$ for a session key, $N$ sends a session key $k_{i,N}$ to $A_i$ securely, confirms the key and collects a signed copy of $k_{i,N}$ from $A_i$, signed by $A_i$ using its secret key $s_{A_i}$.

Assumptions

-N also serves as session key generator and distributor.

-$A_i$ and $N$ have their own certificates. Additionally $A_i$ has the certificate of
$N$ and $N$ knows the certificate of $A_i$.

- $A_i$ synchronizes its clock with that of $N$ at the initial start up.

**Protocol Steps**

1. $A_i \rightarrow N : \{f_{rsk}, \{A_i, r_1, t_1\}_{s_{A_i}}\}_{p_N}$
2. $N \rightarrow A_i : \{f_{ssk}, \{N, \{r_2\}_{k_{i,N}}, k_{i,N}, r_1, t_2\}_{s_N}\}_{p_{A_i}}$
3. $A_i \rightarrow N : \{f_{csk}, \{A_i, r_2, k_{i,N}, t_3\}_{s_{A_i}}\}_{p_N}$

**Protocol actions at each step**

1. $A_i \rightarrow N : A_i$ requests $N$ for a session key. The request is signed by the secret key $s_{A_i}$ of $A_i$. $N$ can verify the origin of the request. An adversary $T$ cannot masquerade as $A_i$ since $s_{A_i}$ is known only to $A_i$. The signed request is again encrypted with the public key of $N$, $p_{N}$. This encryption provides privacy, so that none other than $N$ can decrypt the request. Additionally, the request also contains a nonce $r_1$ for later reference and a time-stamp $t_1$ so that $N$ can verify the freshness of the request to avoid replay attacks. For the same reason time-stamps are also included in the next two steps.

2. $N \rightarrow A_i : N$ decrypts the request received in step 1, verifies the signature of $A_i$ and the freshness of the request. If it is satisfied it then sends a session key $k_{i,N}$ along with other parameters. The nonce $r_1$ is incorporated to convince $A_i$ that $k_{i,N}$ is in response to the original request in step 1. Another nonce $r_2$, encrypted with $k_{i,N}$ is provided for the confirmation of the key. The key and the parameters are signed
Figure 4.1: Protocol for Signed Session Key
by \( N \) using its secret key \( s_N \), and then encrypted with \( p_A \) for proof of origin and privacy.

3. \( A_i \rightarrow N : \) \( A_i \) decrypts the message received in step 2, verifies the signature of \( N \) and the freshness of the message and then collects the nonce \( r_2 \) by decrypting \( \{r_2\}_{k_i,N} \) with \( k_{i,N} \). \( A_i \) sends a signed copy of the session key along with \( r_2 \). \( N \) verifies the signed copy of \( k_{i,N} \). Incorporation of plaintext \( r_2 \) confirms that \( A_i \) received the correct key \( k_{i,N} \). \( N \) stores a time-stamped and signed (by \( N \)) copy \( \{N, \{A_i, r_2, k_{i,N}, t_3\}_{k_{i,N}}, T_N\}_s_n \) for later reference. \( T_N \) is the time-stamp given by \( N \).

**Result:** Thus at the end of the protocol, \( A_i \) gets a session key \( k_{i,N} \) and \( N \) also gets a signed copy of \( k_{i,N} \) signed by \( A_i \). Unless \( N \) gets a signed copy of \( k_{i,N} \), it will not allow \( A_i \) to use \( k_{i,N} \) in successive transactions with \( N \). Hence step 3 is mandatory for \( A_i \) to use \( k_{i,N} \), otherwise, it will be an invalid key.

### 4.2.3 Production of Cases

In this section we discuss a protocol for production of a case, which is a MPMSD. Production of a case involves case examination and production of a part. The verification of membership of a page to a MPMSD as well as to a context of a DPW is based on the existence of active paths in the Category Graph and it is discussed in Chapter 3.

**Notations**

In addition to the notations used in PROTOCOL 1, the following notations are used in the protocol for production of cases.

