Computational Soundness about Formal Encryption in the Presence of Secret Shares and Key Cycles*

Xinfeng Lei[1], Rui Xue[1], and Ting Yu[2]

[1] State Key Laboratory of Information Security
Institute of Software, Chinese Academy of Sciences, Beijing, China
leixinfeng@gmail.com, rxue@is.iscas.ac.cn
[2] Department of Computer Science, North Carolina State University, USA
yu@csc.ncsu.edu

Abstract. The computational soundness of formal encryption is studied extensively following the work of Abadi and Rogaway[1]. Recent work considers the scenario in which secret sharing is needed, and separately, the scenario when key cycles are present. The novel technique is the use of a co-induction definition of the adversarial knowledge. In this paper, we prove a computational soundness theorem of formal encryption in the presence of both key cycles and secret shares at the same time, which is a non-trivial extension of former approaches.

1 Introduction

There are mainly two approaches to security protocols analysis. One is based on formal models and the other is based on computational models. In the approach of formal models [2][3][4][5],

- messages are considered as formal expressions;
- the encryption operation is only an abstract function;
- security is modeled by formal formulas;
- and analysis of security is done by formal reasoning.

In the approach of computational models[6][7][8],

- messages are considered as bit-strings;
- the encryption operation of message is a concrete arithmetic;
- security is defined in terms of that a computationally bounded adversary can only attack successfully with negligible probability;
- and analysis of security is done by reduction.

Each of the approaches has its advantages and disadvantages. In general, the former is succinct but generally does not guarantee computational soundness. The

* This work is partially supported by NSFC grants (No. 60873260 and No. 60903210), the 863 Program (No. 2009AA01Z414), and the 973 Program (No. 2007CB311202).

S. Qing et al. (Eds.): ICICS 2011, LNCS 7043, pp. 29–41, 2011.

latter does exactly the opposite. From 1980"s, these two approaches developed along with their own directions independently. Till the beginning of this century, in their seminal work[1], Abadi and Rogaway started to bridge the gap between the two approaches, and established computational soundness for formal security analysis. Intuitively, in security analysis, computational soundness means that if two formal expressions are equivalent in formal model, their computational interpretations are indistinguishable in computational model. During the last decade, computational soundness has gained a lot of attention[1,9,10,11,12,13,14,15,16,17], and works in this area are still in full swing.

Our analysis is aimed at ensuring the computational soundness about formal encryption with the presence of secret shares and key cycles.

Secret Share. In a secret sharing scheme, a key may be separated into several secret shares, and only those who can get sufficient shares can reveal the key. Otherwise, nothing can be learned about this key. The concept of secret sharing was proposed in[18], and since then, it is used extensively in cryptography. Moreover, it can be used in other security applications. In [19], Miklau and Suciu implement access control policies in data publishing by using of cryptography(specifically, symmetric encryption and secret sharing). Using of secret sharing makes it more flexible to deploy the access control policy. What we concern about is whether a formal treatment of secret sharing keeps its computational soundness.

Key cycle. The concept of key cycles is first stated in [6], and then be noted since the work by Abadi and Rogway [1]. Informally, key cycle means that a key is encrypted by itself directly or indirectly. At the first glance, key cycles may not deserve so much attention due to few occurrences of them in a well-defined protocol. However, key cycles often happen in real world applications. For example, a backup system may store the key on disk and then encrypt the entire disk with the same key. Another example comes from the situation where a key cycle is needed 'by design"[20] in a system for non-transferable anonymous credentials. Moreover, key cycles paly an important role in solving the problem of computational soundness. In general, in a formal model, key cycles are allowed according to the definition of expressions[1] if there is no further restriction. While in a computational model, the occurrence of key cycles is often eliminated according to the standard notion of security for encryption [6]. This is the reason why key cycles gain so much attention in the research of computational soundness.

Related Work. In [1], Abadi and Rogaway give the definition of key cycles and then prove the computational soundness of security under formal setting in absence of key cycles. A natural problem is whether a formal encryption with key cycles is computationally sound. In recent years, this problem has been studied in many works[1,21,13,22,17]. In [21], Laud addresses the problem of reconciling symbolic and computational analysis in presence of key cycles by strengthening the symbolic adversary[21], i.e., weakening the symbolic encryption. Specifically, Laud uses an approach similar to that in [1] except giving adversaries the power to break the encryption with key cycles by adding some additional rules. In

[13,22], instead of using restricted or revised formal models, Adão et al. deal with key cycles by strengthening the computational notion. Specifically, Adão et al. adopt another security notion, i.e., Key-Dependent Message(KDM) security [23] in which the messages are chosen depending on the keys of the encryption scheme itself. Intuitively, different from the standard security notions(CPA or CCA), KDM security implies the security of key cycles and thus is closer to the concept of security in formal models. More and more works are focusing on constructing the KDM secure scheme[23,24,25], but most of them are given in the random-oracle model[23], or by a relaxed notion of KDM security[24], or under restricted adversaries[25]. Therefore, constructing such schemes is not an easy work. [26] shows that it is impossible to prove KDM security if the reductions in the proof of security treat both the adversary and the query function as black boxes. In this paper, we do not consider KDM security. Rather, our work is under CPA security.

In all the approaches mentioned above, when modeling the power of adversaries to obtain keys, an inductive method is used. Very recently, different from the inductive method, Micciancio [17] gives a general approach to dealing with the key cycles in which the power of the adversary to get keys is modeled by co-induction. The generalization of this approach makes it possible to deal with a larger class of cryptographic expressions, e.g., the expressions with pseudo-random keys [27]. Alternatively, in this paper, we will extend this approach to cryptographic expressions that use secret sharing schemes. Abadi and Warinschi [16] have given an approach to bridging the gap between formal and computational views in the presence of secret shares, but their approach does not consider key cycles.

