Algebraic Analysis of Object-Based Key Assignment Schemes

Khair Eddin Sabri
Computer Science Department
King Abdulla II School for Information Technology
The University of Jordan, Amman, Jordan
Email: k.sabri@ju.edu.jo

Abstract— **The confidentiality of information is an important aspect of security. One way to achieve the confidentiality is through restricting access to information to the authorized users only. Access control can be enforced by using encryption. In this case, all the information is encrypted and keys are assigned to users such that each key reveals the authorized part of the information. Key assignment can be classified as key-based or object-based schemes based on the focus of the scheme.**

Sabri and Khedri [1] present algebraic structures to specify algebraically cryptosystems by capturing the common properties of ciphers, secrets and keys. Also, these structures are used for the analysis of security properties in object-based key assignment schemes. However, these structures are abstract, and no linkage has been proposed to the existing cryptosystems. In this paper, we extend their work by giving concrete models for their algebraic structures. We give concrete models to Vigenère, transposition ciphers, DES and RSA cryptosystems. Also, we investigate the effects of extra algebraic properties that some cryptosystems may have on the security analysis of object-based schemes.

Index Terms— **Object-based key assignment schemes, verification, cryptography, algebraic structures.**

I. Introduction

Access control is used to provide confidentiality of information. The typical implementation of access control is through the use of trusted server that gives authorization to users based on predefined policies as in [2], [3], [4]. Another implementation that eliminates the use of a trusted server is through the use of cryptography such that the information is public but encrypted. Therefore, keys are assigned to users such that each key is used to decrypt the authorized part of the information.

Key assignment schemes can be classified into key-based schemes and object-based schemes [1]. Key-based schemes focus on keys and the relationship between them. It is usually used when we have hierarchy in the security labels assigned to users. For example, the relation $u_1 > u_2$ indicates that the user u_1 has more authority than the user u_2. In this case, the key k_1 assigned to user u_1 should be able to reveal more information than the key k_2 assigned to user u_2. We write $k_1 > k_2$ when any information that can be revealed using the key k_2 can also be revealed using the key k_1. One way to implement this relation is through deriving the key k_2 from the key k_1. Several techniques are proposed in the literature to derive a key from another one as in [5], [6]. In [7], we introduce an algebraic model that links several techniques together to specify and analyse key-based schemes.

Object-based schemes focus on objects and the required condition to reveal the information from each object. This scheme is used when several users should cooperate to reveal an information. For example, conditions can be stated as the use of the keys k_1 and k_2 together to reveal a piece of information, while any one of the key k_1 or k_3 should be used to reveal another piece of information.

Sabri and Khedri [1] introduce algebraic structures to specify cryptosystems by specifying the common properties of secrets, ciphers, and keys. Also, they introduce an algebraic structure to specify the interaction between keys and ciphers and another algebraic structure to specify the process of encryption and decryption. They use their algebraic structures to verify, at an abstract level, security properties in systems that employed object-based schemes. Their analysis is based on the common algebraic properties of secrets, keys, and ciphers. However, the use of a specific cryptosystem may affect the satisfaction of security properties due to the fact that some cryptosystems have additional algebraic properties. For example, the composition of encryption of Vigenère is commutative $E_{k1}(E_{k2}(M)) = E_{k2}(E_{k1}(M))$ i.e., the order of multiple encrypting does not matter. Others such as RSA has the homomorphic property $E(M_1).E(M_2) = E(M_1.M_2)$ i.e., concatenating two encrypted messages is the same as encrypting after concatenating the two messages. The effects of the algebraic properties of cryptosystems are taken into consideration mainly in the verification of cryptographic protocols properties [8], [9], [10], where several flows are detected by involving the algebraic properties in the analysis other than the cancelling property $E'_k(E_k(m)) = m$ i.e., the decryption cancels the effect of encryption.

In this paper, we extend the work of Sabri and Khedri [1] by providing concrete algebraic models of cryptosystems to their abstract structures. These cryptosystems are Vigenère, transposition cipher, DES and RSA [11]. We choose these cryptosystems because of

Manuscript received ???; revised ???; accepted ???. ©

doi:10.4304/jsw.9.8.2033-2042

their popularity and their algebraic properties that are useful to illustrate our idea. We are aware that Vigenère and transposition cryptosystems are not used in practice. However, we are presenting them in this paper because they are simple and easy to understand. Therefore, they are suitable to illustrate the algebraic structures presented in [1].

The presented concrete algebraic models act as an intermediate model between the abstract level of an algebraic structure and its implementation. Providing concrete models to abstract algebraic structures is not new. For example, Khedri [12] introduce a concrete model represented as relation algebra to an abstract algebraic model called feature algebra.

We also analyse the effect of the additional algebraic properties of the specified cryptosystems on the security properties of object-based key assignment schemes. To the best of our knowledge, security analysis of object-based schemes based on the algebraic properties of cryptosystems does not exist in the literature. It is worth mentioning that this paper does not investigate the security of cryptosystems themselves, but it analyse the effect of the algebraic properties of cryptosystems on the verification of object-based schemes.

This paper is organized as follows: Section II summarizes related work. Section III gives the necessary mathematical background and present the algebraic structures of [1]. Section IV analyses security properties in object-based schemes. Section V represents algebraically Vigenère, transposition cipher, DES and RSA cryptosystems. Finally, we conclude in Section VI and point to future works.

II. Literature Review

The algebraic properties of cryptosystems is used in many security applications. In this section, we present some of them. Even our algebraic model can be used in all these applications, the focus of this paper is on the analysis of key assignment schemes.

Key Assignment Schemes: Several techniques exist in the literature to provide confidentiality of information through the use of cryptography. Many of these techniques are key-based schemes [5], [13], [6], [14], [15]. These techniques provide a way to implement the relation $k_1 > k_2$ through key derivation i.e., deriving the key k_2 from the key k_1. Other researchers focus on analysing these techniques and comparing them [16], [7]. Others use key derivation to provide confidentiality in data outsourcing [17], [18].

A technique that follows the object-based scheme is the work of Miklau and Suciu [19], where the authors develop a language for specifying access control policies on XML. They represent policies as a tree where its nodes represent objects. Each object is associated with a condition that represents the required keys to reveal that object i.e., the use of the keys k_1 and k_2 together or the use of one of the keys k_1 or k_3. This tree is implemented by using the *xor* operator for secret sharing and AES cryptosystem for encryption. Miklau and Suciu scheme is analysed by Abadi and Warinschi [20]. In our paper, we specify object-based scheme similar to [19]. However, in this paper, we propose several implementations for object-based schemes and analyse the effect of each implementation on the security of schemes based on its algebraic properties.

Analysis of Cryptographic Protocols: Cryptographic protocols are communicated protocols whose messages are encrypted. They provide security properties such as confidentiality of information and authentication between communicated users. Several techniques are presented in the literature [21], [22], [23] to analyse these protocols and verify the satisfactory of their security properties. The analysis is based on abstracting the used cryptosystem in the analysed protocol by focusing on its algebraic properties. The main considered property is the cancellation property which states that decryption cancels the effect of encryption when appropriate keys are used. Researchers report that some cryptosystems have additional properties such as the commutativity and associativity of encryption. Using a cipher with such properties in some protocols would make those protocols vulnerable to attacks [10]. Therefore, new researches focus more on the algebraic properties that cryptosystems may have when analysing cryptographic protocols [24], [9], [25].

