International Journal of Advanced Science and Technology
Vol.76 (2015), pp.9-20
http://dx.doi.org/10.14257/ijast.2015.76.02

Algebraic Specification and Analysis of Health Records

Khair Eddin Sabri

Computer Science Department
King Abdullah II School for Information Technology
The University of Jordan, Amman, Jordan
k.sabri@ju.edu.jo

Abstract

Many organizations are moving their information to the cloud because of the rapid development of cloud computing as a result of the decrease in the running cost of managing information. Among of these organizations are health systems where the privacy of records is their main concern. However, moving the information to the cloud involves some security problems. An example of such problems is that the server that holds the information is a third party and could not be trusted. One way to solve this problem is to have all the records encrypted such that only the allowed users have the keys to decrypt the records. Before the distribution of keys, we should specify policies that state which user holds which key and then prove the correctness of the specification and the satisfaction of predefined security properties. The language used in specification should be flexible to allow specifying combining keys, combining secrets, selecting between keys, and so on.

In this paper, we employ the algebraic structures presented by Sabri and Khedri [1] to specify policies of health records. We show its suitability to specify and analyze health records. We use the tool Prover9 to prove properties. Then, we show that the specification can be linked to several implementations. The algebraic structures are flexible that they enable specifying several kinds of policies.

Keywords: *Health Records, Algebraic Structures, Formal Methods, Prover9*

1. Introduction

The secrecy of sensitive records is an important aspect that many organizations consider it a high priority. This aspect becomes more and more important in healthcare systems where the privacy of users records is an important issue. Healthcare systems have thousands of records that could be shared with health professionals. Therefore, healthcare systems define policies on sharing information in order to protect the privacy of users. Policies should be specified in a manner that has a correct interpretation. Also, policies should be verified according to predefined security properties. Formal methods are suitable for such applications. They have clear interpretation that has no ambiguity. Also, these methods enable formal verification of security properties such as secrecy. Finally, formal specification can be automatically converted into an implementation.

An access control model is a solution for restricting authorization to records based on policies. There are two ways to implement access control policies. One way is hiding the records behind a trusted server to control the access to the records after authenticating the users. Another solution is used in case the server is not trusted. This solution is based on making all records available to all users in encrypted forms. Keys are distributed to users in a way that each key can decrypt and reveal only the records that are allowed to be revealed. This solution has an increasing use in health systems due to the rapid use of cloud computing as a result of the reduction of operation cost.

Access control policies can be represented as a scheme. Sabri and Khedri [1]

ISSN: 2005-4238 IJAST

distinguish between two schemes: key-based scheme and object-based scheme. The first scheme focuses on keys and the relation between them. Several techniques proposed in the literature focus on the key-based scheme with the objective of efficiently creating keys and assigning them to users. The main goal of these techniques is assigning one key to each user. Usually, the authority of users is presented as a hierarchy such that the key assigned to each user can be used to derive all the keys of lower level users, but cannot derive the keys of higher level users. However, some techniques are incorrect or vulnerable to collusion attack as discussed in [2, 3].

On the other hand, the object-based scheme focuses on objects and the required keys to decrypt each on of them. Therefore, it is easier to use in specifying some systems as, for example, healthcare systems. In such systems, a policy may indicate that only the physician of a patient can reveal the information. Other physicians should not be able to reveal the same information. The specification of such policies should be straightforward in the object-based scheme while it is more complicated in the key-based scheme and may lead to incorrect specifications.