Our contribution. The primary contribution of this paper is to show a stronger result over that in [16] and [17]. In particular, we define the equivalence between formal messages in the presence of both key cycles and secret shares, and then prove the computational soundness about formal encryption in this setting.

Organization. The rest of the paper is organized as follows. Section 2 presents the syntax of formal messages, patterns, and the notion of equivalence between messages. In section 3, the computational model is defined, and computational semantics of formal messages is given. Then, in section 4, the main result of this paper, theorem of computational soundness is provided and further discussion is given in section 5. Finally, we conclude in Section 6 and give the further works.

2 Formal Model

In this section we provide the basic notions for our work in a formal setting. We do this by summarizing the main definitions and results in previous papers [1,21,16,22,17] with some changes. Such changes are necessary because we take both key shares and key cycles into consideration.

2.1 Messages

In formal messages, anything is modeled by symbols. We use **Data** and **Keys** to denote the symbol sets of data, and keys respectively. Often, $d, d_1, d_2, \cdots$ range over **Data**, and $k, k_1, k_2, \cdots$ range over **Keys**. Assume a key can be divided in to several different secret shares, denoted by distinct symbols, we then define a set **Shares** as follows:

Assume a key can be divided into n secret shares, and k^j denotes the jth secret share of key k. Given $k \in \mathbf{Keys}$ and $\mathbf{K} \subseteq \mathbf{Keys}$, we can define[1],

- $\mathbf{s}(k) = \{k^j \mid j \in [1, n]\}$;
- $\mathbf{s}(\mathbf{K}) = \bigcup_{k \in \mathbf{K}} \mathbf{s}(k)$;
- $\mathbf{Shares} = \mathbf{s}(\mathbf{Keys})$.

For example, if $\mathbf{Keys} = \{k_1, k_2, k_3\}$ and $n = 2$, by dividing each key into two secret shares, we have $\mathbf{Shares} = \{k_1^1, k_1^2, k_2^1, k_2^2, k_3^1, k_3^2\}$.

The number of shares for every key n is the same. When a key is divided into n shares, we assume that, only when obtaining all the n shares will one be able to recover the key. One can learn nothing about the key with p shares where $p < n$.

Based on **Data**, **Keys** and **Shares**, we can define the set of messages **Msg** and the set of extended messages **MSG**.

$$\begin{aligned} \mathbf{Msg} &::= \mathbf{Data} \mid \mathbf{Keys} \mid \mathbf{Shares} \mid (\mathbf{Msg}, \mathbf{Msg}) \mid \{\!|\mathbf{Msg}|\!\}_{\mathbf{Keys}} \\ \mathbf{MSG} &::= \mathbf{Data} \cup \{\square\} \mid \mathbf{Keys} \cup \{\Diamond\} \mid \mathbf{Shares} \cup \{\Diamond^j\} \\ &\qquad \mid (\mathbf{MSG}, \mathbf{MSG}) \mid \{\!|\mathbf{MSG}|\!\}_{\mathbf{Keys} \cup \{\Diamond\}} \end{aligned}$$

Informally, (m_1, m_2) represents the concatenation of m_1 and m_2, and $\{|m|\}_k$ represents the encryption of m under k. $\square, \Diamond$ and $\Diamond^j$ denote the unknown data, keys and secret shares respectively.

To simplify our presentation, we will use some symbols of the first order logic in the following definition, and accept the following conventions of notations:

- $(m_1, m_2, \cdots, m_n) \triangleq (m_1, (m_2, \cdots, m_n))$;
- $\{\!|(m_1, m_2)|\!\}_k \triangleq \{\!|m_1, m_2|\!\}_k$;
- $\{\!|m|\!\}_\Diamond \triangleq \{\!|m|\!\}$.

Definition 1 (Sub-message). *Let $m, m' \in \mathbf{MSG}$. We say message m' is a sub-message of m, written as $m' \preccurlyeq m$, if one of the following holds:*

1. $m' = m$;
2. $m = (m_1, m_2) \wedge (m' \preccurlyeq m_1 \vee m' \preccurlyeq m_2)$;
3. $m = \{\!|m''|\!\}_k \wedge m' \preccurlyeq m''$.

Definition 2 (Occurrence). *Let $x \in \mathbf{Keys} \cup \mathbf{Shares}$ and $m \in \mathbf{MSG}$. x occurs in m, written as $x \lessdot m$, if one of the following holds:*

[1] By using $j \in [1, n]$, we mean $1 \leq j \leq n$.

1. $x = m$;
2. $m = (m_1, m_2) \wedge (x \lessdot m_1 \vee x \lessdot m_2)$;
3. $m = \{\!|m'|\!\}_k \wedge (x = k \vee x \lessdot m')$.

