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, 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 two main approaches to security 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.

[⋆] This work is partially supported by NSFC grants (No. 60873260 and No. 60903210), the 863 Program (No. 2009AA01Z414), the 973 Program (No. 2007CB311202), and the Natural Science Foundation of Jiangsu province of China (No.BK2008090).

Each of the approaches has its advantages and disadvantages. In general, the former is simpler but cannot guarantee computational soundness. The 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 developed a method to bridge the gap between these two approaches, and established computational soundness of 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][9–15] [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 the specific shares can get this 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 care about is whether a formal treatment of secret sharing can keep its computational soundness.

Key cycle. The concept of Key cycles is first hinted in [6], and then be noted since the work by Abadi and Rogway [1]. Non-strictly speaking, key cycle means that a key encrypts itself directly or indirectly. At the first glance, it seems that key cycle does 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[23–25], 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 proved. Finally, we conclude in Section 5 and discuss further work.

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:

Definition 1 (Shares). *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*[3]*,*

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

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.

Definition 2 (Messages). *The set of messages is denoted by* **Msg** *and can be defined in Backus Naur form as follows:*

$$\mathbf{Msg} ::= \mathbf{Data} \mid \mathbf{Keys} \mid \mathbf{Shares} \mid (\mathbf{Msg}, \mathbf{Msg}) \mid \{\!|\mathbf{Msg}|\!\}_{\mathbf{Keys}}$$

Informally, (m_1, m_2) represents the concatenation of m_1 and m_2, and $\{\!|m|\!\}_k$ represents the encryption of m under k.

Obviously, in a message, some parts may occur in the form of cleartext, and the other parts may occur in the form of ciphertext. Without the decryption key, the parts in form of ciphertext show nothing but its structure at most. To reflect this fact, we need to extend the set of messages **Msg** to the set of extended messages **MSG** by introducing some specific symbols.

Definition 3 (Extended messages). *The set of extended messages, written as* **MSG***, is defined under* $\mathbf{Data} \cup \{\square\}, \mathbf{Keys} \cup \{\Diamond\}$*, and* $\mathbf{Shares} \cup \{\Diamond^j\}$ *with a similar syntax to that of* **Msg***:*

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

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

Intuitively, $\Box, \Diamond$ and $\Diamond^j$ denote the unknown data, keys and secret shares respectively.

From definitions of **Msg** and **MSG**, we can see that **Msg** is in fact included in **MSG**, and thus, in most time, when we refer to *message*, we means a member of **MSG**, and use $m, m', m'', \cdots, m_1, m_2, \cdots$ to range over **MSG**.

Similar to [17], we accept the following conventions of notations:

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

Moreover, to simplify our presentation, we will use the symbols of the first order logic in the following definition. For example, we use $\wedge$ for *and*, $\vee$ for *or*, $\neg$ for *negation*, $\exists$ for *exists*, and $\forall$ for *for all.*

Definition 4 (Sub-message). *Let $m, m' \in$ **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 5 (Occurrence). *Let $x \in$ **Keys**$\cup$**Shares** and $m \in$ **MSG**. x occurs in m, written as $x \lessdot m$, if one of the following holds:*

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

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

Definition 6 (Encryption relation). *Let $m \in$ **MSG**, $k_1, k_2 \in$ **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 $(\{\!|m'|\!\}_{k_1} \preccurlyeq m) \wedge (k_2 \in$ **keys**$(m'))$.*

Example 1. Let $m = \{\!|k_1, k_2^1|\!\}_{k_3}$. We have

- $\{\!|k_1, k_2^1|\!\}_{k_3} \preccurlyeq m, (k_1, k_2^1) \preccurlyeq m, k_1 \preccurlyeq m, k_2^1 \preccurlyeq m$;
- $k_1 \lessdot m, k_2^1 \lessdot m, k_3 \lessdot m$;
- $k_3 \sqsubset_m k_1, k_3 \sqsubset_m k_2$.

Definition 7 (Key cycle). [4]

[4] There are many different definitions of key cycles in the literatures, in which [1] is the most general one. The definition here is similar to the definition in [1] except that secret share is considered. Such a general definition is used to emphasize that any form of key cycle is allowed.