Sabri and Khedri [1] present several structures to specify key-based and object-based schemes. They provide algebraic structures to capture the main properties of encryption and decryption. They capture the properties of secret sharing, combining keys, composing ciphers, and other properties. In this paper, we first identify the main policies used in healthcare systems. Then, we employ the algebraic structures introduced in [1] to specify these policies. Furthermore, based on our specification, we verify some security properties such as secrecy using the theorem prover tool Prover9 [4]. Finally, we propose an implementation of the specifications. It is of note that more implementations other than the proposed one do exist. These implementations may have additional properties of encryption, and decryption that could affect the satisfaction of security properties as discussed in [5]. The main contributions of this paper are:

- Providing an algebraic specification to access health records
- Presenting a formal analysis using the theorem prover Prover9
- Presenting an implementations to our specification

The structure of this paper is organized as follows. Section 2 summarizes the main idea of object-based key assignment schemes. Section 3 summarizes related work. Section 4 presents the necessary mathematical background and the algebraic structures introduced in [1]. Section 5 introduces the main properties of healthcare systems with their specifications using the algebraic structures. Then, it presents proofs of some security properties and a proposed implementation. Section 6 concludes and points to future work.

2. Object-Based Key Assignment Schemes

Key-assignment schemes are policies that state the available objects (records), keys and their relationship. There are two views of these schemes. One view focuses on keys, their relations and the ability of each key to reveal objects. The other view focuses on objects and the required keys to reveal each object. An object-based scheme can state several combinations of keys to reveal each object (record). For example, a policy can be stated as any of the keys k_1 or k_2 should be used to get and access a record. Another policy can be stated as both keys k_1 and k_2 should be used together to get a record, or be stated as key k_1 should be used first to get into the next layer and then key k_2 used to get the record (Order).

To easily visualize policies, we can use a graph representation as given in [6]. For example, Figure 1 shows an object-based scheme which contains six records: R1, R2, R3, R4, R5 and R6. The figure shows that the key k_3 is required to decrypt and read the record R1. Also, it shows that any of the keys k_1 or k_2 is required to read the record R2. However,

the record R2 is in a second layer. This means that the user should decrypt first the record R1 and then use any of the keys k_1 or k_2 to get the record R2. This can be seen as a double encryption. Similarly the encrypted records R5 and R6 can be accessed after decrypting the record R1. The graph shows that using both keys k_1 and k_2 or using the k_5 is required to read the record R5.

As shown in the graph, the object-based scheme focuses on objects and explicitly states the keys required to decrypt each object, which makes it suitable for healthcare systems. However, we need to make sure that keys decrypt the right objects. For example, the key k_3 can decrypt the record R1 while the keys k3 and k5 together can decrypt the records R1, R5, and R6. It should be noted that the record R3 cannot be decrypted because the record R2 should be decrypted first. Manual verification is time consuming and may lead to error, which makes the automation of such step essential.

