Security Modeling for Protecting Electronic Patients’ Consent

Kosmas Kapis and Emmanuel Damas
University of Dar es Salaam, Dar es Salaam, Tanzania
E-mail: kkapis@gmail.com
E-mail: edamas83@gmail.com

ABSTRACT

The adoption of Health Information System (HIS) has emerged as a significant element in the healthcare domain. HIS comprises of Electronic Patient Records (EPR) whose confidentiality is crucial. Patients' consent to EPR access is needed for patients' privacy to be achieved. Research studies have been conducted on Consent Management System (CMS) adoption and integration with HIS as the way to manage EPR access in HIS. However, majority of them provides inadequate security mechanisms to protect the patients’ consent. Attackers could launch attacks such as tampering, repudiation and information disclosure attacks against patients’ consent. These threats could eradicate the relevance of patients’ consent in protecting patient’s privacy. Better security mechanisms should be adopted in order to improve security state of patients’ consent. This study has developed a security model to protect patients’ consent. The developed model has improved patients’ consent security significantly compared to other studies in the reviewed literature.

KEYWORDS

Patient, Confidentiality, Privacy, Consent, Security

1 INTRODUCTION

Rapid advancement in technology has increased adoption of Health Information System (HIS) in the health sector. Cloud computing, web applications, mobile computing and advanced database technologies have significantly accelerated this rate of adoption. HIS has risen as an important element in health care domain. HIS comprises of administrative and Electronic Patient Records (EPR) information. Administrative information is non-patient medical records for example drugs purchase cost [1]. On the other hand, EPR are defined as patient’s electronic medical records [2]. Stakeholders in health care need access to information in HIS. For example decision makers may use EPR stored in HIS to perform predictive data analysis that can help in decision making and insights provision [1].

Despite the fact that the accessibility of EPR from anywhere and anytime by health care stakeholders improves efficiency and collaboration, it also poses a risk of disclosing patients’ EPR without their consent. Consent Management System (CMS) has emerged as the way to give patients’ means to grant EPR access to individuals of their choice through consent creation. Consent refers to the explicit granting or preventing access to specified EPR [3]. It may contain patient’s identifier, health care worker’s identifier and access status. CMS is integrated with HIS in such a way that those who have not been given consent cannot access patients’ EPR. This implies that patients’ consent also needs security protection, for prevention of data breach on EPR to be achieved. As an example, in 2015 University of California Irvine medical center reported a data breach of about 4,859 patients’ medical records. For four years the hospital employee was accessing patients’ medical records without their consent [4].

Protecting patients consent in the health sector will enhance patients’ privacy [3]. This can be achieved by developing security mechanisms that protects patients’ consent at rest and in transit. Consent Management System (CMS) which manages patients’ consent allows patients to permit or deny access of their medical records to particular individuals. Research studies have been conducted on CMS adoption and integration with HIS as the way to control EPR access in HIS. However, most of them provide inadequate security mechanisms to protect the patients’ consent in normal and emergence situations. This leave patients’ consent vulnerable to various attacks and hence, it may jeopardize the privacy of EPR in the Health Information System (HIS).

Health Service Providers (HSP) are required to implement a mechanism that protects the patients' privacy [5]. Most of existing HSP are either using paper based patients' consent information or have integrated their HIS with electronic CMS whose security implementation is weak [6]. According to Olca and Can [5], EPR for an individual can only be accessed in cases required by the law or by the individual explicit consent. This indicates that EPR protection is important as well as patients' consent protection [7]. In this study, a security model to protect patients' consent information has been developed. This model aimed at preventing: processing of non-authentic requests, unauthorized modification of the patient's consent, repudiation of patient's activities, unauthorized access to patients' consent, denial of service and elevation of privileges in the CMS. Integrating this model with HIS and CMS avails patients' rights to choose who can have access to their EPR. This research has developed a security model which protects patients' consent when they are in storage and in transit.

2 RELATED WORKS

Electronic consent model is an important security mechanism to achieve patient's privacy [5]. According to Olca and Can [5] patients' must maintain rights to protect their medical records from unauthorized access. They mentioned the significance of consent management for patients' medical records. Furthermore, their work suggests establishment of the patients' consent policy and conduction of researches regarding frameworks or/and algorithms that govern electronic consent. However, it does not explain how to implement and protect an electronic consent.

In another study, HIPAAT [3] has developed a fine grained consent management model. The study designed a consent model using Service Oriented Architecture (SOA). SOA is a technique for developing system whose services support reusability functionalities via well-defined interfaces [8]. The model were designed to be integrated with web based HIS application to control access to electronic patients' records. This model consists of the following elements: consent policies, consent management service, consent validation service, consent enforcement point and audit service. eXtensible Access Control Markup Language (XACML) and Health Level 7 (HL7) are the standards that have been used by this model. XACML is a security policy language that is used to enforce components of the organization information security policy across organization information systems [9]. On the other hand HL 7 provides a standard way for exchange, integration, sharing and retrieval of electronic health information [10]. This model has focused mostly on having electronic consent. It has not discussed security techniques such as encryption which may be used to protect an electronic consent. In addition, the model allows health care provider to override the restrictions which may result in unauthorized access to patients' medical records.

Amoussou [6] is another CMS model that uses Domain Driven Design (DDD). DDD uses both developer and domain expertise together to solve domain specific problem [11]. Furthermore, the model used both SOAP-based and RESTFUL web service interfaces as data exchange mechanism. SOAP-based web service uses XML format only while RESTful web service can process different formats such as XML, plain text and JSON [12]. The model aims to achieve number of goals. First, helping patients to have fine-grained control over their medical records, control to share with whom and for what purpose. Second, smoothing the process of a patient to access, track, update and revoke their consents remotely. This model has just used XACML the same way as it has been used by [3]. From security point of view, this model has implemented spring security framework. Authorization and authentication are some of security features supported by the framework [13]. Although this model integrated spring security framework in its implementation, it has not stated whether patients' consent information is encrypted or not in the storage. The model has also not discussed on how to handle emergency access to patients' medical records.