\( f_{rdpw} \) : flag for request for a dpw
\( f_{dpw} \): flag for response for a dpw
\( f_{rpage} \): flag for request for a page
\( f_{page} \): flag for response of a page
\( f_{part} \): flag for submission of a part
\( f_{nrs} \): flag for non-repudiation of submission of a part

\( pid \): pageld of a page. Since timestamps are generated by the Arbiter, which also serves as the storage manager, without any loss of generality, we can use the timestamp associated with a page as unique pageld.

\( H(k, P) \): a collision intractable one-way keyed hash function [26] which produces a unique fixed length message digest of a page \( P \) using \( k \) as an input key to the hashing function.

\( P_{cw} \): context page of \( w \)
\( P_{log} \): a log page containing the list of pending cases to be reviewed by \( A_t \).
\( P_x \): a page of category \( x \). \( x \) may be case, part

**PROTOCOL 2 : Case Production Protocol**

**Summary :**
A reviewer selects a dpw. The arbiter sends the context page of the DPW and the pending cases to the reviewer. The reviewer selects a case and the arbiter sends the case. The reviewer composes a page of category part as her own comment and submits it to the arbiter, who plugs it to the case as a new part and to other relevant pages based on the values of the attributes of the profile of the new part. The arbiter now plugs the case to the log page of the next reviewer and also unplugs the case from the previous reviewer. During the composition of the page the reviewer may peruse previous parts of the case and several pages rooted to the context and cite some of them in
the page under composition at different points of references as a part of case examination.

Assumptions:
- $A_i$ and $N$ have set up a signed session key $k_{i,N}$ using PROTOCOL 1.

Protocol Steps

1. $A_i \rightarrow N : \{f_{rdpw}, A_i, N, r_1, t_1, w\}_{k_{i,N}}$
2. $N \rightarrow A_i : \{f_{dpw}, N, A_i, t_2, P_{cw}, P_{log}\}_{k_{i,N}}$
3. $A_i \rightarrow N : \{f_{rpage}, A_i, N, r_2, t_3, pid\}_{k_{i,N}}$
4. $N \rightarrow A_i : \{f_{page}, N, A_i, r_2, t_4, P_{case}\}_{k_{i,N}}$
5. $A_i \rightarrow N : \{f_{rpage}, A_i, N, r_3, t_5, pid\}_{k_{i,N}}$
6. $N \rightarrow A_i : \{f_{page}, N, A_i, r_3, t_6, P_{part}\}_{k_{i,N}}$
7. $A_i \rightarrow N : \{f_{rpage}, A_i, N, r_4, t_7, pid\}_{k_{i,N}}$
8. $N \rightarrow A_i : \{f_{page}, N, A_i, r_4, t_8, P_{ref}\}_{k_{i,N}}$
9. $A_i \rightarrow N : \{f_{part}, A_i, N, r_5, t_9, pid, P_{part}, H(k_{i,N}, P_{part}), A_{i+1}\}_{k_{i,N}}$
10. $N \rightarrow A_i : \{f_{nrs}, N, A_i, r_5, t_{10}, \{pid\}_{s,N}\}_{k_{i,N}}$

Protocol actions at each step

1. $A_i$ requests for a dpw $w$, to $N$. The proof of origin of the message, freshness etc are verifiable as in PROTOCOL 1.
Figure 4.2: Protocol for Case Production
2. If $A_i$ is an authorized reviewer of the dpw $w$ then the context page $P_w$ of $w$ and the log page $P_{log}$ are sent by $N$ to $A_i$.

3. $A_i$ requests $N$ for a case $P_{case}$ with pageId $pid$ for review.

4. If there exists a path from $P_{case}$ to $P_{log}$ then $A_i$ is a valid reviewer of the case and $P_{case}$ is sent by $N$ to $A_i$.

5. $A_i$ requests $N$ for a part of a case $P_{part}$ with pageId $pid$ for review.