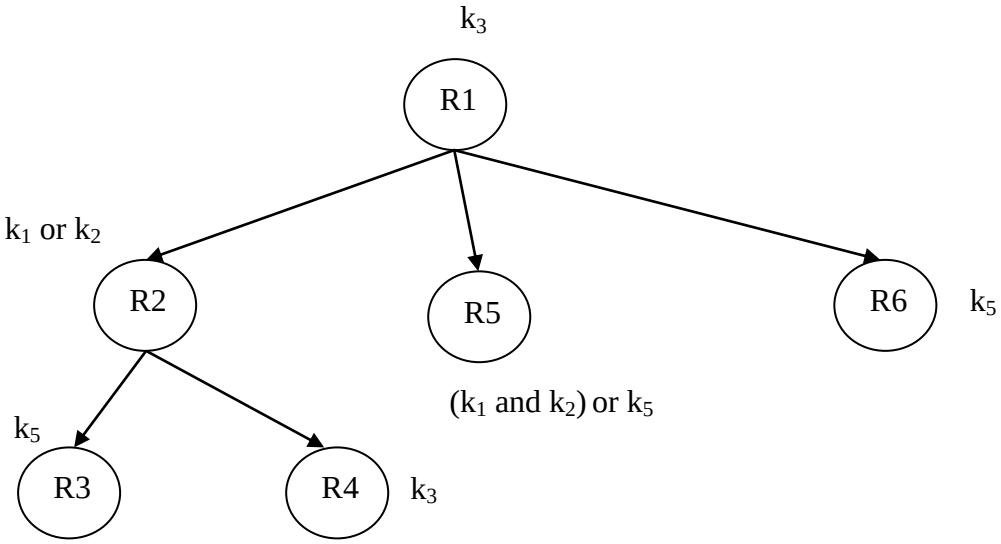

Figure 1. An Example of Object-Based Scheme

2. Related Work

Many access control models are proposed in the literature to handle health records. We distinguish between two approaches of these models. Models follow the first approach use trusted server to manage and enforce the policies. For example, Eyers *et. al.*, [7] present a prototype of electronic health records based on role access control. Becker and Sewell [8] present a role based policy specification language for access control in a distributed system called Cassandra. Their language can express trust-management related policies such as credential-based authorization, automatic trust negotiation and automatic credential retrieval strategies. The two researchers apply the policy language on electronic healthcare record system.

Other methods use cryptography to enforce access control especially the attribute based access control. For example, Li *et. al.*, [9,10] propose a framework for access control to the health records within cloud computing environment. Their scheme is flexible as it supports efficient and on-demand revocation of user access rights, and break-glass access under emergency scenarios. Ibraimi *et. al.*, [11] construct a multi-authority of ciphertext policy attribute-based encryption scheme. Their scheme allows patients to encrypt data according to an access policy over a set of attributes issued by two trusted authorities. The scheme does not require the presence of a central authority to coordinate the work of the trusted authorities. The scheme enables stating expressive access policies such as policies written in disjunctive normal form (DNF) or conjunctive normal form (CNF).

Our proposed method complements the existing attribute based access control by

giving an abstract view to the specification. Then, any attribute based access control can be used for implementation after verifying the correctness of the specification.

3. Mathematical Background

Sabri and Khedri [1] introduce algebraic structures to specify the encryption and decryption of messages. Fir t, they identify the three elements of each encrypted message: key, secret, and cipher. Each encrypted message contains a secret, which is encrypted using a cipher and a key. They specify the main operators of each element and their properties through an algebraic structure. For example, secrets can be combined using the concatenation operator. This operator is associative but not commutative. Then, based on the key and cipher structures the above researchers construct the envelope structure that specifies the interactions between ciphers and keys. Furthermore, based on the envelope and secret structures, they construct the message structure that specifies the encryption and decryption process of secrets. In this section, we present these structures after introducing the required mathematical background needed to understand these structures.

Definition 1: A *Semigroup* is an algebraic structure $A = (S, *)$ such that *S* is a set and $*$ is an associative binary operator i.e. $\forall (a, b, c \mid a, b, c \in S : (a * b) * c = a * (b * c))$.

Definition 2: A *Commutative Semigroup* is an algebraic structure $A = (S, *)$ such that *A* is a semigroup and $*$ is a commutative binary operator i.e. $\forall (a, b \mid a, b \in S : (a * b) = (b * a))$

Definition 3: An *Idempotemt Semigroup* is an algebraic structure $A = (S, *)$ such that *A* is a semigroup and $*$ is an idempotent binary operator i.e. $\forall (a \mid a \in S : a * a = a)$.

Definition 4: A *Group* is an algebraic structure $A = (S, *)$ such that *A* is a semigroup and there exists one identity element 1 i.e. $\forall (a \mid a \in S : a * 1 = 1 * a = a)$ and for each element *a* in *S* there is an element *b* in *S* such that $a * b = 1$. The element *b* is called the inverse of *a*.

Definition 5: A *Semiring* is an algebraic structure $A = (S, *, +)$ such that $(S, +)$ is a commutative semigroup, $(S, *)$ is a semigroup and $*$ distributes over + on both the left and right.

Definition 6: A *Commutative Semiring* is an algebraic structure $A = (S, *, +)$ such that *A* is a semiring and $(S, *)$ is a commutative semigroup.

Definition 7: An *Idempotent Semiring* is an algebraic structure $A = (S, *, +)$ such that *A* is a semiring and $(S, +)$ is an idempotent commutative semigroup.