Both HIPAAT [3] and Amoussou [6] have used the CMS model as depicted in Figure 1. The model has established EPR retrieval procedures from HIS. First, a Health Care Worker (HCW) makes a request to access EPR from HIS. Second, HIS checks from CMS whether HCW is allowed to access those particular EPR. The check is done against privacy preferences that were created by a patient in CMS. Patients are able to create, edit, delete and update their privacy preferences in CMS. Third, CMS responds with access status (permit or deny). If permit is returned as access status HCW is allowed to access those EPR otherwise access is denied.

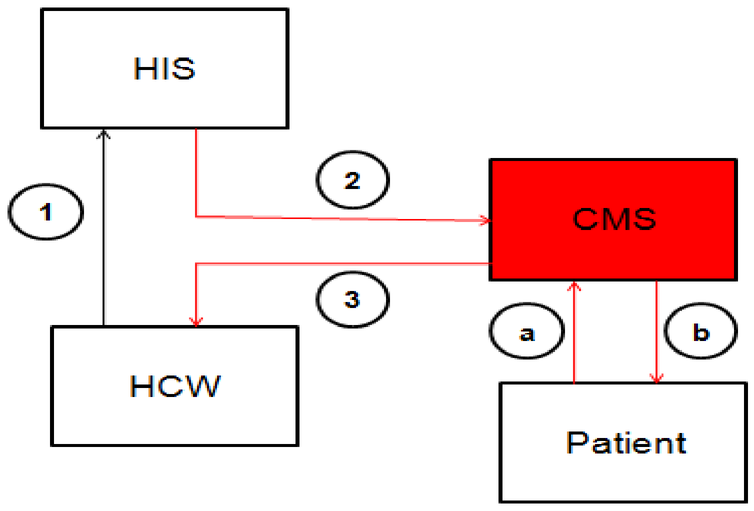


Figure 1. Consent Management System Model

As depicted in Figure 1, these studies on patients' consent have focused on developing and integrating CMS with HIS. They have not established more efficient ways of protecting patients' consent when it is in transit or rest. Therefore, they are leaving a room for patients' consent to suffer security attacks which may cause patients' privacy to be compromised.

3 METHODOLOGY

This study followed security modeling procedures as depicted in Figure 2. Requirement gathering and analysis for HIS and CMS was the first step in the modeling process. HIS and CMS workflows and integration were also analyzed to finalize system identification phase of the security modeling procedure. The Next step involved identification of threats that may affect patients consent. Microsoft threat modeling was used in this research. It has been used by Microsoft in the development process of their software products for more than 12 years [14]. Attack modeling followed threat identification process. Attack trees were constructed and analyzed to determine different ways that patients' consent can be compromised. Each discovered attack has an impact on patients' consent; therefore evaluations of the effects caused by the identified attacks were conducted.

In addition, Damage potential, Reproducibility, Exploitability, Affected users and Discoverability (DREAD) has been used to perform security risk analysis on patient's consent. DREAD provides risk rating for each threat by considering damage of asset, reproducibility of attack, exploitability of an asset, affected users and discoverability of vulnerabilities. It is important to perform risk assessment since it enables application of the cost benefit risk mitigation techniques. Thereafter, the model design was constructed in order to protect patients consent against the identified threats. Cryptography was used in the modeling process to provide security services. AES-256, RSA-2048 and SHA-256 cryptographic algorithms were used in this study during the implementation phase.

The developed model has been integrated with Health Information System (HIS) and Consent

Management System (CMS). OpenMRS and Consent2Share were used as HIS and CMS respectively. Sampled EPR were collected and anonymized from University of Dar es Salaam Health Medical Center. Purpose sampling was used as a sampling technique. This technique aims at particular attributes of a population that are of interest [15]. Patient's age is an important attribute to consider in this study due to its involvement in determining the cryptographic key complexity. Therefore, purpose sampling was more appropriate for this study. Age ranges from 0-30, 31-60 and above 60 were considered. A single patient was selected from each range. Document review was also used as a data collection method for EPRs collection. For patients' consent, simulated data were generated based on the CMS requirements. Collected data were then prepared and entered in the Open MRS and Consent2Share data models. Scenario based attacks like those in the real world were simulated to check whether the developed model can protect patients' consent in the CMS. Various forms of attacks were launched against CMS. The results of these attacks were compared against expected outputs.

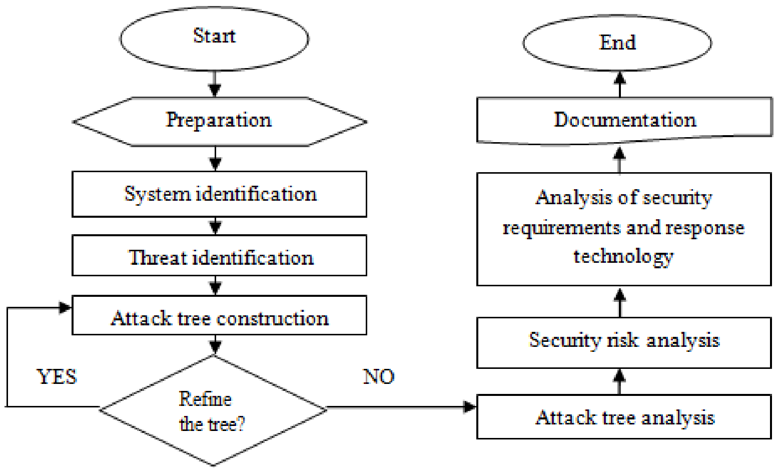


Figure 2. Security Modeling Procedure (Source: Jung-Sook et al., 2015)

the first step in the security modeling procedure as shown in Figure 2. HIS is among the assets under study when modeling Consent Management Security Model (CMSM). CMS is another asset under consideration. EPR stored in HIS makes it an important and sensitive asset. According to Rada [16], HIS comprises of several datasets that support the demands of Hospital Service Provider (HSP), clinicians, patients and policy makers. Locatelli, Restifo, Gastaldi, and Corso, [17] have listed requirements that have to be present in HIS regarding patients' medical records. Table 1 depicts some of the requirements of this study, which were considered in this research.