Threshold Cryptosystem: Secret sharing is an approach used for splitting the secret into shares and distributing them between n users such that k of them should combine their shares to construct the secret. One of the earliest methods is the Shamir method [26] based on interpolation. This approach requires the exchange of the shares usually with a trusted party. Another approach that does not require exchanging the shares but has the same goal of revealing a secret is called threshold cryptosystem. In this approach, k users cooperate in the decryption of a cipher text using their keys to reveal the secret. One of the earliest research on RSA threshold is the work of Boyd [27] based on the the algebraic property $E_{k_1}(E_{k_2}(s)) = E_{k_1 * k_2}(s)$. This property states that encryption/decryption of a secret twice using the keys k_1 and k_2 is equivalent to encrypting/decrypting of the secret using a combined key $k_1 * k_2$.

Cryptanalysis: Cryptanalysis is the process of revealing a secret from an encrypted message without using a key. This field is intensively studied to analyse the security of cryptosystem. Representing cryptosystems algebraically presents another view of analysing them. This representation is used in [28], [29] to algebraically analyse the security of some cryptosystems such as Advanced Encryption Standard (AES).

III. Mathematical Background

In this section, we first introduce the required mathematical background [30]. Then, we present the algebraic structures given by [1] to specify the elements of a cryptosystem and the encryption and decryption of secrets.

A. Algebraic Structures

Definition 3.1: A *semigroup* is an algebraic structure $\mathcal{A} = (S, \cdot)$, where S is a set and $\cdot$ is an associative binary operator. If the operator $\cdot$ is commutative, we call $\mathcal{A}$ a *commutative semigroup*. If the operator $\cdot$ is idempotent, we call $\mathcal{A}$ an *idempotent commutative semigroup*. □

Definition 3.2: A *group* is an algebraic structure $\mathcal{A} = (G, \cdot, \prime, 1)$, where $(G, \cdot)$ is semigroup, 1 is an identity element of $\cdot$, and for every element g in G there is an inverse element g' such that $g \cdot g' = 1$. □

Definition 3.3: Let $S \neq \emptyset$ be a set and $+$ and $\cdot$ binary operations on S, named addition and multiplication. Then $(S, +, \cdot)$ is called a *semiring* if $(S, +)$ is a commutative semigroup, $(S, \cdot)$ is a semigroup, and $\cdot$ distributes over $+$ on both the left and right. □

Definition 3.4: Let $(S, +, \cdot)$ be a semiring.

- If the semigroup $(S, \cdot)$ has a neutral element 1_s, we call 1_s the identity of the semiring $(S, +, \cdot)$.
- If the semigroup $(S, +)$ has a neutral element 0_s, we call it the zero of the semiring $(S, +, \cdot)$. We call 0_s the multiplicatively absorbing if 0_s is absorbing in $(S, \cdot)$ i.e., $\forall(x \mid x \in S : 0_s \cdot x = x \cdot 0_s = 0_s)$
- If $(S, +, \cdot)$ has an identity 1_s, we call a' the inverse of a iff $a \cdot a' = a' \cdot a = 1_s$.
- If $(S, +)$ is an idempotent semigroup, we call $(S, +, \cdot)$ an additively idempotent.
- If $(S, \cdot)$ is a commutative semigroup, we call $(S, +, \cdot)$ a commutative semiring.

□

Definition 3.5: Let $\mathcal{A} = (A, \cdot)$ be commutative semigroup and $\mathcal{S} = (S, +, \cdot)$ be a semiring. We call $({}_{\mathcal{S}}A, \cdot)$ a *left-quasi semimodule* over $\mathcal{S}$ or $\mathcal{S}$-left-quasi-semimodule if there exists a function $S \times A \longrightarrow A$ such that for all $r, s \in S$ and $a, b \in A$, we have:

1) $r(a \cdot b) = ra \cdot rb$
2) $r(sa) = (rs)a$

An $\mathcal{S}$-left-quasi-semimodule $({}_{\mathcal{S}}A, \cdot)$ is called *unital* if $\mathcal{S}$ has an identity 1_s and $\forall(a \mid a \in A : 1_s a = a)$. Furthermore, $({}_{\mathcal{S}}A, \cdot)$ is called *zero-preserving* if there are zeros 0_a and 0_s respectively of $\mathcal{A}$ and $\mathcal{S}$ that satisfy $\forall(a \mid a \in A : 0_s a = 0_a)$. □

Definition 3.6: Let $({}_{\mathcal{S}}A, +)$ be a quasi-left-semimodule. We call $({}_{\mathcal{S}}A, +)$ a *left-semimodule* over $\mathcal{S}$ or $\mathcal{S}$-left-semimodule if $(r + s)a = ra + sa$. □

For simplicity, we use the term quasi-semimodule to denote left-quasi-semimodule and the term semimodule to denote left-semimodule.

Definition 3.7: Let A, B, and C be sets, and P and Q be relations such that $P \subseteq A \times B$ and $Q \subseteq B \times C$.

- $P;Q \triangleq \{(x, z) \mid \exists(y \mid y \in B : (x, y) \in P \wedge (y, z) \in Q)\}$
- $P^{\cup} \triangleq \{(x, y) \mid (y, x) \in P\}$

□

In Definition 3.7, $P;Q$ denotes relational composition, and $P^{\cup}$ denotes the converse of the relation P.

B. Algebraic Structures of Cryptosystems

Sabri and Khedri [1] introduce algebraic structures to specify the encryption and decryption of messages. They define three structures to specify *secrets*, *keys*, and *ciphers*. Also, they define a structure to specify the interaction between keys and ciphers, and another structure to specify the process of encryption and decryption. The secret structure that captures properties of secrets is defined in [1] as

Definition 3.8 (Secret Structure [1]): Let $\mathcal{S} \stackrel{\text{def}}{=} (S, +_s, *_s, 0_s)$ be an algebraic structure that is an additively idempotent semiring with a multiplicatively absorbing zero. We call $\mathcal{S}$ a *secret structure*. □

In the secret structure, S is a set of secrets. The operator $+_s$ intuitively represents selecting between secrets while the operator $*_s$ represents combining secrets. The 0_s represents a null secret.

Definition 3.9 (Key Structure [1]): Let $\mathcal{K} \stackrel{\text{def}}{=} (K, +_k, *_k, 0_k)$ be an algebraic structure that is an additively idempotent commutative semiring with a multiplicatively absorbing zero. We call $\mathcal{K}$ a *key structure*. □

In the key structure, K is a set of keys. The two binary operators $+_k$ and $*_k$ are used to specify combining two keys such that the $+_k$ operator represents combining keys in a way that only one key is used to encrypt or decrypt one unit of a message i.e., the choice of a key, while the $*_k$ operator represents combining keys in a way that both of them are used simultaneously to encrypt or decrypt one unit of a message i.e., key sharing. The 0_k represents a key that is not suitable for encryption or decryption.

The key structure is a secret structure since a key can be seen as a secret. However, the combining operator $*_s$ should be commutative to represent sharing a key between several users, so that the order of combining keys does not matter.

Ciphers and keys are used together to encrypt and decrypt a secret. A cipher gives the transformation approach while the key selects a particular transformation. Therefore, both of them should be used. The cipher structure given below deals with both of them as one block. It describes the transformation without taking into consideration the used key.