Definition 8: The multiplicatively absorbing zero in a semiring $A = (S, *, +)$ is an element 0 such that 0 is an identity element in $(S, +)$ and absorption element in $(S, *)$ i.e. $\forall (a \mid a \in S : a * 0 = 0 * a = 0)$.

Definition 9: A *Left-Quasi Semimodule over S* or *S-Left-Quasi-Semimodule* is an algebraic structure $({}_S A, \oplus)$ such that $(S, *, +)$ is a semiring, $(A, \oplus)$ is a semigroup, and there exists a function $S \times A \to A$ such that for all $r, s \in S$ and $a, b \in A$ we have

1. $r(a \oplus b) = ra \oplus rb$
2. $r(sa) = (r * s)a$

Definition 10: A *Unital Left-Quasi Semimodule over S* is an algebraic structure $({}_S A, \oplus)$ such that $({}_S A, \oplus)$ is a left-quasi semimodule over S and $\forall(a \mid a \in A : 1_s a = a)$

Definition 11: A *Zero Preserving Left-Quasi Semimodule over S* is an algebraic structure $({}_S A, \oplus)$ such that $({}_S A, \oplus)$ is a left-quasi semimodule over S and $\forall(a \mid a \in A : 0_s a = 0_A)$

Definition 12: A *Left-Semimodule over S* or *S-Left-Semimodule* is an algebraic Structure $({}_S A, \oplus)$ is a left-quasi semimodule over *S* and $(r + s)a = ra \oplus sa$

Next, we define the algebraic structures used to specify the encryption and decryption process of messages.

Definition 13: A *Secret Structure* is an algebraic structure $S = (S, *_s, +_s, 0_s)$ such that *S* is an additively idempotent semiring with a multiplicatively absorbing zero.

The secret structure is used to specify secrets. A secret can be considered as a set of strings. Two operators are defined within the secret structure. The operator $*_s$ is used to represent combining secrets as for example the concatenation of string. The operator $+_s$ can be considered as a set union. The 0_s secret can be seen as an empty set.

Definition 14: A *Key Structure* is an algebraic structure $K = (K, *_k, +_k, 0_k)$ such that *K* is an additively idempotent commutative semiring with a multiplicatively absorbing zero.

The key structure is used to specify keys. Two operators are defined within the key structure. The operator $*_k$ is used to represent that both keys are required to encrypt or decrypt a secret while the operator $+_k$ is used to specify that one key among several keys is required to reveal the secret. The key 0_k can be seen as corrupted key that cannot be used for neither encryption nor decryption.

Definition 15: A *Cipher Structure* is an algebraic structure $C = (C, *_c, +_c, 0_c, 1_c, ')$ such that *C* is an additively idempotent semiring with a multiplicatively absorbing zero, an identity, and a multiplicative inverse for each element of *C*.

The cipher structure specifies the properties of ciphers. A cipher is an algorithm used to transform a secret from one form to another. The $*_c$ can be seen as a composition of algorithms while the $+_c$ can be seen as a selection between ciphers. The cipher a′ represents the decryption algorithm of the encryption algorithm a. The 1_c is an identity cipher that has no effect on secrets.

The envelope structure defined below specifies the interactions between keys and ciphers. We define two structures: the multiplicative envelope and the additive envelope based on the main operators $*_c$ and $+_c$ of ciphers.

Definition 16: A *Multiplicative Envelope* is an algebraic structure $E^* = ({}_K C, *_c)$ such that E is a quasi-semimodule over a key structure K and a C is a group.

Definition 17: An *Additive Envelope* is an algebraic structure $E^{+}=(_{K} C_{0},+_{c})$ such that *K* is a key structure and C_0 is an idempotent semigroup with an identity element 0_c.

Definition 18: An *Envelope Structure* is an algebraic structure $E=(E^{+},E^{*})$ such that $*_c$ distributes over $+_c$ on both left and right.

The encryption and decryption process is specified using the message algebraic structure specified as given in the following definition.