Table 1. HIS Requirements

No	Function
1	HIS shall support patient admission
2	HIS shall store patient diagnosis tests and results
3	HIS shall be able to record and retrieve patient medical history
4	HIS shall store patient visits

In addition to HIS, CMS which stores patients' consent information is another asset in designing CMSM. CMS aims to preserve the patients' privacy through access control mechanism [18].

Protecting patients' consent information which is kept in CMS is the main focus of this study. This goal has been achieved by developing CMSM. CMS has to provide a number of functionalities regarding patients' consent management.

Ruan [18] has listed CMS requirements as depicted in Table 2. These requirements were customized in the Consent2Share CMS and integrated with CMSM.

Table 2. CMS Requirements

No	Function
1	CMS shall allow patients to choose the subjects (role, individual , system and organizations) to whom the consent is given
2	CMS shall allow patients to select type of medical data for protection
3	CMS shall allow patients to define access rights allowed or prohibited on their medical data
4	CMS shall allow patients to choose context regarding access to their medical data
5	CMS shall allow patients to specify period of validity for their consents
6	CMS shall store patient’s consent and allow external entities such as HIS and patients to access them

threat identification techniques were applied. Threat identification as an important step in modeling CMSM, it reveals potential risks that may have an impact on patients’ consent. Clearly, threat identification aims to identify and mitigate threats before they become exploitable vulnerabilities in an information asset. Before starting identifying threats to a system, it is crucial to understand how data flows in the system. Understanding data flow in CMS requires a tool that shows how data flows in, within and out. Data Flow Diagram (DFD) was used to show how data enters, transverses and leaves CMS. Figure 3 represents a level 1 DFD for CMS.

After developing the DFD for CMS, threat identification was conducted. STRIDE was applied as the identification technique. STRIDE, which denotes Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege, is a threat modeling technique which has been used at Microsoft for several years.

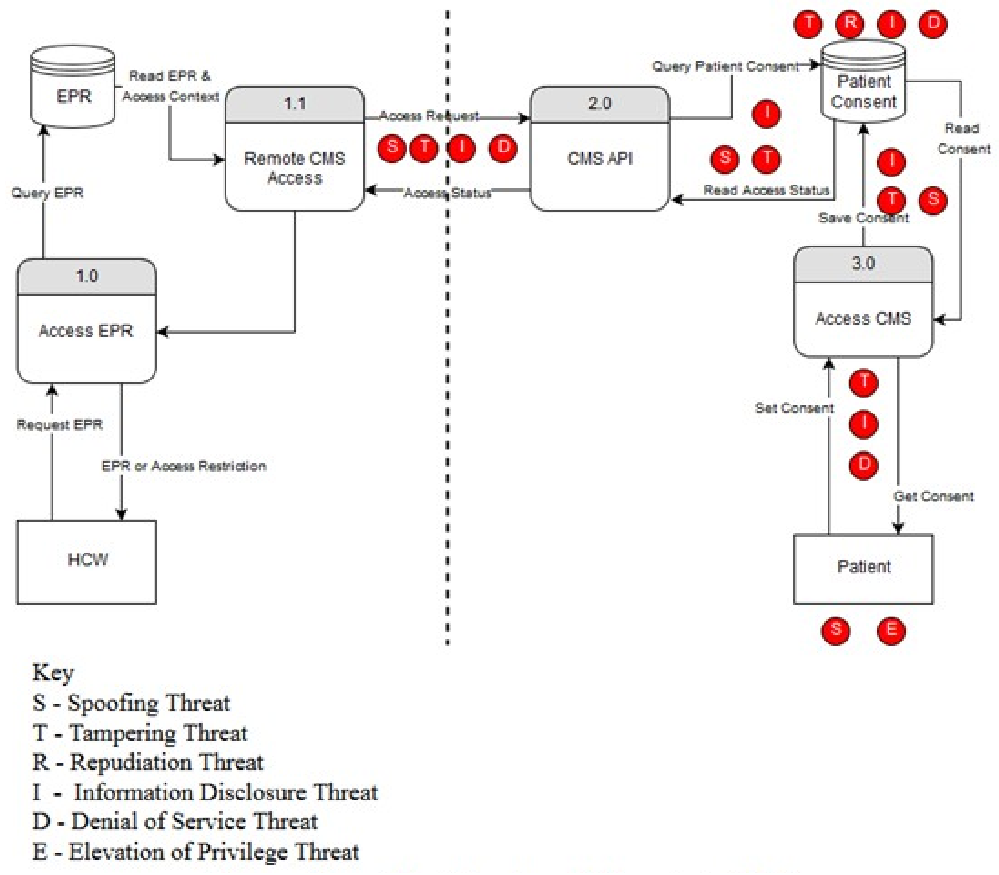


Figure 3. Identification of Threats to CMS

Table 3. Established Threats to CMS

No	Threat	Impact on CMS	Security Service Required
1	Spoofing	Unauthorized user imitates a legitimate CMS user	Authentication
2	Tampering	Unauthorized modification of patients' consent or CMS source code	Integrity
3	Repudiation	Claiming to have not conducted an action in CMS	Non-Repudiation
4	Information Disclosure	Unauthorized access to patients' consent or CMS source code	Confidentiality
5	Denial of Service	Deny legitimate users to access patients' consent in CMS	Availability
6	Elevation of Privilege	A legitimate user or system performing unauthorized actions on patients' consent	Authorization

It involves a manual review of DFD to determine the possibility of a threat to occur. As can be seen from Figure 3, threats were noted between CMS application side and the storage and between CMS and external parties such as HIS and patient. In addition, threats were also seen possible for patients' consent when they are in storage. This threat modeling shows how an adversary can pose risks to CMS by supplying it with malicious information or by interacting with it. Table 3 shows the potential threats that may affect patients' consent, their impacts and security services required for mitigation. Up to this point, a baseline for developing a mechanism for securing a CMS was set.
Hence, security model was designed to protect patients' consent against these threats

Attack tree construction and analysis procedures were applied immediately after identification of threats. This is due to the fact that identification of threats sets a foundation of attack trees construction and analysis. Attack trees become more comprehensive when more scenarios are considered [19]. Attack trees have been used to understand the possible ways that an adversary could use to launch attacks against patients' consent. Through attack trees relevant goals that an attacker wants to achieve on CMS were identified and analyzed. Figure 4 shows confidentiality attack trees on patients' consent as an example of attack tree which was developed in this study.