6. If there exists a path from $P_{part}$ to $P_{log}$ then $A_i$ is a valid reviewer of the case and $P_{part}$ is sent by $N$ to $A_i$.

7. $A_i$ requests $N$ for a reference page $P_{ref}$ with pageId $pid$ for perusal during case examination. $P_{ref}$ belongs to the context $P_w$ obtained in step 2.

8. If there exists a path from $P_{ref}$ to $P_w$ then $P_{ref}$ is sent by $N$ to $A_i$. Steps 7 and 8 may be repeated for many times.

9. $A_i$ produces the new part $P_{part}$ as comments on the case. $A_i$ may mark $P_{ref}$ as a citation in $P_{part}$. $A_i$ generates her signature $H(k_i,N,P_{part})$ after completion of composition, and marks $A_{i+1}$ as the next reviewer. If the next reviewer is not explicitly given then the default next reviewer is automatically selected. Finally, $A_i$ submits the new signed page to $N$ as a part to be plugged to the case $P_{case}$, whose pageId $pid$ is obtained in step 3.

10. $N \rightarrow A_i$: $N$ records the time of receipt, $t_r$. If there exists a path from $P_{case}$, whose pageId is $pid$, to $P_{log}$, and it is not unplugged and $P_{case}$ is in active state then $A_i$ is the current reviewer of the case. It then verifies the signature of $A_i$ on $P_{part}$ by recomputing $H(k_i,N,P_{part})$ and
then comparing with the value received from $A_i$ in step 9. If the signature matches then $P_{part}$ is registered with pageId $pnd = t_r$. $N$ plugs $P_{part}$ to $P_{case}$. All citations marked in $P_{part}$ are also plugged by $N$ with $t_r$ as the time of plugging.

Issues of part integrity and the reuse of parts does not arise since the parts are added to the case by the arbiter. Moreover the flow is controlled by the arbiter, hence the access right issue is also automatically addressed. $N$ generates the evidence of non-repudiation of submitting, by encrypting $pnd$ with the secret key of $N$, of $P_{part}$, stores the original copy of it and then sends a verbatim copy to $A_i$. Since $N$ is trusted, if a dispute arises, the original copy available with $N$ will be accepted as the only valid proof. The verbatim copy is available to $A_i$ is for information only.

**Result:** At the end of the protocol a MPMSD is produced as a case. Once it is closed by the last reviewer, it becomes a precedent, goes into in reference state and is automatically plugged by $N$ to $P_{cw}$.

In this protocol we have not attempted to provide mandatory proof of receipt, as we do not think it is so important in our problem. Of course, if required then mandatory proof of receipt can be ensured by redesigning the protocol in the line of the fair non-repudiation protocol discussed in [44].

In the protocol, $N$ has multiple roles of a trusted third party, an arbiter in case of disputes, the co-ordinator of the flow of documents and the storage agent of the signed documents. This centralization has enabled us to use shared secret keys and this has brought in efficiency. Session keys will have to be changed periodically to prevent security attacks. However, $A_i$ presents a signed (using a public key based scheme) version of the session key being used to $N$ who stores it with a time-stamp. If $A_i$'s secret key of the public-
key scheme is later compromised, $A_i$ cannot repudiate the signature on the session key as $N$ will certify that it was signed before the time of compromise. In case of a dispute, $N$ will produce all the messages received from $A_i$ using the shared session key and since $N$ is trusted, this evidence will be the basis of resolving the dispute. It is easy to see that all the security requirements enumerated in section 1.8.2 and 1.8.3 are met. For the sake of brevity we do not examine these requirements separately.

4.3 Discussion

In this chapter a security framework has been presented to study the security of the DPW problem. A protocol for secure production of MPMSDs using a central arbiter is also presented. This protocol addresses all the security issues particular to the production of MPMSDs. Although, a formal proof is not provided, it should be clear from the above discussion that no protocol can be designed without a central TTP component to address the security issues of production of MPMSDs.