With Definition 2, we can define a function $\mathbf{keys} : \mathbf{MSG} \rightarrow \mathbf{Keys}$. Intuitively, $\mathbf{keys}(m)$ returns the set of keys that occur in a message or whose shares occur in this message. More formally, given $m \in \mathbf{MSG}$, we have

$$\mathbf{keys}(m) = \{k | (k \in \mathbf{Keys}) \wedge ((k \lessdot m) \vee \exists j \in [1, n].(k^j \lessdot m))\}.$$

Definition 3 (Encryption relation). *Let $m \in \mathbf{MSG}, k_1, k_2 \in \mathbf{keys}(m)$. We say k_1 encrypts k_2 in m, written as $k_1 \sqsubset_m k_2$, if there exists a message m' such that $(\{\!|m'|\!\}_{k_1} \preccurlyeq m) \wedge (k_2 \in \mathbf{keys}(m'))$.*

Definition 4 (Key cycle)

1. *The key graph of a message m is a directed graph $G = (V, E)$, in which $V = \{k | k \in \mathbf{keys}(m)\}$ is the set of the vertexes, and $E = \{(k_1 k_2) | k_1 \in V \wedge k_2 \in V \wedge k_1 \sqsubset_m k_2\}$ is the set of the edges.*
2. *We say there exists a key cycle in the message m, if and only if there exists a cycle in the key graph of m.*

From the definitions above, we can see that secret shares are considered in messages. Moreover, the rest of our work does not eliminate key cycles from messages. Both of them make our work different from previous ones.

2.2 Patterns

Owing to the presence of secret shares, the keys related to a message become more complicated. So, before formally defining the pattern, we need to give several functions from **MSG** to **Keys**:

$$\begin{aligned}
\mathbf{sbk}(m) &= \{k | (k \in \mathbf{Keys}) \wedge ((k \preccurlyeq m) \vee \exists j \in [1, n].(k^j \preccurlyeq m))\} \\
\mathbf{rck}(m) &= \{k | (k \in \mathbf{Keys}) \wedge ((k \preccurlyeq m) \vee \forall j \in [1, n].(k^j \preccurlyeq m))\} \\
\mathbf{psk}(m) &= \mathbf{sbk}(m) \setminus \mathbf{rck}(m) \\
\mathbf{eok}(m) &= \mathbf{keys}(m) \setminus \mathbf{sbk}(m).
\end{aligned}$$

Intuitively, "sbk" is an abbreviation for "sub-message keys", "rck" for "recoverable keys", "psk" for "partially shared keys" and "eok" for "encryption-only keys". By the definition above, we have more intuitive properties as follows:

$$(\mathbf{sbk}(m) \cup \mathbf{eok}(m) = \mathbf{keys}(m)) \wedge (\mathbf{sbk}(m) \cap \mathbf{eok}(m) = \emptyset); \quad (1)$$

$$(\mathbf{rck}(m) \cup \mathbf{psk}(m) = \mathbf{sbk}(m)) \wedge (\mathbf{rck}(m) \cap \mathbf{psk}(m) = \emptyset). \quad (2)$$

To define the pattern of a message, we need the functions of $\mathbf{p}$ and an auxiliary function **struct**, which are defined in Fig. 1.

$\mathbf{struct}(d) = \square$;	$\mathbf{p}(d, \mathbf{K}) = d$;
$\mathbf{struct}(k) = \Diamond$;	$\mathbf{p}(k, \mathbf{K}) = k$;
$\mathbf{struct}(k^j) = \Diamond^j$;	$\mathbf{p}(k^j, \mathbf{K}) = k^j$, (for $j \in \{1..n\}$);
$\mathbf{struct}((m_1, m_2)) =$	$\mathbf{p}((m_1, m_2), \mathbf{K}) = (\mathbf{p}(m_1, \mathbf{K}), \mathbf{p}(m_2, \mathbf{K}))$;
$(\mathbf{struct}(m_1), \mathbf{struct}(m_2))$;	$\mathbf{p}(\{\!\|m\|\!\}_k, \mathbf{K}) = \begin{cases} \{\!\|\mathbf{p}(m)\|\!\}_k & (\text{ if } k \in \mathbf{K}); \\ \{\!\|\mathbf{struct}(m)\|\!\}_k & (\text{otherwise.}). \end{cases}$
$\mathbf{struct}(\{\!\|m\|\!\}_k) = \{\!\|\mathbf{struct}(m)\|\!\}$.	

Fig. 1. Rules defining the function $\mathbf{p}$, and auxiliary function **struct**

The function $\mathbf{p}$ and **rck** satisfy the following fundamental properties:

$$\mathbf{p}(m, \mathbf{keys}(m)) = m \tag{3}$$

$$\mathbf{p}(\mathbf{p}(m, \mathbf{K}), \mathbf{K}') = \mathbf{p}(m, \mathbf{K} \cap \mathbf{K}') \tag{4}$$

$$\mathbf{rck}(\mathbf{p}(m, \mathbf{K})) \subseteq \mathbf{rck}(m) \tag{5}$$

These three properties are similar to the properties of $\mathbf{p}$ and $\mathbf{r}$ in [17]. Moreover, about $\mathbf{p}$, we have the following proposition:

Proposition 1. *If* $\mathbf{K}' \cap \mathbf{keys}(m) = \emptyset$, *then* $\mathbf{p}(m, \mathbf{K} \cup \mathbf{K}') = \mathbf{p}(m, \mathbf{K})$.

Intuitively, this proposition means that, given a message m and a key set $\mathbf{K}$, additional keys which are unrelated to m cannot provide additional information about m.

Definition 5 (Function $\mathcal{F}_m$). *Given a message m, a function $\mathcal{F}_m : \wp(\mathbf{Keys}) \to \wp(\mathbf{Keys})$ can be defined*[2]*. Precisely, given a set $\mathbf{K} \subseteq \mathbf{Keys}$, we have*

$$\mathcal{F}_m(\mathbf{K}) = \mathbf{rck}(\mathbf{p}(m, \mathbf{K})) \tag{6}$$

Intuitively, given message m and a key set $\mathbf{K}$, $\mathcal{F}_m(\mathbf{K})$ computes the set of keys which occur as the sub-message of $\mathbf{p}(m, \mathbf{K})$, or whose secret shares fully occur in $\mathbf{p}(m, \mathbf{K})$.