At this stage in the modeling process, a list of threats affecting patients' consent has been identified including attack scenarios. Now, the next step was to assign a risk rating to each threat based on the risks they pose. DREAD risk rating technique was used for this task. Table 4 shows security risk analysis for patients' consent management. Equal opportunity in exploiting patients' consent was among the factors considered in assigning risk rating for each threat. Therefore, designed model consist of protection mechanisms against all established threats.

Table 4. Established Threats to CMS

THREAT	D	R	E	A	D	Overall Risk Rating	
Spoofing	6	6	5	10	10	7.4	H
Tampering	10	10	7	5	10	8.4	H
Repudiation	8	8	10	5	10	8.2	H
Information Disclosure	10	10	5	5	10	8.0	H
Denial of Service	10	10	5	10	10	9.0	H
Elevation of Privilege	9	9	10	10	10	9.6	H

Scale was categorized into three levels; low medium and high. 0 to 4 for low risk (L), 5 to 6 for medium risk (M) and 7 to 10 high risk (H)

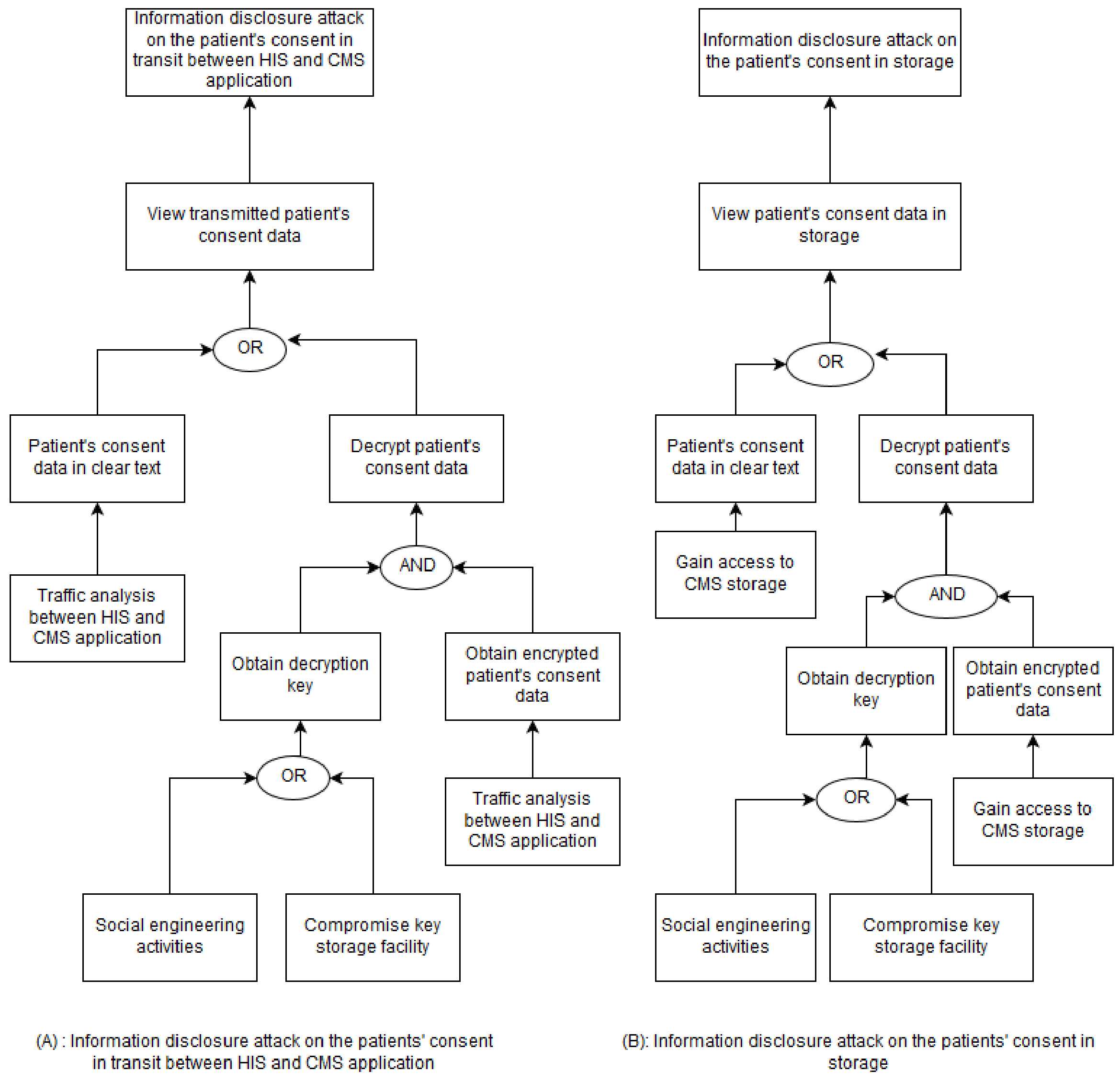

Information disclosure attack on the patient's consent in transit between HIS and CMS application
View transmitted patient's consent data
OR
Patient's consent data in clear text
Decrypt patient's consent data
Traffic analysis between HIS and CMS application
AND
Obtain decryption key
Obtain encrypted patient's consent data
OR
Traffic analysis between HIS and CMS application
Social engineering activities
Compromise key storage facility
(A) : Information disclosure attack on the patients' consent in transit between HIS and CMS application
Information disclosure attack on the patient's consent in storage
View patient's consent data in storage
OR
Patient's consent data in clear text
Decrypt patient's consent data
Gain access to CMS storage
AND
Obtain decryption key
Obtain encrypted patient's consent data
OR
Gain access to CMS storage
Social engineering activities
Compromise key storage facility
(B): Information disclosure attack on the patients' consent in storage

Figure 4. Confidentiality attack trees

With the completion of establishment of threats and their risk ratings, the model for protecting patients' consent was developed. It has four parts which are key management, CMS client and CMS block, HIS and CMS block and emergency situation. It is also important to note the notations which were used during the design. Consider A_{BC} as a notation where by A is an item with state B at context C. Another notation is A_B where by A is an item at context B. Figure 5 show high level interaction between the model blocks. The interactions involve a key request and provision. Key Management System (KMS) is responsible to manage and provide symmetric and asymmetric keys. These keys are used along with the security services such as confidentiality, integrity and availability which are provided by the security services engine. These security services are used by CMS Client, CMS and HIS to provide security to data that is in transit and storage.