Definition 19: A *Message Structure* is an algebraic structure $M=(_{E} S,+_{s})$ such that M is a unitary zero preserving semimodule over an envelope structure E with an associative operator $*_s$, an idempotent operator $+_s$, and $*_s$ distributes over $+_s$ on both left and right.

4. Healthcare System

In this section, we first introduce the main requirements in healthcare systems. Then, we give a specification based on an illustrative example. Finally, we analyze the specification and give a possible implementation. In this section, we present the main policies that should be considered when handling healthcare records as given in the literature. We present here a part of the system. However, the full specification should not be an issue. A healthcare system has several users, and each user plays a role such as patient, physician, nurse, and administrative role. Patient record consists of three main parts:

1. General Part: This part contains general information which can be read by nurses and physician
2. Sensitive Part: This part contains the information that can be read only by the physician of that patient
3. Administrative Part: This part deals with appointments and contact information.

In the next subsection, we give an illustrative example and show its specification.

4.1. Specification

We assume that the system handles information of two patients Mark and David. The patient Mark is assigned to the physician John, while the patient David is assigned to the physician William. There are three parts of the record assigned to Mark: the general part of his record R_{M_G}, the sensitive part of his record R_{M_S}, and the administrative part of his record R_{M_A}. Assume that the key k_m is assigned to the patient Mark that should be able to decrypt all parts of his record. Therefore, we have

$$m_1 = (k_m \circ a)(R_{M\_G} + R_{M\_S} + R_{M\_A})$$

Similarly, a record with three parts is assigned to David: the general part of his record R_{D_G}, the sensitive part of his record R_{D_S}, and the administrative part of his record R_{D_A}. Assume that the key k_D is assigned to the patient David which should be able to decrypt all parts of his record. Therefore, we have

$$m_2 = (k_D \circ a)(R_{D\_G} + R_{D\_S} + R_{D\_A})$$

The sensitive part of Mark record is also encrypted using the key of the physician John. Two keys are assigned to John. One key is assigned to all physicians k_P and the other key is only for John k_J. To be able to decrypt the sensitive part of Mark record, both keys

should be combined. Therefore, if John is no longer playing the role of physician, he should not be able to access the sensitive information of Mark

$$m_3 = ((k_P *_k k_J) \circ a) R_{M\_S}$$

The sensitive part of David record is also encrypted using the key of the physician William. Two keys are assigned to William. One key is assigned to all physicians k_P and the other key is only for William k_W. To be able to decrypt the sensitive part of David record, both keys should be combined.

$$m_4 = ((k_P *_k k_W) \circ a) R_{D\_S}$$

Assume that Marry and Sarah are playing the role of nurse. Therefore, they are able to access the general part of Mark and David records. Both Mary and Sarah are assigned the key assigned to nurses k_N

$$m_5 = (k_N \circ a)(R_{M\_G} +_s R_{D\_G})$$

Assume that the physicians are also allowed to access the general information of patients. Therefore, we need to add another policy such as

$$m_6 = (k_P \circ a)(R_{M\_G} +_s R_{D\_G})$$

User playing the administrative role should be assigned the key k_A to be able to access the administrative records of the patients.

$$m_7 = (k_A \circ a)(R_{M\_A} +_s R_{D\_A})$$

It should be noted that the above example can be extended to include the department of users where each user is able to access only the records within his/her department. In this case, each department should have its own key distributed to all users working in that department.

4.2. Analysis

We use Prover9 to analyze our specification. First, we write our specification in Prover9 as given in the appendix. Then, we use Prover9 to prove theorems related to the correctness of our specifications and security properties. To prove our properties, we represent all the messages specified previously as

$$m = m_1 +_s m_2 +_s m_3 +_s m_4 +_s m_5 +_s m_6 +_s m_7$$

Then, we specify the following five properties as:

P1: David is able to access the sensitive part of his record.

$$R_{D\_S} \leq_s (k_D \circ a')(m)$$

P2: Nurses can access the general part of Mark record.

$$R_{M\_G} \leq_s (k_N \circ a')(m)$$

P3: The physician William can access the general part of Mark record.

$$R_{M\_G} \leq_s ((k_P +_k k_w) \circ a')(m)$$

P4: Nurses cannot access the sensitive part of Mark record.

$$\neg(R_{M\_S} \leq_s (k_N \circ a')(m))$$

P5: The physician William cannot access the sensitive part of Mark record.