Proposition 2. *The function $\mathcal{F}_m : \wp(\mathbf{Keys}) \to \wp(\mathbf{Keys})$ is monotone.*

The monotonicity of the function $\mathcal{F}_m$ makes it possible to define the greatest fixpoint of $\mathcal{F}_m$: $\mathbf{FIX}(\mathcal{F}_m) = \bigcap_{i=0}^{\ell} \mathcal{F}_m^i(\mathbf{keys}(m))$, where $\ell = |\mathbf{keys}(m)|$. Obviously, by the the monotonicity of $\mathcal{F}_m$, we have $\mathbf{FIX}(\mathcal{F}_m) = \mathcal{F}_m^{\ell}(\mathbf{keys}(m))$.

Definition 6 (Patterns of messages). *The pattern of the message m, written as* $\mathbf{pattern}(m)$, *is define as:*

$$\mathbf{pattern}(m) = \mathbf{p}(m, \mathbf{FIX}(\mathcal{F}_m)) \tag{7}$$

[2] By using $\wp(\mathbf{Keys})$, we mean the power set of **Keys**.

2.3 Equivalence

As usual, the keys in a formal message are considered as bound names(like in spi calculus[5]). Thus, they can be renamed without effecting the essential meaning of the formal message. However, since the secret shares of keys are considered in the formal model, we must redefine the renaming.

Definition 7 (Renaming). *There are three types of renaming: K-renaming (Keys renaming), KS-renaming(Keys and shares renaming) and S-renaming (Shares only renaming). KS-renaming and S-renaming are all defined based on K-renaming.*

1. *Let* $\mathbf{K} \subseteq \mathbf{Keys}$*. A K-renaming on* $\mathbf{K}$ *is a bijection on* $\mathbf{K}$*, often written as* $\sigma[\mathbf{K}]$ *or* $\theta[\mathbf{K}]$*.*

2. *KS-renaming is defined by extending K-renaming. Let* $\mathbf{K}, \mathbf{K}' \subseteq \mathbf{Keys}$*,* $\mathbf{K} \subseteq \mathbf{K}'$*, and* $\sigma[\mathbf{K}']$ *be a K-renaming. A KS-renaming on* $\mathbf{K} \cup \mathbf{s}(\mathbf{K})$*, written as* $\bar{\sigma}[\mathbf{K} \cup \mathbf{s}(\mathbf{K})]$*, is defined as follows:*

$$\begin{aligned} \bar{\sigma}(k) &= \sigma(k) && (k \in \mathbf{K}) \\ \bar{\sigma}(k^j) &= \sigma(k)^j && (k^j \in \mathbf{s}(\mathbf{K})) \end{aligned}$$

3. *S-renaming is also defined based on K-renaming. Let* $\mathbf{K}, \mathbf{K}' \subseteq \mathbf{Keys}$*,* $\mathbf{K} \subseteq \mathbf{K}'$*, and* $\sigma[\mathbf{K}']$ *be a K-renaming. An S-renaming on* $\mathbf{s}(\mathbf{K})$*, written as* $\hat{\sigma}[\mathbf{s}(\mathbf{K})]$*, is defined as follows:*

$$\hat{\sigma}(k^j) = \sigma(k)^j \qquad (k^j \in \mathbf{s}(\mathbf{K}))$$

Recall that the secret shares of a key are different from each other. From definition 7, KS-renaming and S-renaming are also bijections.

As a conventional notation, we have

$$\sigma(\mathbf{K}) \triangleq \{k' | k \in \mathbf{K} \wedge \sigma(k) = k'\}.$$

Similar notations can be used on $\bar{\sigma}$ and $\hat{\sigma}$. When there is no confusion according to the context, we often write $\sigma[\mathbf{K}], \bar{\sigma}[\mathbf{K} \cup \mathbf{s}(\mathbf{K})]$ and $\hat{\sigma}[\mathbf{s}(\mathbf{K})]$ as σ, $\bar{\sigma}$ and $\hat{\sigma}$ respectively for short. Also, we use $m\sigma, m\bar{\sigma}$ and $m\hat{\sigma}$ to denote the applying of $\sigma, \bar{\sigma}$ and $\hat{\sigma}$ respectively to message m.

Here, KS-renaming is consistent renaming[16]. Informally speaking, consistent renaming means that, when k_i occurring in m is renamed to $k_{i'}$, the share of k_i, say k_i^j, is renamed to $k_{i'}^j$ accordingly. Obviously, S-renaming is not a consistent renaming, because the links between a key and its secret shares may be broken after applying S-renaming.

Definition 8 (Equivalence of messages). *Given* $m, m' \in \mathbf{MSG}$*, Message* m' *is said to be equivalent to* m*, written as* $m' \cong m$*, if and only if, there exists a KS-renaming* $\bar{\sigma}$ *based on K-renaming* $\sigma[\mathbf{keys}(\mathbf{pattern}(m))]$*, or, additionally an S-renaming* $\hat{\theta}$ *based on K-renaming* $\theta[\mathbf{psk}(\mathbf{pattern}(m)\bar{\sigma})]$*, such that one of the following holds:*

1. $\mathbf{pattern}(m') = \mathbf{pattern}(m)\bar{\sigma}$
2. $\mathbf{pattern}(m') = (\mathbf{pattern}(m)\bar{\sigma})\hat{\theta}$

This definition of equivalence differs from the equivalence in [16] in that the S-renaming is considered. So, for example, $(\{|k_2|\}_{k_1}, k_1^1)$ and $(\{|k_2|\}_{k_1}, k_3^1)$ are equivalent according to Definition 8, but not equivalent in [16].