Cryptographic keys have to be kept in a safe place after being generated. This may involve storing a key in a special hardware such as Hardware Storage Module (HSM). HSM provides both physical and software security of cryptographic keys. Another way of protecting keys is to store them in an encrypted form in an operating system file or database. In this research keys were stored in encrypted form in database and operating system file. However, for production environment, it is more secure to store the keys in HSM especially for a key such as the master key. The following shows key generation procedures in this model.

i) Key Management System (KMS) generates master key, K_K by using the key generation function, *G* and store K_K in Key Management System Storage (KMSS).
$K_K = G(K)$
$KMS \xrightarrow{K_K} KMSS$

ii) KMS generates *KEK*, encrypts it by using encryption function, *E* and master key, *K* and store encrypted *KEK* in KMSS.
$KEK_K = G(KEK)$
$KEK_{EK} = E(KEK_K,K)$
$KMS \xrightarrow{KEK_{EK}} KMSS$

iii) KMS generates Hashing Key, *HK* and Hashing Key Hash, *HKH*. KMS encrypts *HK* with *KEK*. Both *HK* and *HKH* are stored in KMSS. Start and end age limit, start date, end date and key purpose are also included. This is repeated based on the number of age ranges in the list, *L*. Age range was included to enhance key complexity.
For each patient age range
{
$HK_K = G(HK)$
$L[HK_{EK}] = E(L[HK_K], KEK_K)$

$L[HKH_{HK}] = H(L[HK_K], KEK_K)$
}
$KMS \xrightarrow{L[HK_{EK}],\ L[HKH_{HK}]} KMSS$

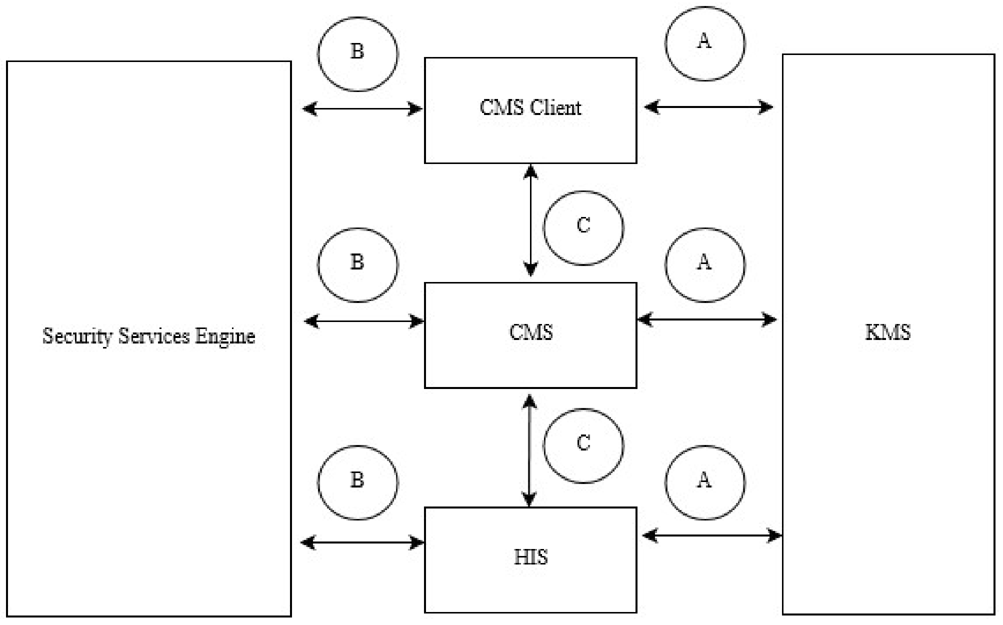
iv) KMS generates Encryption/Decryption Key (Symmetric Key, *SK*). KMS encrypts *SK* with *KEK* and store in KMSS.
For each patient age range
{
$SK_K = G(SK)$
$L[SK_{EK}] = E(L[SK_K], KEK_K)$
$L[SKH_{HK}] = H(L[SK_K], KEK_K)$
}
$KMS \xrightarrow{L[SK_{EK}],\ L[SKH_{HK}]} KMSS$

v) CMS/HIS/CMS client generates Asymmetric Keys, *ASK*. CMS/HIS/CMS client sends the public key to KMS storage and store private key in CMS/HIS/CMS client.
$ASK = G(P,PU)$

CMS client and CMS block is among the components in the design of the model. CMS client is the part of CMS which enables CMS users to interact with the application. It gives patients the ability to create, read, update and disable their consent. Subsequently adversary can take advantage of the flow of information between CMS client and CMS including the information on CMS storage. Due to this, the model included this workflow in the design of protection mechanism for patients' consent. The following shows procedures in the model that protect patients' consent in CMS client and CMS block. These procedures were also modified to provide security requirements for HIS and CMS block and emergency situation.