$\neg(R_{M\_s} \leq_s ((k_P +_k k_w) \circ a')(m))$

By using Prover9, we were able to prove the first three properties. However, we were not able to prove the negation of properties 4 and 5, which indicates that nurses and the physician William cannot access the sensitive part of Mark record. Such verification can be obvious in case of small examples, but it is more complicated for large systems.

4.3. Implementation

There can be several models (implementations) that satisfy the properties of the algebraic structures presented above as given in [1,5]. For example, a cipher can be an RSA cryptosystem. A secret is a set of natural numbers. The $+_s$ is set union while $*_s$ is a concatenation of strings. A key is an RSA key where $+_k$ is set union and $*_k$ is number multiplication. Note that this is one implementation. Other implementations can be found in [5].

5. Conclusion

In this paper, we employ the algebraic structures defined in [1] to specify and analyze health records. We show the suitability of these structures to specify policies formally in healthcare systems. Then, we show the verification of security properties using the theorem prover Prover9. Finally, we show that our specification can be implemented using RSA cryptosystem. One of the problems that we face is that it takes a long time to give a result when proving some properties. As a future work we intend to prove auxiliary propositions in order to simplify and facilitate proving the properties. Also, we intend to build a tool that generates the encryption code automatically from the specification.

6. APPENDIX

```
% Axiom of Secrets
% PS is + on secrets
% DS is * on secrets
% RS is the partial order relation

PS(v1,v2)=PS(v2,v1).
PS(v,v)=v.
PS(v1,PS(v2,v3))=PS(PS(v1,v2),v3).
PS(v,"0s")=v.
DS(v1,DS(v2,v3))=DS(DS(v1,v2),v3).
DS(v,"0s")="0s".

DS(PS(v1,v2),v3)=PS(DS(v1,v3),DS(v2,v3)).
DS(v1,PS(v2,v3))=PS(DS(v1,v2),DS(v1,v3)).
RS(v1,v2) <-> PS(v1,v2)=v2.

% Axiom of Keys
% PK is + on keys
% DK is * on keys
% RK is the partial order relation

PK(x1,x2)=PK(x2,x1).
```



```
PK(x,x)=x.
PK(x1,PK(x2,x3))=PK(PK(x1,x2),x3).
PK(x,"0k")=x.
DK(x1,x2)=DK(x2,x1).
DK(x1,DK(x2,x3))=DK(DK(x1,x2),x3).
DK(x,"0k")="0k".
DK(PK(x1,x2),x3)=PK(DK(x1,x3),DK(x2,x3)).
DK(x1,PK(x2,x3))=PK(DK(x1,x2),DK(x1,x3)).
RK(x1,x2) <-> PK(x1,x2)=x2.

% Axiom of Ciphers
% PC is + on ciphers
% DC is * on ciphers
% RC is the partial order relation
PC(y1,y2)=PC(y2,y1).
PC(y,y)=y.
PC(y1,PC(y2,y3))=PC(PC(y1,y2),y3).
PC(y,"0c")=y.
DC(y1,DC(y2,y3))=DC(DC(y1,y2),y3).
DC(y,"0c")="0c".
y!="0c" -> DC(y,y')="1c".
y!="0c" -> DC(y',y)="1c".
DC(y,"1c")=y.
DC("1c",y)=y.