Definition 3.10 (Cipher Structure [1]): Let $\mathcal{C} \stackrel{\text{def}}{=} (C, *_c, +_c, 1_c, \overset{c}{\prime}, 0_c)$ be an algebraic structure that is an additively idempotent semiring with a multiplicatively absorbing zero, an identity, and a multiplicative inverse for each element of C. We call $\mathcal{C}$ a *cipher structure*. □

A cipher is a set of transformation methods. The operator $+_c$ represents selecting between two ciphers. The operator $*_c$ represents applying one cipher after another. The operator $\overset{c}{\prime}$ is used to represent the inverse cipher i.e., the decryption of an encrypted message. The 1_c is a cipher that has no effect on messages.

The envelope structure states explicitly the interaction between a cipher and a key. It involves one operator between a cipher and a key. This operator can be seen as an application of a key into a cipher. For example, a cipher can be seen as a function and a key as a parameter to that function. The following two structures represent

the properties that involve ciphers and keys based on the operators of the cipher structure.

Definition 3.11 (Multiplicative Envelope [1]): Let $\overset{*}{\mathcal{E}} \overset{\text{def}}{=} \left({}_{\mathcal{K}}C, *_c\right)$ be a quasi-semimodule over a key structure $\mathcal{K} \overset{\text{def}}{=} (K, +_k, *_k, 0_k)$. Moreover, 1_c is the identity of $(C, *_c)$, and there exists an inverse for each element of C (i.e., $\left(C, *_c, 1_c, \overset{c}{I}\right)$ is a group). We call $\overset{*}{\mathcal{E}}$ a *multiplicative envelope*. □

Definition 3.12 (Additive Envelope [1]): Let $0_c \notin C$ and $C_0 \overset{\text{def}}{=} C \cup \{0_c\}$. Let $\overset{+}{\mathcal{E}} \overset{\text{def}}{=} \left({}_{\mathcal{K}}C_0, +_c\right)$ be a zero-preserving semimodule over a key structure $\mathcal{K} \overset{\text{def}}{=} \left(K, +_k, *_k\right)$, where $+_c$ is idempotent and 0_c is its identity. We call $\overset{+}{\mathcal{E}}$ an *additive envelope*. □

Definition 3.13 (Envelope [1]): Let $\mathcal{E} \overset{\text{def}}{=} (\overset{+}{\mathcal{E}}, \overset{*}{\mathcal{E}})$ be a system where $*_c$ distributes over $+_c$ on both the left and right. We call $\mathcal{E}$ an *envelope structure*. □

Definition 3.14 (Message [1]): Let $\mathcal{M} = \left({}_{\mathcal{E}}S, +_s\right)$ be a unitary zero-preserving semimodule over an envelope structure $\mathcal{E}$ with an associative operator $*_s$, an idempotent operator $+_s$, and where $*_s$ distributes over $+_s$ on both the left and right. We call $\mathcal{M}$ a *message structure*. □

The message structure represents the encryption and decryption of secrets. The message structure consists of a secret and an envelope. The binary operator in the semimodule represents using an envelope to encrypt or decrypt a secret. The axioms of the envelope and message structures are presented in the appendix.

IV. Analysis of Object-Based Schemes

Object-based schemes define policies that describe a set of secrets and state the required keys to reveal each one of them. However, these policies should satisfy a predefined properties. For example, assume that we have three secrets s_1, s_2, and s_3. Also, assume that we assign three keys to the users of the system as follow: the key k_1 to Alice, k_2 to Bob and k_3 to Carol. Assume that we have the policies:

- Carol should be able to get the secret s_1. This policy can be specified as
  $m_1 := (k_3 \circ a) \cdot s_1$.
- The secret s_2 is decrypted twice using first the key of Bob and then the key of Alice. This policy states an order of decryption to get the secret. The encrypted message should be decrypted first by Alice then by the Bob. Here, Bob can get the secret but after the approval of Alice. $m_2 := (k_1 \circ a) \cdot ((k_2 \circ a) \cdot s_2)$
- The secret s_3 can be revealed by using keys of Alice and Bob together, or by using Carol key.
  $m_3 := (((k_1 *_k k_2) +_k k_3) \circ a) \cdot s_3$

We use our framework with the aid of Prover9 [31] to prove the following properties:

1) Carol can reveal the secrets s_1 and s_3 i.e., applying the key of Carol k_3 to the existing messages m_1, m_2, and m_3 should reveal s_1 and s_3.
   $(s_1 + s_3) \leq_s (k_3 \circ \overset{c}{a'}) \cdot (m_1 +_s m_2 +_s m_3)$.
2) The secret s_2 can only be revealed by decrypting the message using the key k_1 followed by the key k_2 as:
   $(k_2 \circ \overset{c}{a'}) \cdot ((k_1 \circ \overset{c}{a'}) \cdot ((k_1 \circ a) \cdot ((k_2 \circ a) \cdot s_2))) = s$
3) The secret s_3 encrypted using Bob key cannot be produced from the message m_3. $\neg\, \exists (x \mid x \in E : (k_2 \circ a) \cdot s_3 \leq_s x \cdot m_3)$

Note that the third property can be generalized to any key and any secret from any message. Such generalization would increase the complexity of the analysis. However, our goal in this paper is presenting attacks algebraically on object-based schemes which can be achieved through the presented properties.

We can use our framework to verify the satisfaction of properties. The first property is a correctness property. We were able to verify this property by assuming that $k_3 \neq 0_k$ since the 0_k could not reveal any secret. Also, we were able to prove the second property by having $k_1 \neq 0_k$ and $k_2 \neq 0_k$. Also, to enforce the order of decryption to obtain the secret, we should assume that $k_1 \neq k_2$. The third property states that we cannot obtain s_3 encrypted using k_2 from the message m_3. However, the message m_3 consists of s_3 encrypted using k_3 and consists of s_3 encrypted using $k_1 *_k k_2$. Therefore, to prove the third property, we should assume that $k_2 \neq k_3$ and combining the keys k_1 and k_2 does not produce k_2 i.e., $k_1 *_k k_2 \neq k_2$.

V. Algebraic Models

In this section, we introduce four concrete algebraic models for the algebraic structures presented in Section III-B. These models act as an intermediate level between the abstract level and the implementation level as shown in Figure 1 similar to [12]. Also, these models may include additional algebraic properties that affect the analysis of object-based schemes. We introduce algebraic models for Vigenère, transposition cipher, DES and RSA cryptosystems by giving concrete meaning to secrets, keys, and ciphers. In all these cryptosystems, the envelope operator is represented as applying a key to a cipher, while the operator in the message structure is represented as using an envelope to encrypt or decrypt a secret.