3 Computational Model

In computational model, a message is just a bit-string which belongs to $\{0,1\}^*$.

Definition 9 (Computational model). *A computational model is a 4-tuple* $\mathbf{M} = (\mathbf{\Pi}, \mathbf{\Lambda}, \omega, \gamma)$*, in which*

- $\mathbf{\Pi}$ *is an encryption scheme which is composed of a tuple of algorithms* $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$*, namely, the key-generation algorithms* **Gen***, the encryption algorithm* **Enc***, and the decryption algorithm* **Dec***. It is required that* $\mathbf{Dec}_k(\mathbf{Enc}_k(m)) = m$*.*
- $\mathbf{\Lambda}$ *is a secret sharing scheme which is composed of a tuple of algorithms* $(\mathbf{Com}, \mathbf{Crt})$*, namely, the share creation algorithm* **Crt** *and the share combination algorithm* **Com***. It is required that* $\mathbf{Com}(\mathbf{Crt}(k, 1^\eta)) = k$*.*
- $\omega : \mathbf{Data} \to \{0,1\}^*$ *is an interpretation function to evaluate each symbol in* **Data** *to a bit-string.*
- $\gamma : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *is a function to connect two bit-strings to a single bit-string. It can be viewed as the computational counterpart of message concatenation in formal model.*

Some strict definitions about the schemes in model $\mathbf{M}$ can be found in Appendix.

Definition 10 (Computational interpretation of messages). *Given a computational model* $\mathbf{M} = (\mathbf{\Pi}, \mathbf{\Lambda}, \omega, \gamma)$ *and a formal message* m*, we can get the computational interpretation of* m*, that is, associate a collection of distributions (i.e., ensemble) over a bit-string* $[\![m]\!]_{\mathbf{M}} = \{[\![m]\!]_{\mathbf{M}(\eta)}\}_{\eta \in \mathbb{N}}$ *to the formal message* m*. Assume* $\ell = |\mathbf{keys}(m)|$ *and the number of shares for each key is* n*, we can get* $[\![m]\!]_{\mathbf{M}}$ *by the following steps:*

1. *Initialization. Construct an* ℓ *vector* κ *to save the interpretation of keys, and an* $\ell \times n$ *array* ς *to save the interpretation of shares. Then, evaluate* $\kappa[i](1 \le i \le \ell)$ *and* $\varsigma[i,j](1 \le i \le \ell, 1 \le j \le n)$ *by the following procedure:*

$$\textit{for } i = 1 \textit{ to } \ell \textit{ do } \left\{ \begin{array}{l} \kappa[i] \leftarrow \mathbf{Gen}(1^\eta); \\ \{\varsigma[i,1], \varsigma[i,2], \cdots, \varsigma[i,n]\} \leftarrow \mathbf{Crt}(\kappa[i], 1^\eta). \end{array} \right\}$$

2. *Interpretation. Interpretation of the message* m *can be done recursively as follows:*
   - $[\![d]\!]_{\mathbf{M}} = \omega(d)$*, for* $d \in \mathbf{Data}$*.*
   - $[\![k_i]\!]_{\mathbf{M}} = \kappa[i]$*, for* $k_i \in \mathbf{Keys}$ *and* $1 \le i \le \ell$*.*
   - $[\![k_i^j]\!]_{\mathbf{M}} = \varsigma[i,j]$*, for* $k_i^j \in \mathbf{Shares}$ *and* $1 \le j \le n$*.*
   - $[\![(m_1, m_2)]\!]_{\mathbf{M}} = \gamma\left([\![m_1]\!]_{\mathbf{M}}, [\![m_2]\!]_{\mathbf{M}}\right)$*.*
   - $[\![\{\!|m|\!\}_{k_i}]\!]_{\mathbf{M}} = \mathbf{Enc}_{[\![k_i]\!]_{\mathbf{M}}}[\![m]\!]_{\mathbf{M}}$*.*
   - $[\![\mathbf{struct}(m)]\!]_{\mathbf{M}} = 0^{|[\![m]\!]_{\mathbf{M}}|}$*, where* $|[\![m]\!]_{\mathbf{M}}|$ *denotes the length of* $[\![m]\!]_{\mathbf{M}}$*.*

4 Computational Soundness

Intuitively, Computational soundness means that, if two messages are equivalent in the formal model, their interpretation in computational model will be indistinguishable. To prove the computational soundness in this paper, we need some lemmas.

Lemma 1. *Let* $m \in \mathbf{MSG}$, $\bar{\sigma}$ *be a KS-renaming based on K-renaming* $\sigma[keys(m)]$. *Given a computational model* $\mathbf{M}$, *it holds that* $[\![m]\!]_{\mathbf{M}} \approx [\![m\bar{\sigma}]\!]_{\mathbf{M}}$.

Lemma 2. *Let* $m \in \mathbf{MSG}$, $\hat{\theta}$ *be an S-renaming based on K-renaming* $\theta[\mathbf{psk}(m)]$. *Given a computational model* $\mathbf{M} = (\mathbf{\Pi}, \mathbf{\Lambda}, \omega, \gamma)$, *if* $\mathbf{\Pi}$ *is a* **CPA** *secure encryption scheme and* $\mathbf{\Lambda}$ *is a secure secret sharing scheme, then, it holds that* $[\![m]\!]_{\mathbf{M}} \approx [\![m\hat{\theta}]\!]_{\mathbf{M}}$.