DC(PC(y1,y2),y3)=PC(DC(y1,y3),DC(y2,y3)).
DC(y1,PC(y2,y3))=PC(DC(y1,y2),DC(y1,y3)).
RC(y1,y2) <-> PC(y1,y2)=y2.

% Additive Envelop
E(PK(x1,x2),y) = PC(E(x1,y),E(x2,y)).
E(x,PC(y1,y2)) = PC(E(x,y1),E(x,y2)).
E(x1,E(x2,y)) = E(DK(x1,x2),y).
E("0k",y) = "0c".

% Multiplicative Envelop
E(x,DC(y1,y2)) = DC(E(x,y1),E(x,y2)).
E(x1,E(x2,y)) = E(DK(x1,x2),y).
E("0k",y) = "0c".

% Additional Envelope properties
E(x1,"0c")="0c".
x1!="0k" -> E(x1,"1c")="1c".

% Secret-E Semimodule
S(PC(y1,y2),v) = PS(S(y1,v),S(y2,v)).
S(y,PS(v1,v2)) = PS(S(y,v1),S(y,v2)).
S(y1,S(y2,v)) = S(DC(y1,y2),v).
S("0c",v) = "0s".
S("1c",v) = v.

% Properties Regarding the semimodule
E(x,"0c")="0c".
-(E(x,y')="0c") -> E(x,y')=E(x,y)'.
```

```
exists y (-(E(x,y)="0c") & -(E(x,y')="0c"))   ->
E(x,"1c")="1c".
S(x,"0s")="0s".

% Properities of the ordering relation:
RS(v1,v2) -> RS(PS(v1,v3),PS(v2,v3)).
RS(v1,v2) -> RS(PS(v3,v1),PS(v3,v2)).
RS(PS(v1,v2),v3) -> RS(v1,v3) & RS(v2,v3).
RS(v1,PS(v1,v2)).
RS("0s",v).
RS(v1,v2) & RS(v3,v4) -> RS(PS(v1,v3),PS(v2,v4)).
RS(v1,v2) -> RS(DS(v1,v3),DS(v2,v3)).
RS(v1,v2) & RS(v3,v4) -> RS(DS(v1,v3),DS(v2,v4)).
RS(v1,v2) -> RS(v1,PS(v2,v3)).
RK(x1,x2) -> RK(PK(x1,x3),PK(x2,x3)).
RK(x1,x2) -> RK(PK(x3,x1),PK(x3,x2)).
RK(PK(x1,x2),x3) -> RK(x1,x3) & RK(x2,x3).
RK(x1,PK(x1,x2)).
RK("0k",x).
RK(x1,x2) & RK(x3,x4) -> RK(PK(x1,x3),PK(x2,x4)).
RK(x1,x2) -> RK(DK(x1,x3),DK(x2,x3)).
RK(x1,x2) & RK(x3,x4) -> RK(DK(x1,x3),DK(x2,x4)).

% Additional Properties
RK(x1,x2) -> RS(S(E(x1,y),v),S(E(x2,y),v)).
-(y="0c") -> exists x (S(x,S(y,v)))=v.
RK(x1,x2) -> RC(E(x1,y),E(x2,y)).
RS(v1,v2) & RS(v,S(y1,v1)) -> RS(v,S(y1,v2)).
RC(y1,y2) -> RS(S(E(x,y1),v),S(E(x,y2),v)).
S(E(x,y'),S(E(x,y),z))=S(DC(E(x,y'),E(x,y)),z).
S(DC(E(x,y'),E(x,y)),z)=S(E(x,DC(y',y)),z).
S(E(x1,y1),S(E(x,y'),S(E(x,y),z))) = S (E(x1,y1), S( DC(
E(x,y'),E(x,y)),z) ).

x1!="0k" & y1!="0c" & x!="0k" & y!="0c" ->
S(E(x1,y1),S(E(x,y'),S(E(x,y),z))) = S(E(x1,y1),z).
x!="0k" -> S(E(x,a'),S(E(x,a),z))=z.
DC(a',a)="1c".
x!="0k" -> S(E(x,DC(a',a)),z)= S(E(x,"1c"),z).

% Specifying the messages of the example
m1 = S(E(km,a),PS(PS(Rmg,Rms),Rma)).
m2 = S(E(kd,a),PS(PS(Rdg,Rds),Rda)).
m3 = S(E(DK(kp,kj),a),Rma).
m4 = S(E(DK(kp,kw),a),Rda).
m5 = S(E(kn,a),PS(Rmg,Rdg)).
m6 = S(E(kp,a),PS(Rmg,Rdg)).
m7 = S(E(ka,a),PS(Rma,Rda)).
m = PS(m1,PS(m2,PS(m3,PS(m4,PS(m5,PS(m6,m7)))))).

% Assuming that the cipher is not 0 and all the keys are not
zero as well and not equivalent

distinct([x, y : z]) ->
```



```
  ( (x != y) &
    distinct([x : z]) &
    distinct([y : z])).
distinct(["0k",km,kd,kp,kj,kw,kn,ka]).
distinct([Rmg,Rms,Rma,Rdg,Rds,Rda,"0s",m1,m2,m3]).
```