1. Patient requests CMS client access to CMS by providing Username, *U* and Password, *PA*. Public Key *(PU)*, Private Key *(P)*, encryption function (*E)*, decryption function *(D)* and hashing function *(H)* are used to provide security services at both sides.
$CMSU \xrightarrow{U_L,PA_L} CMSC$
$U_{EL} = E(U_L,PU_L)$
$PA_{EL} = E(PA_L,PU_L)$
$CMSC \xrightarrow{U_{EL},PA_{EL}} CMS$
$U_{DC} = D(U_{EC},P_C)$
$PA_{DC} = D(PA_{EC},P_C)$
$U_{HC} = H(U_{DC}, AK)$
$PA_{HC} = H(PA_{DC}, AK)$
If ($U_{HC} == U_{HS}$ AND $PA_{HC} = PA_{HS}$)
{

Patient Age, PA and User Unique Identifier, UUI are retrieved from CMS storage
$UUI_{EC} = E(UUI_C, PU_C)$



Key
A- Key Request And Provision
B- Security Service Request and Provision
C- Secured Data Transmission

Figure 5. Model interactions block diagram

$CMS \xrightarrow{PAG, UUI_{EC}} CMSC$
$CMSC \xleftarrow{P_L} CMSCS$
$UUI_{DL} = D(UUI_{EL}, P_L)$
}
Else
{
Stop processing the request
}

2. Patient requests CMS client to perform data operation(s), *DO*. By data operation(s), it means data creation, view, update or status change.
   $CMSU \xrightarrow{DO} CMSC$
3. KEK is retrieved from storage. KEK is used to encrypt and decrypt hashing and encryption keys.
   $KMS \xleftarrow{KEK_{EK}} KMSS$
   $KEK_{DK} = D(KEK_{EK}, K)$
   $KEK_{EK} = E(KEK_{DK}, PU_K)$
   $CMSC \xleftarrow{KEK_{EL}} KMS$
   $KEK_{DL} = D(KEK_{EL}, P_L)$
4. CMS client generates a nonce, N_L by us- e generator, N that will be incorporated with a request to CMS. For each request there will be a new nonce.
   $N_L = N(n)$
5. Hashing Key *(HK)*, Hashing Key Hash *(HKH)*, Encryption Key *(EK)* and Encryption Key Hash *(EKH)* are retrieved from KMS. Keys are retrieved based on Validity Period *(VP)*, Patient's Age *(PAG)* and Key Purpose, *(KP)*.
   $CMSC \xrightarrow{VP, PAG, KP} KMS$
   $CMSC \xleftarrow{HK_{EL}} KMS$
   $CMSC \xleftarrow{HKH_L} KMS$
   $CMSC \xleftarrow{EK_{EL}} KMS$
   $CMSC \xleftarrow{EKH_L} KMS$
6. Retrieved keys are decrypted using the KEK and decryption function, *D*
   $HK_{DL} = D(HK_{EL}, KEK_{DL})$

$EK_{DL} = D(EK_{EL}, KEK_{DL})$

7. Integrity check is performed on decrypted keys. They are compared against hashes stored during their creation.

   If ($HKH_L == H(HK_{DL}, KEK_{DL})$ AND

   $EKH_L == H(EK_{DL}, KEK_{DL})$)

```
{
Proceed with the next step
}
Else
{
Stop processing the request
}
```

8. Data, D_L from CMS client to CMS is hashed by using a hash function, keys and nonce.

```
For each $L[D_L]$
{
$L[D_{HL}] = H(L[D_L], HK_{DL}, N_L)$
}
```

9. Data from CMS client to CMS together with nonce are encrypted using encryption function and keys.

```
For each $L[D_L]$
{
$L[D_{EL}] = E(L[D_L], EK_{DL})$
}
```

   $N_{EL} = E(N_L, EK_{DL})$

   $U_{EL} = E(U_L, EK_{DL})$

   $PA_{EL} = E(PA_L, EK_{DL})$

   $T_{EL} = E(T_L, EK_{DL})$

   $UUI_{EL} = E(UUI_{DL}, EK_{DL})$

10. Hashed and encrypted data from CMS client are sent to CMS together with encrypted username, password, nonce and timestamp.

$$CMSC \xrightarrow{L[D_{HL}], L[D_{EL}], N_{EL}, U_{EL}, PA_{EL}, T_{EL}, UUI_{EL}, PAG, SDI}$$

11. CMS retrieved hashing and decryption keys from KMS.

$$KMS \xleftarrow{KEK_{EK}} KMSS$$

$$KEK_{DK} = D(KEK_{EK}, K)$$

$$KEK_{EK} = E(KEK_{DK}, PU_K)$$

$$CMS \xleftarrow{KEK_{EC}} KMS$$

$$CMS \overset{P_C}{\leftarrow} CMSKS$$

$$KEK_{DC} = D(KEK_{EC}, P_C)$$

12. CMS retrieved *HK* and *DK* and their hashes from KMS.

$$CMS \xrightarrow{VP, PAG, KP} KMS$$

$$CMS \xleftarrow{HK_{EC}} KMS$$

$$CMS \xleftarrow{HKH_C} KMS$$

$$CMS \xleftarrow{DK_{EC}} KMS$$

$$CMS \xleftarrow{DKH_C} KMS$$

13. Retrieved keys are decrypted by using KEK, then, they are compared against their hashes to determine any integrity violation

    $HK_{DC} = D(HK_{EC}, KEK_{DC})$

    $DK_{DC} = D(DK_{EC}, KEK_{DC})$

14. CMS decrypts HIS patient's identifier *(UUI)*, Nonce *(N)*, Username *(U)*, Password *(PA)* and Timestamp *(T)* by using the decryption function and keys.

```
For each $L[D_{EC}]$
{
$L[D_{DC}] = D(L[D_{EC}], DK_{DC})$
}
```

    $N_{DC} = D(N_{EC}, DK_{DC})$

    $U_{DC} = D(U_{EC}, DK_{DC})$

    $PA_{DC} = D(PA_{EC}, DK_{DC})$

    $T_{DC} = D(T_{EC}, DK_{DC})$

    $UUI_{DC} = D(UUI_{EC}, DK_{DC})$

15. Decrypted data from CMS client is hashed using hash function and keys then is compared to hashed data from CMS client. If the output of the comparison is different, then no further processing of the request will proceed. In addition, data validation for selected fields is performed by validation function, *V* at this stage in order to protect the CMS application from attacks. CMS verifies authenticity of the request from CMS client. It checks whether the following are correct; username, password, CMS client identification number, nonce and timestamp validity window. Any mismatch will stop further processing of the request.