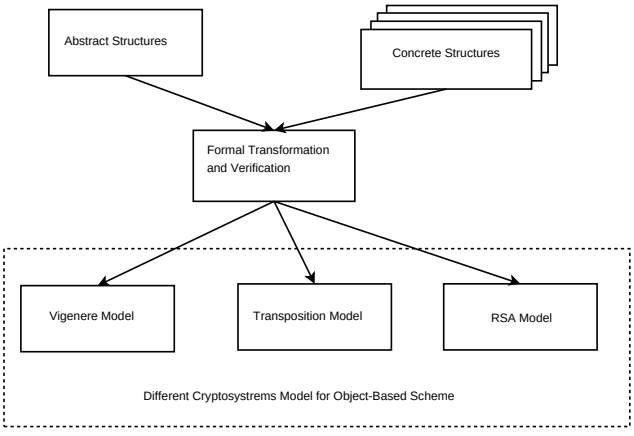

Figure 1. A diagram that shows the relation between the abstract and concrete models

A. Vigenère

Vigenère is a substitution cipher such that each alphabet in the plain-text is substituted with another alphabet based on a key. There are two different ways to represent Vigenère algebraically. One of them as a relation and the other as a function. Here we show the functional representation since it is more common. First, we give an example and then we introduce an algebraic model for Vigenère. A secret is a string. For example it can be the word *security*. The key is also a string e.g., the word *test*. Each character is encrypted individually by using the function $c_i = f^{-1}(f(s_i) + f(k_i) \pmod{26})$ where $f(n)$ gives the numerical value of the character n. For example, $f(a) = 0$, $f(b) = 1$, and so on. Note that the key should be repeated if it is shorter than the secret. The produced cipher-text from the given secret and key is "*liunkmlr*". A generalization to this representation can consider each of a secret and a key as a set of strings.

Proposition 5.1: Let $\mathcal{S} = (S, +_s, *_s, 0_s)$, where $S \triangleq \mathcal{P}(A)$ is a set of string. The operator $P +_s Q \triangleq P \cup Q$ is a set union. The operator $P *_s Q \triangleq \{pq \mid p \in P \ \wedge \ q \in Q\}$ is the concatenation of strings, $0_s \triangleq \emptyset$ is the empty set. The structure $\mathcal{S}$ is a secret structure.

Proof: The union $\cup$ is commutative, associative, and idempotent. Its identity element is $\emptyset$. String concatenation is associative. The operator $*_s$ distributes over $+_s$ because of the distributivity of $\wedge$ over $\vee$. Finally, $\emptyset$ is an annihilator for $*_s$ as $x \in \emptyset \Leftrightarrow \mathsf{false}$ and $\{x \mid \mathsf{false}\} = \emptyset$. ■

Proposition 5.2: Let $\mathcal{K} = (K, +_k, *_k, 0_k)$, where $K \triangleq \mathcal{P}(A)$ is a set of strings. The operator $P +_s Q \triangleq P \cup Q$ is a set union. The operator $P *_k Q \triangleq \{f^{-1}(f(p_i) + f(q_i) \pmod{26}) \mid p \in P \ \wedge \ q \in Q\}$ is the addition of two strings, $0_k \triangleq \emptyset$ is the empty set. The structure $\mathcal{K}$ is a key structure.

Proof: The union $\cup$ is commutative, associative, and idempotent. Its identity element is $\emptyset$. Number addition is commutative and associative. The operator $*_s$ distributes over $+_s$ because of the distributivity of $\wedge$ over $\vee$. Finally, $\emptyset$ is an annihilator for $*_s$ as $x \in \emptyset \Leftrightarrow \mathsf{false}$ and $\{x \mid \mathsf{false}\} = \emptyset$. ■

The key is also defined as a set of strings. We define the $+_k$ operator as the set union while the operator $*_k$ the addition of the alphabet values. For example $\{guke\} *_k \{njip\} = \{test\}$ i.e., $f(g) = 6$, $f(n) = 13$, and $f^{-1}(19) = t$ and so on for the other characters.

Proposition 5.3: Let $\mathcal{C} \stackrel{\text{def}}{=} (C, *_c, +_c, 1_c, c^{'}, 0_c)$, where C is a set of Vigenère ciphers i.e., $c(s) = s + k$. Let $c^{'}(s) = s - k$, $P *_c Q \triangleq \{p; q \mid p \in P \ \wedge \ q \in Q\}$ where ; is a function composition, the operator $+_c$ be set union, the operator 1_c be the identity function, and the operator 0_c be the empty set. The structure $\mathcal{C}$ is a cipher structure.

Proof: The union $\cup$ is commutative, associative, and idempotent. Its identity element is $\emptyset$. $c(c^{'}(s)) = s + k - k = s$. The composition of functions is associative and its identity element is the identity function. The operator $*_s$ distributes over $+_s$ because of the distributivity of $\wedge$ over $\vee$. Finally, $\emptyset$ is an annihilator for $*_s$ as $x \in \emptyset \Leftrightarrow \mathsf{false}$ and $\{x \mid \mathsf{false}\} = \emptyset$. ■

A cipher is a function that encrypts or decrypts a unit of a text. The identity cipher is the identity function $c(s) = s$. The $+_c$ is the set union, and 0_c is the empty set. It has been proved in the literature that Vigenère has the following algebraic properties:

1) $e_1 *_c e_2 = e_2 *_c e_1$
2) $(k_1 \circ c) *_c (k_2 \circ c) = (k_1 *_k k_2) \circ c$.
3) $e^n = 1_c$
4) $1_k *_k k = k *_k 1_k = k$

where e_1, e_2 are envelopes, k_1, k_2 are keys, and c is a cipher. We have:

The first identity states that Vigenère is commutative. Identity 2 states that double encryption in Vigenère is equivalent to a single encryption with a combined key. Identity 3 states that Vigenère is cyclic i.e., encrypting a message 26 times (the number of alphabets) using the same keyword produces the same secret. Identity 4 states the existence of an identity key 1_k i.e., a string of a's since $f(a) = 0$.

Vigenère satisfies the properties of algebraic structures presented in Section III-B. However, as shown above, Vigenère has additional properties. These properties affect the analysis of object-based schemes. When we add these properties to the analysis of the illustrative example, we found that the properties 2 and 3 cannot be verified. Therefore, the scheme contains flaws as shown in the following scenarios.

Scenario of an Attack: The second property in the illustrative example states that Alice should decrypt the message first and then Bob in order to get the secret s_2. Bob can get the secret but it needs the approval of Alice. By having the commutative property of ciphers, we cannot state an order on the decryption process. For example, decrypting using Bob key and then Alice reveals the secret.

$$
\begin{aligned}
& (k_1 \circ a^{c'}) \cdot ((k_2 \circ a^{c'}) \cdot ((k_1 \circ a) \cdot ((k_2 \circ a) \cdot s_2))) \\
= & \quad \langle \text{ Definition 3.14 } \rangle \\
& ((k_1 \circ a^{c'}) *_c (k_2 \circ a^{c'}) *_c (k13 \circ a) *_c (k_2 \circ a)) \cdot s_2 \\
= & \quad \langle \text{ Vigenère is commutative } \rangle \\
& ((k_2 \circ a^{c'}) *_c (k_1 \circ a^{c'}) *_c (k_1 \circ a) *_c (k_2 \circ a)) \cdot s_2 \\
= & \quad \langle \text{ Definition 3.10 and Definition 3.11 } \rangle \\
& ((k_2 \circ a^{c'}) *_c (k_2 \circ a)) \cdot s_2 \\
= & \quad \langle \text{ Definition 3.10 and Definition 3.11 } \rangle \\
& 1_c \cdot s_2 \\
= & \quad \langle \text{ Definition 3.14 } \rangle \\
& s_2
\end{aligned}
$$

The above derivation shows that decrypting the message m_2 with an order other that the specified one in the second property could reveal the secret s_2 which violate the second property.