References

[1] K. E. Sabri and R. Khedri, "Algebraic Framework for the Specification and Analysis of Cryptographic-Key Distribution", Fundamenta Informaticae, vol. 112, no. 4, **(2011)**, pp. 305-335.
[2] K. E. Sabri, "Algebraic Analysis of Akl and Taylor Key Assignment Scheme", International Review on Computers and Software, vol. 9, no. 11, **(2014)**, pp. 1881-1887.
[3] J. Crampton, K. Martin and P. Wild, "On Key Assignment for Hierarchical Access Control", 19th IEEE Computer Security Foundations Workshop, Venice, Italy, **(2006)**.
[4] W. McCune, Prover9 and Mace4, http://www.cs.unm.edu/~mccune/mace4, accessed on **(2015)**.
[5] K. E. Sabri, "Algebraic Analysis of Object-Based Key", Journal of Software, vol. 9, no. 9, **(2014)**, pp. 2033-2042.
[6] G. Miklau and D. Suciu, "Controlling access to published data using cryptography", Proceedings of the 29th international conference on very large data bases, Berlin, Germany, **(2003)**.
[7] D. Eyers, J. Bacon and K. Moody, "OASIS role-based access control for electronic health records", IEE Proceedings Software, vol. 153, no. 1, **(2006)**, pp. 16-23.
[8] M. Becker and P. Sewell, "Cassandra: Flexible Trust Management, Applied to Electronic Health Records", 17th IEEE Computer Security Foundations Workshop, Pacific Grove, USA, **(2004)**.
[9] M. Li, S. Yu, K. Ren and W. Lou, "Securing Personal Health Records in Cloud Computing: Patient-Centric and Fine-Grained Data Access Control in Multi-owner Settings", SecureComm, Singapore, **(2010)**.
[10] M. Li, S. Yu, Y. Zheng, K. Ren and W. Lou, "Scalable and Secure Sharing of Personal Health Records in Cloud Computing using Attribute-based Encryption", IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 1, **(2013)**, pp. 131-143.
[11] L. Ibraimi, M. Asim and M. Petkovic, "Secure management of personal health records by applying attribute-based encryption", 6th International Workshop on Wearable Micro and Nano Technologies for Personalized Health (pHealth), Oslo, **(2009)**.
[12] K. E. Sabri and R. Khedri, "A Generic Algebraic Model for the Analysis of Cryptographic-Key Assignment Schemes", Foundations and Practice of Security, Montreal, QC, Canada, **(2013)**.

Author

Khair Eddin Sabri has been working as an assistant professor in the Computer Science Department at The University of Jordan since 2010. He obtained his B.Sc. degree in Computer Science from the Applied Science University, Jordan in June 2001. He also received M.Sc. degree in Computer Science from The University of Jordan in January 2004 and a Ph.D. degree in Software Engineering from McMaster University, Ontario Canada in June 2010. He is a member of the Formal Requirements and Information Security Enhancement (FRAISE) Research Group. His main research interest is the formal verification and analysis of security properties.