    VAS = FALSE

```
$CMS \xrightarrow{UUI_{DC}, T_{DC}, SDI} CMSAL$
If ($V(U_{DC}, PA_{DC}, SDI)$ AND $N(t)_{DC} != N(t+1)_{DC}$ AND $T(t)_{DC} < T(t+1)_{DC}$)
{
  VAS = TRUE
}
HMS = FALSE
For each $L[D_{DC}]$
{
  If ($L[D_{HL}] != H(L[D_{DC}], HK_{DC}, N_{DC})$)
  {
    HMS = TRUE
    Exit Loop
  }
}
If (VAS == TRUE AND HMS == FALSE)
{
  Save the new nonce and timestamp information
  Proceed with the next step
}
Else
{
  Stop processing the request
}
```

16. For the communication between CMS and CMS storage, CMS performs either of the following
    i. CMS encrypts data before sending to CMS storage if it is a saving or updating operation. At this stage only patient's consent (Access Status), *AS* is encrypted, but any other information may be encrypted when there is a need to do so. Moreover, hashing of data is done at this stage if needed.
       $AS_{EC} = E(AS_{DC}, EK_{DC})$
       $CMS \xrightarrow{AS_{EC}} CMSS$
    ii. CMS decrypts data received from CMS storage if it is a read operation.
       $CMS \xleftarrow{AS_{EC}} CMSS$
       $AS_{DC} = D(AS_{EC}, DK_{DC})$
17. CMS uses encryption/decryption keys from KMS which are in memory to encrypt the response from CMS storage before sending to CMS client. In addition, CMS hashes the response with the hash function and keys which are still in memory.

```
$N_C = N(n)$
For each $L[D_{DC}]$
{
  $L[D_{EC}] = E(L[D_{DC}], EK_{DC})$
}
$N_{EC} = E(N_{EC}, EK_{DC})$
$T_{EC} = E(T_{EC}, EK_{DC})$
For each $L[D_{DC}]$
{
  $L[D_{HC}] = H(L[D_{DC}], HK_{DC}, N_C)$
}
```

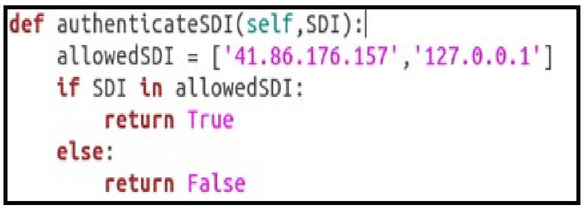
18. Hashed and encrypted CMS response is sent to CMS client. In addition, encrypted nonce, timestamp and server identifier are sent to CMS client.
    $CMSC \xleftarrow{L[D_{HL}], L[D_{EL}], N_{EL}, T_{EL}, SDI} CMS$
19. CMS client decrypts the CMS response, nonce, timestamp and server identifier by using the decryption function and keys. The keys which are used here are those in CMS client memory.

```
For each $L[D_{EL}]$
{
  $L[D_{DL}] = D(L[D_{EL}], DK_{DL})$
}
$N_{DL} = D(N_{EL}, DK_{DL})$
$T_{DL} = D(T_{EL}, DK_{DL})$
```

20. CMS client verifies authenticity of the information from CMS. It checks whether the following are correct; server identifier, nonce and timestamp validity window. Any mismatch will stop further processing of the response. Decrypted CMS response is hashed by using hash function and key, and then is compared to hashed CMS response. The response will not be processed further if there is a discrepancy in the output of the comparison. Data validation is also performed at this stage to protect the CMS client from de-

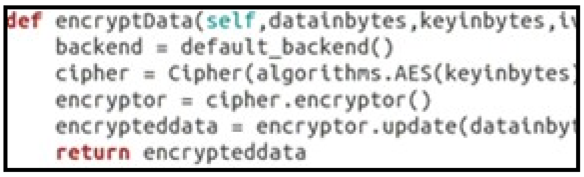
```
nial of service. From this point any further activity may proceed at the CMS client.
VAS = FALSE
If (V(SDI) AND $N(t)_{DL} != N(t+1)_{DL}$ AND $T(t)_{DL} < T(t+1)_{DL}$ )
{
  VAS = TRUE
}
HMS = FALSE
For each $L[D_{DL}]$
{
  If ($L[D_{HL}] != H(L[D_{DL}], HK_{DL}, N_{DL})$)
  {
    HMS = TRUE
    Exit Loop
  }
}
If(VAS == TRUE AND HMS == FALSE)
{
  Save new nonce and timestamp information
  Proceed with the next process
}
Else
{
  Stop processing the request
}
```

The model was implemented by using python programming language. Figure 6 and 7 show examples of the implementation parts of the model.



```
def authenticateSDI(self,SDI):
    allowedSDI = ['41.86.176.157','127.0.0.1']
    if SDI in allowedSDI:
        return True
    else:
        return False
```

Figure 6. Verification of source device identity



```
def encryptData(self,datainbytes,keyinbytes,i
    backend = default_backend()
    cipher = Cipher(algorithms.AES(keyinbytes)
    encryptor = cipher.encryptor()
    encrypteddata = encryptor.update(datainbyt
    return encrypteddata
```

Figure 7. Symmetric encryption implementation

4 FINDINGS AND DISCUSSIONS

During model testing, the key findings were identified from the analysis. The findings revealed provision of authentication, integrity, non-repudiation, confidentiality, availability and authorization services to patients' consent.

Consent Management Security Model (CMSM) enforced Consent Management System (CMS) to process requests from trusted sources only. CMS was configured to accept requests from device with 41.86.176.156 IP address and its own IP address. The CMS client launched a request to CMS with 41.86.176.157 IP address. The request was not successful due to the fact that CMSM did not allow CMS to process any request from such IP address. Therefore, CMSM protects CMS from processing requests from untrusted sources. It is worth to note that other source device identifiers such as Media Access Control (MAC) address could also be used as filtering criterion apart from IP address. In addition, CMS processed request with particular nonce, timestamp and allowed source address only once. That is to say, CMSM did not allow CMS to process the replay request which is among forms of spoofing attacks. Hence, it helps to protect CMS against replay attacks. This was evidenced by repeating sending a request to CMS from CMS client with the previous nonce and timestamp

The model assured the existence of integrity on consent information. Consent was hashed in CMS with 41.86.177.137 IP address before was sent to HIS. Once it reached HIS, consent was decrypted and hashed again, then compared with the hashed consent received from CMS. The output of hashed consent at the HIS was the same as that at CMS. This means CMS has provided assurance on the consent integrity. The same procedure could be applied to verify data integrity residing in CMS storage. In addition, the model provided audit trails that prevent patients from denying activities which they have performed on their consent information. Patient's identifier, HCW identifier, SDI, activity type, activity timestamp and remarks about the activity were captured and stored in the audit trails. This information will be used in case privacy violations occur on patients' consent or EPR to settle disputes. Additionally, the information could be used as a log that provides opportunity to review previous activities which can detect any malicious activity attempts.