Scenario of an Attack: Assume that there is a trusted system that takes the two keys, combines them and then

reveals the secret s_3 to each one of them. An intruder playing the role of Bob can trick Alice by sending the identity key 1_k instead of his key. In this case, the trusted server sends the secret s_3, encrypted using the key of Bob, to Alice and Bob. Therefore, Alice cannot get the secret as shown in the derivation below.

$$
\begin{array}{ll}
 & ((k_1 *_k 1_k) \circ a^{\overset{c}{\prime}}) \cdot (((k_1 *_k k_2) +_k k_3) \circ a) \cdot s_3) \\
= & \quad \langle \text{ A property of an identity key } \rangle \\
 & (k_1 \circ a^{\overset{c}{\prime}}) \cdot ((((k_1 *_k k_2) +_k k_3) \circ a) \cdot s_3) \\
= & \quad \langle \text{ Definition 3.12 } \rangle \\
 & (k_1 \circ a^{\overset{c}{\prime}}) \cdot (((k_1 *_k k_2) \circ a +_c k_3 \circ a) \cdot s_3) \\
= & \quad \langle \text{ Identity of Vigenère } \rangle \\
 & (k_1 \circ a^{\overset{c}{\prime}}) \cdot (((k_1 \circ a *_c k_2 \circ a) +_c k_3 \circ a) \cdot s_3) \\
= & \quad \langle \text{ Definition 3.14 } \rangle \\
 & (k_1 \circ a^{\overset{c}{\prime}}) \cdot ((k_1 \circ a *_c k_2 \circ a) \cdot s_3 +_s (k_3 \circ a) \cdot s_3) \\
= & \quad \langle \text{ Definition 3.14 } \rangle \\
 & (k_1 \circ a^{\overset{c}{\prime}} *_c k_1 \circ a *_c k_2 \circ a) \cdot s_3 +_s \\
 & (k_1 \circ a^{\overset{c}{\prime}} *_c k_3 \circ a) \cdot s_3 \\
= & \quad \langle \text{ Definition 3.11 } \rangle \\
 & (k_1 \circ (a^{\overset{c}{\prime}} *_c a) *_c k_2 \circ a) \cdot s_3 +_s \\
 & (k_1 \circ a^{\overset{c}{\prime}} *_c k_3 \circ a) \cdot s_3 \\
= & \quad \langle \text{ Definition 3.10 } \rangle \\
 & (k_1 \circ 1_c *_c k_2 \circ a) \cdot s_3 +_s (k_1 \circ a^{\overset{c}{\prime}} *_c k_3 \circ a) \cdot s_3 \\
= & \quad \langle \text{ Definition 3.11 } \rangle \\
 & (1_c *_c k_2 \circ a) \cdot s_3 +_s (k_1 \circ a^{\overset{c}{\prime}} *_c k_3 \circ a) \cdot s_3 \\
= & \quad \langle \text{ Definition 3.10 } \rangle \\
 & (k_2 \circ a) \cdot s_3 +_s (k_1 \circ a^{\overset{c}{\prime}} *_c k_3 \circ a) \cdot s_3
\end{array}
$$

The obtained message when Bob sends 1_k instead of his key is $(k_2 \circ a) \cdot s_3$ and $(k_1 \circ a^{\overset{c}{\prime}} *_c k_3 \circ a) \cdot s_3$. The message $(k_2 \circ a) \cdot s_3$ indicates that the secret s_3 is encrypted using the key of Bob k_2. Therefore, Bob can obtain the secret but not Alice. Alice cannot obtain the secret from other message $(k_1 \circ a^{\overset{c}{\prime}} *_c k_3 \circ a) \cdot s_3$ since it needs the key of Carol k_3. Therefore, Bob can get the secret but Alice cannot.

B. Transposition Ciphers

Transposition ciphers change the position of characters in the plain-text to produce a cipher-text, while the identity of characters remains the same. We give an example of a transposition cipher and a mathematical representation of its three elements: secret, key and cipher.

Assume that the secret to be encrypted is the word *security*. We represent this word as the relation $\{(s,1),(e,2),(c,3),(u,4),(r,5),(i,6),(t,7),(y,8)\}$. The key gives the permutation of the string. A key can be given as $\{(1,3),(2,5),(3,2),(4,6),(5,1),(6,8),(7,4),(8,7)\}$ which indicates that the first alphabet of the secret should be moved into the third position, the second alphabet into the fifth position and so on. The key should be a bijective function with a length equal to that of the secret. However, if the secret is longer, it can be divided into blocks, but if the key is longer we expand the secret with extra characters. The cipher can be represented as the relational composition between the secret and the cipher as $s;k = \{(s,3),(e,5),(c,2),(u,6),(r,1),(i,8),(t,4),(y,7)\}$. Therefore, the produced cipher-text is "*rcsteuyi*". The decryption function is $s;k^{\cup}$ where $k^{\cup}$ is the converse of the relation k. Our algebraic structures enable us to represent a generalisation to the transposition ciphers. It allows handling a set of secrets and a set of keys as shown in the following propositions.

Proposition 5.4: Let $S \triangleq \mathcal{P}(\mathbb{A} \times \mathcal{N})$, where $\mathbb{A}$ is a set of alphabet and $\mathcal{N}$ is the set of natural numbers. Let the operator $P +_s Q \triangleq P \cup Q$ be a set union, the operator $P *_s Q \triangleq \{pq \mid p \in P \ \wedge \ q \in Q\}$ be the concatenation of strings, and $0_s \triangleq \emptyset$ be the empty set. The structure $\mathcal{S} = (S, +_s, *_s, 0_s)$ is a secret structure.

Proof: The proof is similar to the proof of Proposition 5.1. ■

In this model, a secret is a set of strings each is represented as a relation. The operator $+_s$ indicates set union, while the operator $*_s$ indicates the concatenation of strings represented as relations.

Proposition 5.5: Let $K \triangleq \mathcal{P}(\mathcal{N} \times \mathcal{N})$ where $\mathcal{N}$ is the set of natural numbers. Let the operator $P +_s Q \triangleq P \cup Q$ be a set union, the operator $P *_s Q \triangleq \{p \cap q \mid p \in P \ \wedge \ q \in Q\}$ be the intersection of relations, and $0_s \triangleq \emptyset$ be the empty set. The structure $\mathcal{K} = (K, +_k, *_k, 0_k)$ is a key structure.

Proof: The union $\cup$ is commutative, associative, and idempotent. Its identity element is $\emptyset$. Set intersection $\cap$ is commutative and associative. The operator $*_s$ distributes over $+_s$ because of the distributivity of $\wedge$ over $\vee$. Finally, $\emptyset$ is an annihilator for $*_s$ as $x \in \emptyset \Leftrightarrow \mathsf{false}$ and $\{x \mid \mathsf{false}\} = \emptyset$. ■

Our key model allows representing a set of keys and handling two operators. The operator $+_k$ is the set union and the operator $*_k$ is the set intersection. Note that this is one representation of keys. There could be other representations that satisfy the properties of the key structure.

Proposition 5.6: Let $\mathcal{C} \overset{\text{def}}{=} (C, *_c, +_c, 1_c, \overset{c}{\prime}, 0_c)$, where C is a set of transposition ciphers i.e., $c(s) = s;k$. Let $c^{\overset{c}{\prime}}(s) = s;k^{\cup}$, $P *_c Q \triangleq \{p;q \mid p \in P \ \wedge \ q \in Q\}$ where ; is a relation composition, the operator $+_c$ be set union, the operator 1_c be the identity relation, and the operator 0_c be the empty set. The structure $\mathcal{C}$ is a cipher structure.