Lemma 3. *Let* $m \in \mathbf{MSG}$. *Given a K-renaming* $\theta[\mathbf{psk}(m)]$, *and thus an S-renaming* $\hat{\theta}[\mathbf{s}(\mathbf{psk}(m))]$, *if* $\theta(\mathbf{psk}(m)) \cap \mathbf{keys}(m) = \emptyset$, *then*

$$[\![\mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta}))]\!]_{\mathbf{M}} \approx [\![\mathbf{p}(m, \mathbf{rck}(m))]\!]_{\mathbf{M}}$$

Lemma 4. *Given a formal message* m, *and a computational model* $\mathbf{M} = (\mathbf{\Pi}, \mathbf{\Lambda}, \omega, \gamma)$, *if* $\mathbf{\Pi}$ *is a* **CPA** *secure encryption scheme and* $\mathbf{\Lambda}$ *is a secure secret sharing scheme, it holds that* $[\![m]\!]_{\mathbf{M}} \approx [\![\mathbf{p}(m, \mathbf{rck}(m))]\!]_{\mathbf{M}}$.

Lemma 5. *Given a formal message* m, *and a computational model* $\mathbf{M} = (\mathbf{\Pi}, \mathbf{\Lambda}, \omega, \gamma)$, *if* $\mathbf{\Pi}$ *is a* **CPA** *secure encryption scheme and* $\mathbf{\Lambda}$ *is a secure secret sharing scheme, then it holds that* $[\![m]\!]_{\mathbf{M}} \approx [\![\mathbf{pattern}(m)]\!]_{\mathbf{M}}$.

Lemma 1 holds because the distribution associated with a message is decided only by their meaning, not by the symbols used in the message. Lemma 2 can be proved by reduction based on the security of $\mathbf{\Lambda}$. Lemma 3 can be got by Proposition 1 and Lemma 2. It plays an important role in reconciling the gap between Lemma 2 and Lemma 4. Then, with the help of Lemma 3, Lemma 4 can be proved by reduction based on the **CPA** security of $\mathbf{\Pi}$. Lemma 5 can be easily got by Proposition 2 and Lemma 4 according to hybrid argument. Due to space limitation, we omit the detailed proofs of the lemmas which can be found in the full version of this paper[28].

Theorem 1. *Given two formal messages* m, m', *from which key cycles are not eliminated, and a computational model* $\mathbf{M} = (\mathbf{\Pi}, \mathbf{\Lambda}, \omega, \gamma)$, *in which* $\mathbf{\Pi}$ *is an* **CPA** *secure encryption scheme and* $\mathbf{\Lambda}$ *is a secure secret sharing scheme, if* $m \cong m'$, *then,* $[\![m]\!]_{\mathbf{M}} \approx [\![m']\!]_{\mathbf{M}}$.

Proof. Since $m \cong m'$, from Definition 8, we know that there exists a KS-renaming $\bar{\sigma}$ based on K-renaming $\sigma[\mathbf{keys}(m)]$, or, additionally an S-renaming $\hat{\theta}$ based on K-renaming $\theta[\mathbf{psk}(m\bar{\sigma})]$, such that $\mathbf{pattern}(m) = \mathbf{pattern}(m')\bar{\sigma}$ or $\mathbf{pattern}(m) = (\mathbf{pattern}(m')\bar{\sigma})\hat{\theta}$, both of which imply $[\![\mathbf{pattern}(m)]\!]_{\mathbf{M}} \approx [\![\mathbf{pattern}(m')]\!]_{\mathbf{M}}$ by using Lemma 1 and Lemma 2. Moreover, from Lemma 5, we have $[\![m]\!]_{\mathbf{M}} \approx [\![\mathbf{pattern}(m)]\!]_{\mathbf{M}}$ and $[\![m']\!]_{\mathbf{M}} \approx [\![\mathbf{pattern}(m')]\!]_{\mathbf{M}}$. Therefore, by the transitivity of indistinguishability, we get $[\![m]\!]_{\mathbf{M}} \approx [\![m']\!]_{\mathbf{M}}$.

This completes our proof.

5 Further Discussion

Since the detailed proofs of the lemmas are omitted, we will scratch the main idea in our proofs, and show that the extension in this paper is non-trivial.

Informally speaking, to prove the indistinguishability of two messages in the computational model, we can firstly assume they can be distinguished by a distinguisher, and then construct an adversary based on this distinguisher to break the security of the encryption scheme or secret sharing scheme. In such construction, the adversary is often required to evaluate a message with the help of encryption oracles. When considering key cycles and secret shares, it is infeasible for the adversary to evaluate a message by querying encryption oracle like in [16], because the adversary cannot invent a key and submit it to the oracle for encryption under itself. Still, the approach in [17] can only solve part of the problem. That is, the adversary is given the power to get the cycled keys and then completes the encryption without querying the encryption oracle. Both of them say nothing about how to evaluate secret shares in the presence of key cycle. In our setting, the encryption under a key to itself and the encryption under a key to parts of its secret shares are both defined as key cycles. The former is considered insecure, while the latter is considered secure, which means that the adversary cannot get the encryption key. Then, to evaluate the message in the latter case, the adversary can neither query the encryption oracle, nor complete such encryption by himself. This problem is solved in this paper with the help of S-renaming. That is, the shares of a partially shared key can be renamed to the shares of a newly generated key without changing their meaning in sense of indistinguishability.