Confidentiality on patients' consent was achieved through asymmetric and symmetric encryption. Patient information including her consent transmitted from CMS client to CMS was encrypted using symmetric encryption and a corresponding symmetric key. But, it is important to note that the exchange of symmetric keys between parties was achieved through asymmetric encryption. Furthermore, it was not possible to break encrypted data which were generated by developed model. Generated random keys and initialization vectors were used to try to decrypt cipher text with value772c87f614705e268362. Expected plain text in Hexadecimal format was 323031372d30382d3233. All results did not match the expected result which implies that the model is providing consent confidentiality accordingly.

The model ensured availability of consent upon request. Major attack on availability of a patients' consent is a DOS attack. It comes in various forms, but the one which caused by the supply of unwanted content was considered. Username field in the login form was selected to test the Denial of Service (DOS) attack by supplying it with unwanted characters. The predefined list of unwanted characters can be expanded as much as per the risk appetite level. The same control could be applied to any other form field in the CMS application to provide DOS protection. Each attempt was supplied along with a valid existing password in the storage. All attempts were not successful. The model which is integrated in CMS was able to deny the processing of these attempts before they reached CMS storage. Hence, assurance of availability of a patients' consent up on request was guaranteed. The model was also able to prevent authorized users to escalate privilege. Attempts to login into CMS as a patient by using legitimate doctor's credentials were not successful. This is due to the fact that access to CMS was based on the combination of a user's role and credentials. Moreover, storage of user credentials was segmented within CMS storage. This means if the part which stores patients' credentials has been comprised, then the other parts which keep HCW and administrator's credentials will remain resilient to the attack. The same test was repeated by using patient's credentials against doctor's privileges. It was not successful as the other case.

5 CONCLUSION

Information security plays an important part in protecting patients' consent. In fact, it was found that the developed model utilized cryptographic mechanisms to protect patients' consent against confidentiality, integrity and authentication attacks. Furthermore, the developed model provided; access to Consent Management System (CMS) based on privileges, non repudiation capability for activities performed by patients in CMS and reliable availability of CMS to authentic source of requests. The study provided procedures on how these were achieved. Data was collected from University of Dar Es Salaam health medical center and others were simulated in order to meet requirements for evaluation of the model. Authentication, integrity, confidentiality, availability, non repudiation and authorization were the measuring parameters for this model. The results from this study have been compared with existing research studies and found that the developed model provides better security protection on CMS.

6 REFERENCES

1. WHO. (2008). Health information system. Retrieved from http://goo.gl/oBvzeb.
2. Kapis, K., Lubbe, J., Cartrysse, K., & Gedrojc, B. (2005). Security scheme for electronic patient ' s records incroporating full privacy. Retrieved from http://goo.gl/1b3DeG.
3. Health Information Protection and Associated Technologies. (2009). Implementing consumer privacy preferences in health information exchange (HIE) using service-oriented architecture (SOA). Florida, USA. Retrieved from http://goo.gl/YZJNSw.
4. Identity Theft Resource Center. (2015). Data breach reports. San Diego,CA.

5. Olca, E., & Can, O. (2014). Providing patient rights with consent management. In 2nd international symposium on digital forensics and security (ISDFS) (pp. 1–4). Houston,TX.

6. Amoussou, J. (2013). Consent2Share software architecture. Retrieved from http://goo.gl/1uaA3j.

7. Enrico, C., & Roger, C. (2004). E-consent: the design and implementation of consumer consent mechanism in an electronic environment. Journal of the American Medical Informatics Association, 11(2), 129–140. Retrieved from goo.gl/aUkukm

8. Lewis, G. (2010). Getting started with Service-Oriented Architecture (SOA) terminology. Pittsburgh,USA: Carnegie Mellon University. Retrieved from http://goo.gl/TTFh0b.

9. OASIS. (2013). eXtensible access control markup language version 3.0. Retrieved from http://goo.gl/LsllTM

10. HL7 International CTO. (2014). Introduction to health level seven international. Illinois,USA. Retrieved from https://goo.gl/w1aJzH.

11. Evans, E. (2003). Domain-driven design: Tackling complexity in the heart of software (1st ed.). San Francisco,California: Prentice Hall.

12. Rodriquez, A. (2015). RESTful web services: The basics. Retrieved from http://goo.gl/5RcXw6.

13. Alex, B., Taylor, L., & Winch, R. (2012). Spring security reference. Retrieved from https://goo.gl/xZ6KXw.

14. Microsoft. (2003). Chapter 3 Threat Modeling. Retrieved from https://goo.gl/dE2vFT.

15. Mugo, F. (2002). Sampling in research. Retrieved from http://goo.gl/BQIV6z.

16. Rada, R. (2008). Information systems and healthcare enterprises. Hershey,PA: IGI Publishing. http://doi.org/10.4018/978-1-59904-651-8

17. Locatelli, P., Restifo, N., Gastaldi, L., & Corso, M. (2012). Health care information systems: architectural models and governance. In Innovative Information Systems Modelling Techniques (pp. 73–98). Italy. http://doi.org/10.5772/38212.

18. Ruan, C., & Yeo, S.-S. (2009). Modeling of an intelligent e-consent system in a healthcare domain. Journal of Universal Computer Science, 15(12), 2429–2444.

19. Tentilucci, M., Roberts, N., Kandari, S., Johnson, D., Bogaard, D., Stackpole, B., & Markowsky, G. (2015). Crowdsourcing Computer Security Attack Trees. In 10th annual symposium on information assurance (ASIA '15), June 2-3, 2015, Albany,NY (pp. 19–23). Albany,New York. Retrieved from http://goo.gl/LzxCBuf.