Proof: The union $\cup$ is commutative, associative, and idempotent. Its identity element is $\emptyset$. $c *_c c^{\overset{c}{\prime}} = s;k;k^{\cup} = s$ and $c^{\overset{c}{\prime}} *_c c = s;k^{\cup};k = s$. The composition of relations is associative and its identity element is the identity relation. The operator $*_s$ distributes over $+_s$ because of the distributivity of $\wedge$ over $\vee$. Finally, $\emptyset$ is an annihilator for $*_s$ as $x \in \emptyset \Leftrightarrow \mathsf{false}$ and $\{x \mid \mathsf{false}\} = \emptyset$. ■

Transposition ciphers do not have extra algebraic properties that could affect the security properties of object-based scheme presented in the example. However, we do

not claim in this paper that transposition cipher is secure. It is known that transposition ciphers is vulnerable to attacks based on frequency analysis. Our focus of this paper is analysing the effects of the algebraic properties of cryptosystems on object-based schemes. We do not discus the security of cryptosystems themselves.

C. Data Encryption Standard (DES)

DES is a shared key cryptosystem such that the key used in encryption is the same used in decryption. DES is a block cipher that takes a block of plain-text as input and produces a cipher-text of the same length. DES is based on sixteen rounds. Each round consists of four stages that perform expansion, key mixing, permutation, and substitution. In our representation, we denote the encryption algorithm by c and the decryption algorithm by $c^{'}$. We represent secrets and keys as sets of numbers.

Proposition 5.7: Let $S \triangleq \mathcal{P}(\mathcal{N})$, where $\mathcal{N}$ is the set of natural numbers. Let the operator $P +_{s} Q \triangleq P \cup Q$ be a set union, the operator $P *_{s} Q \triangleq \{pq \mid p \in P \ \wedge \ q \in Q\}$ be the concatenation of strings represented as numbers, and $0_{s} \triangleq \emptyset$ be the empty set. The structure $\mathcal{S} = (S, +_{s}, *_{s}, 0_{s})$ is a secret structure.

Proof: The proof is similar to the proof of Proposition 5.1. ∎

Proposition 5.8: Let $K \triangleq \mathcal{P}(\mathcal{N})$, where $\mathcal{N}$ is the set of natural numbers. Let the operator $P +_{s} Q \triangleq P \cup Q$ be a set union, the operator $P *_{s} Q \triangleq \{p \oplus q \mid p \in P \ \wedge \ q \in Q\}$ be the multiplication of keys, and $0_{s} \triangleq \emptyset$ be the empty set. The structure $\mathcal{K} = (K, +_{k}, *_{k}, 0_{k})$ is a key structure.

Proof: The union $\cup$ is commutative, associative, and idempotent. Its identity element is $\emptyset$. The *xor* operator $\oplus$ is commutative and associative. The operator $*_{s}$ distributes over $+_{s}$ because of the distributivity of $\wedge$ over $\vee$. Finally, $\emptyset$ is an annihilator for $*_{s}$ as $x \in \emptyset \Leftrightarrow \mathsf{false}$ and $\{x \mid \mathsf{false}\} = \emptyset$. ∎

Proposition 5.9: Let C be a set of DES algorithms, the operator $*_{c}$ represents running algorithms consecutively, the operator $+_{c}$ be set union, the operator 1_{c} be an algorithm that has not effect, and the operator 0_{c} be the empty set.

Proof: Set union $\cup$ is commutative, associative and idempotent with an identity 0_{c}. The operator $*_{c}$ is associative with an identity 1_{c}. ∎

Scenario of an Attack: The presented key model has the following property

$$k *_{k} k = 1_{k}$$

This property leads to the following derivation.

$$\begin{aligned} & (((k_1 *_{k} k_1) +_{k} k_3) \circ a) \cdot s_3 \\ = & \quad \langle \text{ A property of the key model } \rangle \\ & ((1_{k} +_{k} k_3) \circ a) \cdot s_3 \\ = & \quad \langle \text{ Definition 3.12 } \rangle \\ & (1_{k} \circ a +_{c} k_3 \circ a) \cdot s_3 \\ = & \quad \langle \text{ Definition 3.14 } \rangle \\ & (1_{k} \circ a) \cdot s_3 +_{s} (k_3 \circ a) \cdot s_3 \\ = & \quad \langle \text{ Definition 3.11 } \rangle \\ & 1_{c} \cdot s_3 +_{s} (k_3 \circ a) \cdot s_3 \\ = & \quad \langle \text{ Definition 3.14 } \rangle \\ & s_3 +_{s} (k_3 \circ a) \cdot s_3 \end{aligned}$$

By having the $k_1 = k_2$, we can reveal the secret which violates the secrecy property. Therefore, we should assume that $k_1 \neq k_2$.

Scenario of an Attack: DES has the following property, when the keys k_1 and k_2 are semi-weak keys.

$$(k_1 \circ a)((k_2 \circ a) \cdot s) = s$$

This property affects the satisfaction of the second property of the illustrative example as shown in the following derivation

$$\begin{aligned} & (k_3 \circ a) \cdot ((k_2 \circ a) \cdot s_2) \\ = & \quad \langle \text{ A property of semi-weak keys } \rangle \\ & s \end{aligned}$$

Therefore, the secret s_2 would be revealed without decrypting the message using the keys of Alice and Bob. Therefore, to prove the property, we should make sure that the used keys are not semi-weak.

D. RSA

RSA is a public key cryptosystem whose key consists of two parts: a public part (e, n) used for encryption, and a private part (d, n) used for decryption. The encryption of a secret s is performed as $s^{e} \mod n$ while the decryption of a message m as $m^{d} \mod n$.

Proposition 5.10: Let $S \triangleq \mathcal{P}(\mathcal{N})$, where $\mathcal{N}$ is the set of natural numbers. Let the operator $P +_{s} Q \triangleq P \cup Q$ be a set union, the operator $P *_{s} Q \triangleq \{pq \mid p \in P \ \wedge \ q \in Q\}$ be the concatenation of strings represented as numbers, and $0_{s} \triangleq \emptyset$ be the empty set. The structure $\mathcal{S} = (S, +_{s}, *_{s}, 0_{s})$ is a secret structure.

Proof: The proof is similar to the proof of Proposition 5.1. ∎

In this model, a secret is a set of numbers. The operator $+_{s}$ indicates set union while the operator $*_{s}$ indicates the concatenation of strings.

Proposition 5.11: Let $K \triangleq \mathcal{P}(\{(a, b, m) \mid a, b \in \mathbb{N}, m \in M \ \wedge \ 1 < a < \phi(m) \ \wedge \ gcd(a, \phi(m)) = 1 \ \wedge \ a \times b \equiv 1 \pmod{\phi(m)}\})$ where $\mathcal{N}$ is the set of natural numbers. Let the operator $P +_{s} Q \triangleq P \cup Q$ be a set union, the operator $P *_{s} Q \triangleq \{(e, f, m) \mid \exists((a, c, m), (b, d, m) \mid (a, c, m) \in P \ \wedge \ (b, d, m) \in Q \ : \ e = a \times b \pmod{\phi(m)} \ \wedge \ f = c \times d \pmod{\phi(m)})\}$ be the multiplication of keys, and $0_{s} \triangleq \emptyset$ be the empty set. The structure $\mathcal{K} = (K, +_{k}, *_{k}, 0_{k})$ is a key structure.