6 Conclusion

We show the computational soundness of formal encryption in the presence of both secret shares and key cycles. Our work is an extension to that in [16] and [17], but the result is non-trivial. For example, when both keys and shares occur in a key cycle, we must reconsider what keys can be recovered from it and what cannot. Moreover, by using CPA secure encryption scheme in a computational model, we must deal with the conflict between the definition of CPA and key cycles, especially when secret shares are involved.

For future researches, one can extend this work to the setting of asymmetric cryptography. Another direction is to prove the computational soundness in the presence of active adversaries.

References

1. Abadi, M., Rogaway, P.: Reconciling Two Views of Cryptography (the Computational Soundness of Formal Encryption). In: Watanabe, O., Hagiya, M., Ito, T., van Leeuwen, J., Mosses, P.D. (eds.) TCS 2000. LNCS, vol. 1872, pp. 3–22. Springer, Heidelberg (2000)

2. Dolev, D., Yao, A.C.: On the security of public-key protocols. IEEE Transactions on Information Theory 30(2), 198–208 (1983)
3. Burrows, M., Abadi, M., Needham, R.: A logic of authentication. ACM Transactions on Computer Systems 8(1), 18–36 (1990)
4. Paulson, L.C.: The inductive approach to verifying cryptographic protocols. Journal of Computer Security 6, 85–128 (1998)
5. Abadi, M., Gordon, A.D.: A calculus for cryptographic protocols: The spi calculus. Information and Computation 148(1), 1–70 (1999)
6. Goldwasser, S., Micali, S.: Probabilistic encryption. JCSS 28(2), 270–299 (1984)
7. Yao, A.C.: Theory and application of trapdoor functions. In: Proc. 23rd IEEE Symp. on Foundations of Comp. Science, Chicago, pp. 80–91 (1982)
8. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
9. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42th IEEE Symposium on Foundations of Computers Science, pp. 136–145 (2001)
10. Backes, M., Pfitzmann, B., Waidner, M.: A universally composable cryptographic library. Report 2003/015, Cryptology ePrint Archive (January 2003)
11. Herzog, J.: Computational soundness for standard assumptions of formal cryptography. PhD thesis, Massachusetts Institute of Technology (2004)
12. Micciancio, D., Warinschi, B.: Soundness of Formal Encryption in the Presence of Active Adversaries. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 133–151. Springer, Heidelberg (2004)
13. Adão, P., Bana, G., Herzog, J., Scedrov, A.: Soundness of Formal Encryption in the Presence of Key-cycles. In: di Vimercati, S.D.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 374–396. Springer, Heidelberg (2005)
14. Laud, P.: Symmetric encryption in automatic analyses for confidentiality against active adversaries. In: IEEE Symposium on Security and Privacy, pp. 71–85. IEEE Computer Society (2004)
15. Blanchet, B., Pointcheval, D.: Automated Security Proofs with Sequences of Games. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 537–554. Springer, Heidelberg (2006)
16. Abadi, M., Warinschi, B.: Security analysis of cryptographically controlled access to XML documents. Journal of the ACM 55(2), 6:1–6:29 (2008)
17. Micciancio, D.: Computational Soundness, Co-induction, and Encryption Cycles. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 362–380. Springer, Heidelberg (2010)
18. Shamir, A.: How to share a secret. Communications of the ACM 22, 612–613 (1979)
19. Miklau, G., Suciu, D.: Controlling access to published data using cryptography. In: Freytag, J.C., Lockemann, P.C., Abiteboul, S., Carey, M.J., Selinger, P.G., Heuer, A. (eds.) VLDB 2003: Proceedings of 29th International Conference on Very Large Data Bases, Berlin, Germany, Los Altos, CA 94022, USA, September 9–12, pp. 898–909. Morgan Kaufmann Publishers (2003)
20. Camenisch, J., Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–117. Springer, Heidelberg (2001)
21. Laud, P.: Encryption cycles and two views of cryptography. In: Proceedings of the 7th Nordic Workshop on Secure IT Systems – NORDSEC 2002, Karlstad, Sweden, pp. 85–100 (2002)

22. Adão, P., Bana, G., Herzog, J., Scedrov, A.: Soundness and completeness of formal encryption: The cases of key cycles and partial information leakage. Journal of Computer Security 17(5), 737–797 (2009)
23. Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme Security in the Presence of Key-dependent Messages. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 62–75. Springer, Heidelberg (2003)
24. Hofheinz, D., Unruh, D.: Towards Key-dependent Message Security in the Standard Model. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 108–126. Springer, Heidelberg (2008)
25. Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure Encryption from Decision Diffie-hellman. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 108–125. Springer, Heidelberg (2008)
26. Haitner, I., Holenstein, T.: On the (Im)possibility of Key Dependent Encryption. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 202–219. Springer, Heidelberg (2009)
27. Micciancio, D.: Pseudo-randomness and partial information in symbolic security analysis. Cryptology ePrint Archive, Report 2009/249 (2009), `http://eprint.iacr.org/`
28. Lei, X., Xue, R., Yu, T.: Computational soundness about formal encryption in the presence of secret shares and key cycles. Cryptology ePrint Archive, Report 2010/467 (2010), `http://eprint.iacr.org`

A Some Definitions

We briefly recall some definitions used in this paper. The detailed definitions can be found in many literatures about modern cryptography.