Proof: The union $\cup$ is commutative, associative, and idempotent. Its identity element is $\emptyset$. Number multiplication is commutative and associative. The operator $*_{s}$

distributes over $+_s$ because of the distributivity of $\wedge$ over $\vee$. Finally, $\emptyset$ is an annihilator for $*_s$ as $x \in \emptyset \Leftrightarrow \mathsf{false}$ and $\{x \mid \mathsf{false}\} = \emptyset$. ■

An RSA key consists of three parts: e, d, n. In our structure, a key is a set of RSA keys. The operator $+_k$ is a set union, while the operator $*_k$ represents combining keys that have the same value n by multiplying together the values of d and the values of e.

Proposition 5.12: Let C be a set of RSA ciphers, the operator $*_c$ be a function composition, the operator $+_c$ be set union, the operator 1_c be the identity function, and the operator 0_c be the empty set.

Proof: The proof uses properties of set and function theories. ■

It has been proved in the literature that RSA has the homomorphism algebraic property

$$e \cdot s_1 *_s e \cdot s_1 = e \cdot (s_1 *_s s_2)$$

By analysing the object-based scheme presented in the illustrative example, we find that this property does not affect its security property.

VI. Conclusion

In this paper, we extend the work of [1] in analysing object-based schemes algebraically in two directions. First, we present concrete algebraic models for Vigenère, transposition ciphers, DES and RSA cryptosystems. These models act as an intermediate level between the abstract level and the implementation. Second, we show the effects of the additional algebraic properties of those concrete models on the security properties of object-based schemes. We find that some algebraic properties of cryptosystems such as commutativity could affect the security of object-based schemes based on the scheme and the security properties.

As a future work, we aim at building a tool that can be used for automatically verify security policies. The input to the tool are security properties and an object based scheme. The security properties can be represented as logic formulas, while the object based schemes are set of policies that can be represented as terms of the algebraic structures presented in this paper. The tool should store the algebraic properties of several cryptosystems. By using prover9, the tool should be able to indicate automatically which property is satisfied and which one is not satisfied. Such a tool would enhance the security of systems implementing access control based on object-based schemes.

Appendix

A. Axioms of the Envelope Structure

1) $\forall(a,k,l \mid a \in C, k,l \in K : (k+_k l)a = ka +_c la)$
2) $\forall(a,b,k \mid a,b \in C, k \in K : k(a +_c b) = ka +_c kb)$
3) $\forall(a,b,k \mid a,b \in C, k \in K : k(a *_c b) = ka *_c kb)$
4) $\forall(a,k,l \mid a \in C, k,l \in K : l(ka) = (l *_k k)a)$
5) $\forall(a \mid a \in C : 0_k a = 0_c)$
6) $\forall(k \mid k \in K : k0_c = 0_c)$
7) $\forall(k \mid k \in K - \{0_c\} : k1_c = 1_c)$
8) $\forall(k,a \mid k \in K, a \in A : ka'^{c} = (ka)'^{c})$
9) $\forall(a,k,l \mid a \in C, k,l \in K : (k +_k l)a = (l +_k k)a)$
10) $\forall(a,k,l,h \mid a \in C, k,l,h \in K : ((k +_k l) +_k h)a = (k +_k (l +_k h))a)$
11) $\forall(a,k \mid a \in C, k \in K : (k +_k k)a = ka)$
12) $\forall(a,k,l \mid a \in C, k,l \in K : (k *_k l)a = (l *_k k)a)$
13) $\forall(a,k,l,h \mid a \in C, k,l,h \in K : ((k *_k l) *_k h)a = (k *_k (l +_k h))a)$
14) $\forall(a,k,l \mid a \in C, k \in K : (k *_k 0_k)a = 0_k a)$
15) $\forall(a,k,l \mid a \in C, k \in K : (k +_k 0_k)a = ka)$
16) $\forall(k,l,h,a \mid k,l,h \in K, a \in C : (k *_k (l +_k h))a = ((k *_k l) +_k (k *_k h))a)$
17) $\forall(k,l,h,a \mid k,l,h \in K, a \in C : ((k +_k l) *_k h)a = ((k *_k h) +_k (l *_k h))a)$
18) $\forall(a,b,k \mid a,b \in C, k \in K : k(a +_c b) = k(b +_c a))$
19) $\forall(a,b,k \mid a,b,c \in C, k \in K : k((a +_c b) +_c c) = k(a +_c (b +_c c)))$
20) $\forall(a,b,k \mid a,b,c \in C, k \in K : k((a *_c b) *_c c) = k(a *_c (b *_c c)))$
21) $\forall(a,k \mid a \in C, k \in K : k(a +_c a) = ka)$
22) $\forall(a,k \mid a \in C, k \in K : k(a +_c 0_c) = ka)$
23) $\forall(a,k \mid a \in C, k \in K : k(a *_c 0_c) = k *_c 0_c)$
24) $\forall(a,k \mid a \in C, k \in K : k(a *_c a') = k1_c)$
25) $\forall(a,k \mid a \in C, k \in K : k(a *_c 1_c) = ka)$
26) $\forall(a,k \mid a \in C, k \in K : k(1_c *_c a) = ka)$
27) $\forall(k,a,b,c \mid k \in K, a,b,c \in C : k(a *_c (b +_c c)) = k((a *_c b) +_c (a *_c c)))$
28) $\forall(k,a,b,c \mid k \in K, a,b,c \in C : k((a +_c b) *_c c) = k((a *_c c) +_c (b *_c c)))$

B. Axioms of the Message Structure

1) $\forall(a,b,s \mid a,b \in C, s \in S : (a +_c b)s = as +_s bs)$
2) $\forall(a,r,s \mid a \in C, r,s \in S : a(r +_s s) = ar +_s as)$
3) $\forall(a,b,s \mid a,b \in C, s \in S : a(bs) = (a *_c b)s)$
4) $\forall(s \mid s \in S : 1_c s = s)$
5) $\forall(s \mid s \in S : 0_c s = 0_s)$
6) $\forall(a \mid a \in C : a0_s = 0_s)$
7) $\forall(a,b,s \mid a,b \in C, s \in S : (a +_c b)s = (b +_c a)s)$
8) $\forall(a,b,c,s \mid a,b,c \in C, s \in S : ((a +_c b) +_c c)s = (a +_c (b +_c c))s)$
9) $\forall(a,s \mid a \in C, s \in S : (a +_c a)s = as)$
10) $\forall(a,s \mid a \in C, s \in S : (a +_c 0_c)s = as)$
11) $\forall(a,b,c,s \mid a,b,c \in C, s \in S : ((a *_c b) *_c c)s = (a *_c (b *_c c))s)$
12) $\forall(a,s \mid a \in C, s \in S : (a *_c 1_c)s = as)$
13) $\forall(a,s \mid a \in C, s \in S : (a *_c a')s = 1_c s)$
14) $\forall(a,s \mid a \in C, s \in S : (a *_c 0_c)s = 0_c s)$
15) $\forall(a,b,c,s \mid a,b,c \in C, s \in S : (a *_c (b +_c c))s = ((a *_c b) +_c (a *_c c))s)$
16) $\forall(a,b,c,s \mid a,b,c \in C, s \in S : ((a +_c b) *_c c)s = ((a *_c c) +_c (b *_c c))s)$
17) $\forall(a,s,r \mid a \in C, s,r \in S : a(s +_s r) = a(r +_s s))$

18) $\forall(a,s,r,t \mid a \in C, s,r,t \in S : a((s +_s r) +_s t) = a(s +_s (r +_s t)))$
19) $\forall(a,s \mid a \in C, s \in S : a(s +_s 0_s) = as)$
20) $\forall(a,s \mid a \in C, s \in S : a(s +_s s) = as)$
21) $\forall(a,s,r,t \mid a \in C, s,r,t \in S : a((s *_s r) *_s t) = a(s *_s (r *_s t)))$
22) $\forall(a,s,r,t \mid a \in C, s,r,t \in S : a(s *_s (r +_s t)) = a((s *_s r) +_s (s *_s t)))$
23) $\forall(a,s,r,t \mid a \in C, s,r,t \in S : a((s +_s r) *_s t) = a((s *_s t) +_s (r *_s t)))$

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, pp. 305–335, 2011.

[2] J. Crampton, "Specifying and enforcing constraints in role-based access control," in *Proceedings of the eighth ACM symposium on Access control models and technologies*, ser. SACMAT '03. New York, NY, USA: ACM, 2003, pp. 43–50.

[3] G. Bruns and M. Huth, "Access-control policies via belnap logic: Effective and efficient composition and analysis," in *Proceedings of the 2008 21st IEEE Computer Security Foundations Symposium*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 163–176.

[4] K. E. Sabri, R. Khedri, and J. Jaskolka, "Verification of information flow in agent-based systems," in *Proceedings of the 4th MCETECH Conference on e-Technologies*, ser. Lecture Notes in Business Information Processing, G. Babin, P. Kropf, and M. Weiss, Eds., vol. 26. Springer-Verlag Berlin Heidelberg, May 2009, pp. 252–266.

[5] S. Akl and P. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," *ACM Transaction on Computer Systems*, vol. 1, no. 3, pp. 239–248, 1983.

[6] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, "Dynamic and efficient key management for access hierarchies," *ACM Transactions on Information and System Security*, vol. 12, no. 3, pp. 1–43, 2009.

[7] K. E. Sabri and R. Khedri, "A generic algebraic model for the analysis of cryptographic-key assignment schemes," in *5th International Symposium on Foundations and Practice of Security (FPS)*, ser. Lecture Notes in Computer Science, vol. 7743. Springer-Verlag, 2013, pp. 62–77.

[8] P. Lafourcade, D. Lugiez, and R. Treinen, "Intruder deduction for the equational theory of abelian groups with distributive encryption," *Information and Computation*, vol. 205, no. 4, pp. 581–623, April 2007.

[9] S. Bursuc and H. Comon-Lundh, "Protocol security and algebraic properties: Decision results for a bounded number of sessions," in *Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA)*, ser. Lecture Notes in Computer Science, vol. 5595. Springer-Verlag, 2009, pp. 133–147.

[10] V. Cortier, S. Delaune, and P. Lafourcade, "A survey of algebraic properties used in cryptographic protocols," *Journal of Computer Security*, vol. 14, no. 1, pp. 1–43, January 2006.

[11] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 1996.

[12] R. Khedri, "Formal model driven approach to deal with requirements volatility," McMaster University, Tech. Rep. CAS-08-03-RK, 2008.

[13] H. T. Liaw, S. J. Wang, and C. L. Lei, "A dynamic cryptographic key assignment scheme in a tree structure," *Computers & Mathematics with Applications*, vol. 25, no. 6, pp. 109–114, March 1993.

[14] R. S. Sandhu, "On some cryptographic solutions for access control in a tree hierarchy," in *ACM '87: Proceedings of the 1987 Fall Joint Computer Conference on Exploring technology: today and tomorrow*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1987, pp. 405–410.

[15] G. Ateniese, A. D. Santis, A. L. Ferrara, and B. Masucci, "A note on time-bound hierarchical key assignment schemes," *Information Processing Letters*, vol. 113, no. 5-6, pp. 151–155, 2013.

[16] J. Crampton, K. Martin, and P. Wild, "On key assignment for hierarchical access control," in *Proceedings of the 19th IEEE workshop on Computer Security Foundations (CSFW '06)*. Venice, Italy: IEEE Computer Society, 2006, pp. 98–111.

[17] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "A data outsourcing architecture combining cryptography and access control," in *Proceedings of the 2007 ACM workshop on Computer security architecture*, ser. CSAW '07. New York, NY, USA: ACM, 2007, pp. 63–69.

[18] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, ser. CCSW '09. New York, NY, USA: ACM, 2009, pp. 55–66.

[19] G. Miklau and D. Suciu, "Controlling access to published data using cryptography," in *Proceedings of the 29th international conference on Very large data bases - Volume 29*, ser. VLDB '03. VLDB Endowment, 2003, pp. 898–909.

[20] M. Abadi and B. Warinschi, "Security analysis of cryptographically controlled access to xml documents," in *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, ser. PODS '05. New York, NY, USA: ACM, 2005, pp. 108–117.

[21] M. Baudet, V. Cortier, and S. Delaune, "Yapa: A generic tool for computing intruder knowledge," *ACM Transactions on Computational Logic*, vol. 14, no. 1, pp. 4:1–4:32, Feb. 2013.

[22] E. M. Clarke, S. Jha, and W. Marrero, "Verifying security protocols with Brutus," *ACM Transactions on Software Engineering and Methodology*, vol. 9, no. 4, pp. 443–487, October 2000.

[23] I. Al-Azzoni, D. G. Down, and R. Khedri, "Modeling and verification of cryptographic protocols using Coloured Petri Nets and Design/CPN," *Nordic Journal of Computing*, vol. 12, no. 3, pp. 200–228, September 2005.

[24] S. Mödersheim, "Algebraic properties in alice and bob notation," in *Proceedings of the The Forth International Conference on Availability, Reliability and Security (ARES)*, ser. IEEE Computer Society, 2009, pp. 433–440.

[25] S. Erbatur, S. Escobar, D. Kapur, Z. Liu, C. Lynch, C. Meadows, J. Meseguer, P. Narendran, S. Santiago, and R. Sasse, "Effective symbolic protocol analysis via equational irreducibility conditions," in *17th European Symposium on Research in Computer Security (ESORICS)*, ser. Lecture Notes in Computer Science, vol. 7459, 2012, pp. 73–90.

[26] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, November 1979.

[27] C. Boyd, "Some applications of multiple key ciphers," in *Proceedings of Advances in Cryptology-EUROCRYPT'88*, ser. Lecture Notes in Computer Science, vol. 330. Springer-Verlag New York, Inc., May 1988, pp. 455–467.

[28] M. Renauld, F.-X. Standaert, and N. Veyrat-Charvillon, "Algebraic side-channel attacks on the aes: Why time also matters in dpa," in *11th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, ser. Lecture Notes in Computer Science, vol. 5747. Springer, 2009, pp. 97–111.

[29] N. Courtois, S. O'Neil, and J.-J. Quisquater, "Practical algebraic attacks on the hitag2 stream cipher," in *12th International Conference on Information Security (ISC)*, ser. Lecture Notes in Computer Science, vol. 5735. Springer, 2009, pp. 167–176.

[30] U. Hebisch and H. J. Weinert, *Semirings Algebraic Theory and Applications in Computer Science*. World Scientific Publishing Co. Pte. Ltd., 1993.

[31] W. McCune, "Prover9 and mace4," http://www.cs.unm.edu/ mccune/prover9/.

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.