Definition 11 (Indistinguishability). *Let $D = \{D_\eta\}_{\eta\in\mathbb{N}}$ be an ensemble, i.e., a collection of distributions over strings. We say two ensembles D and D' are indistinguishable, written as $D \approx D'$, if for every probabilistic polynomial-time adversary $\mathcal{A}$, there exists a negligible function* **negl**, *such that*

$$\mathbf{Pr}[x \leftarrow D_\eta : \mathcal{A}(1^\eta, x) = 1] - \mathbf{Pr}[x \leftarrow D'_\eta : \mathcal{A}(1^\eta, x) = 1] = \mathbf{negl}(\eta)$$

where $x \leftarrow D_\eta$ means that x is sampled from the distribution D_η.

A typical property of indistinguishability is that it is transitive [21], i.e.,

$$\text{if } D \approx D' \text{ and } D' \approx D'', \text{ then } D \approx D''$$

Definition 12 (Private-key encryption scheme). *A private-key encryption scheme is a tuple of algorithms $\mathbf{\Pi} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ such that:*

1. *The key-generation algorithm* **Gen** *takes as input the security parameter 1^η and outputs a key k. This process can be written as $k \leftarrow \mathbf{Gen}(1^\eta)$.*
2. *The encryption algorithm* **Enc** *takes as input a key k and a message $m \in \{0,1\}^*$, and outputs a ciphertext c. This process can be written as $c \leftarrow \mathbf{Enc}_k(m)$.*
3. *The decryption algorithm* **Dec** *takes as input a key k and a ciphertext c, and outputs a message m. This process is often written as $m := \mathbf{Dec}_k(c)$.*

It is required that $\mathbf{Dec}_k(\mathbf{Enc}_k(m)) = m$.

We will use a standard notion of security for encryption scheme: indistinguishability against chosen plaintext attacks(CPA).

Definition 13 (CPA security). *For any probabilistic polynomial time adversaries* $\mathcal{A}$ *and polynomial* **poly**, *let* $\mathbf{\Pi} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ *be an encryption scheme,* $n = \mathbf{poly}(\eta)$, $k_1, \cdots, k_n$ *be the keys generated by* **Gen**, $O_b(i, m)$ *be an encryption oracle that outputs* $\mathbf{Enc}_{k_i}(m)$ *if* $b = 1$, *or* $\mathbf{Enc}_{k_i}(0^{|m|})$ *if* $b = 0$. *The encryption scheme* $\mathbf{\Pi}$ *is indistinguishable under chosen plaintext attack(or is* **CPA***-secure) if there exists a negligible function* **negl** *such that*

$$\mathbf{Pr}[\mathcal{A}^{O_1}(1^\eta) = 1] - \mathbf{Pr}[\mathcal{A}^{O_0}(1^\eta) = 1] = \mathbf{negl}(\eta)$$

This definition is equivalent to the definition of **IND-CPA** in which only one encryption oracle is given[17].

Definition 14 (Secret sharing scheme). *An* n*-out-of-*n *secret sharing scheme for sharing keys of an encryption scheme* $\mathbf{\Pi}$ *is a tuple of algorithms* $\mathbf{\Lambda} = (\mathbf{Crt}, \mathbf{Com})$ *such that:*

1. *The share creation algorithm* **Crt** *takes as input a key* k *and the security parameter* 1^η *and outputs* n *shares of* $k : k^1, k^2, \cdots, k^n$. *This process can be written as* $\{k^1, k^2, \cdots, k^n\} \leftarrow \mathbf{Crt}(k, 1^\eta)$.
2. *The share combination algorithm* **Com** *takes as input* n *shares* $k^1, k^2, \cdots, k^n$ *and outputs a key* k. *This process can be written as* $k := \mathbf{Com}(k^1, k^2, \cdots, k^n)$.

It is required that $\mathbf{Com}(\mathbf{Crt}(k, 1^\eta)) = k$.

Definition 15 (Security of secret sharing). *For any probabilistic polynomial time adversaries* $\mathcal{A}$ *and polynomial* **poly**, *let* $\mathbf{\Pi} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ *be an encryption scheme,* $\mathbf{\Lambda} = (\mathbf{Crt}, \mathbf{Com})$ *be a secret sharing scheme,* $n = \mathbf{poly}(\eta)$, $\mathbf{sh}(k)$ *be the set of* n *secret shares of key* k *generated by* **Crt**, *and* $\mathbf{sh}(k)|_S$ *be the restriction of* $\mathbf{sh}(k)$ *to the secret shares whose indexes are in* $S \subseteq \{1, \cdots, n\}$. *The secret sharing scheme* $\mathbf{\Lambda}$ *is secure if for any* $S \subset \{1, \cdots, n\}$*(this implies that* $S \neq \{1, \cdots, n\}$*), there exists a negligible function* **negl** *such that*

$$\begin{aligned}&\mathbf{Pr}\left[k_0, k_1 \leftarrow \mathbf{Gen}(1^\eta), \mathbf{sh}(k_0) \leftarrow \mathbf{Crt}(k_0, 1^\eta) : \mathcal{A}(k_0, k_1, \mathbf{sh}(k_0)|_S) = 1\right] - \\ &\mathbf{Pr}\left[k_0, k_1 \leftarrow \mathbf{Gen}(1^\eta), \mathbf{sh}(k_1) \leftarrow \mathbf{Crt}(k_1, 1^\eta) : \mathcal{A}(k_0, k_1, \mathbf{sh}(k_1)|_S) = 1\right] \\ =\ &\mathbf{negl}(\eta)\end{aligned}$$