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

Since a message may contain some sub-messages in the form of ciphertext, a message will show different views given different keys. When given no further information other than the message itself, the view of the message can be uniquely determined. Informally speaking, this view is just the pattern of the message.

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[5].

- $\mathbf{sbk}(m) : \mathbf{MSG} \to \mathbf{Keys}$, the set of keys which are the sub-messages of m, or whose shares are the sub-messages of m:
$$\mathbf{sbk}(m) = \{k | (k \in \mathbf{Keys}) \wedge ((k \preccurlyeq m) \vee \exists j \in [1, n].(k^j \preccurlyeq m))\}$$
- $\mathbf{rck}(m) : \mathbf{MSG} \to \mathbf{Keys}$, the set of keys which can possibly be recovered from m. Specifically, it returns the keys which are the sub-messages of m, or all of whose shares are sub-messages of m:
$$\mathbf{rck}(m) = \{k | (k \in \mathbf{Keys}) \wedge ((k \preccurlyeq m) \vee \forall j \in [1, n].(k^j \preccurlyeq m))\}$$
- $\mathbf{psk}(m) : \mathbf{MSG} \to \mathbf{Keys}$, the set of keys which do not occur directly as the sub-message of m, but whose secret shares partially occur in m. It can be simply defined by $\mathbf{sbk}(m)$ and $\mathbf{rck}(m)$:
$$\mathbf{psk}(m) = \mathbf{sbk}(m) \setminus \mathbf{rck}(m)$$
- $\mathbf{eok}(m) : \mathbf{MSG} \to \mathbf{Keys}$, the set of keys which only occur in m as the encryption keys:
$$\mathbf{eok}(m) = \mathbf{keys}(m) \setminus \mathbf{sbk}(m)$$

By the definition above, we have more intuitive properties as follows:

$$\mathbf{sbk}(m) \cup \mathbf{eok}(m) = \mathbf{keys}(m); \tag{1}$$
$$\mathbf{sbk}(m) \cap \mathbf{eok}(m) = \emptyset; \tag{2}$$
$$\mathbf{rck}(m) \cup \mathbf{psk}(m) = \mathbf{sbk}(m); \tag{3}$$
$$\mathbf{rck}(m) \cap \mathbf{psk}(m) = \emptyset. \tag{4}$$

[5] Intuitively, "sbk" is an abbreviation for "sub-message keys", "rck" for "recoverable keys", "psk" for "partially shared keys" and "eok" for "encryption-only keys"

Example 2. This example is given to illustrate various functions about keys[6]. Let $m = (\{\!|k_1, k_2^1|\!\}_{k_1}, \{\!|k_3, \{\!|\{\!|k_4|\!\}_{k_5}|\!\}_{k_4}|\!\}_{k_2}, \{\!|k_2^2|\!\}_{k_3}, \{\!|k_4^2|\!\}_{k_6}, k_5^1, k_6^1, \{\!|k_4^1|\!\}_{k_7})$, we have

$$\begin{aligned} \mathbf{keys}(m) &= \{k_1, k_2, k_3, k_4, k_5, k_6, k_7\} & \mathbf{sbk}(m) &= \{k_1, k_2, k_3, k_4, k_5, k_6\} \\ \mathbf{rck}(m) &= \{k_1, k_2, k_3, k_4\} & \mathbf{psk}(m) &= \{k_5, k_6\} \\ \mathbf{eok}(m) &= \{k_7\} \end{aligned}$$

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.

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

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

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

$$\mathbf{p}(m, \mathbf{keys}(m)) = m \tag{5}$$

$$\mathbf{p}(\mathbf{p}(m, \mathbf{K}), \mathbf{K}') = \mathbf{p}(m, \mathbf{K} \cap \mathbf{K}') \tag{6}$$

$$\mathbf{rck}(\mathbf{p}(m, \mathbf{K})) \subseteq \mathbf{rck}(m) \tag{7}$$

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* $\mathbf{p}(m, \mathbf{K} \cup \mathbf{K}') = \mathbf{p}(m, \mathbf{K})$.

Proof. Given $k \in \mathbf{keys}(m)$, since $\mathbf{K}' \cap \mathbf{keys}(m) = \emptyset$, we have $k \notin \mathbf{K}'$. So, if $k \in \mathbf{K} \cup \mathbf{K}'$, then $k \in \mathbf{K}$. On the other hand, if $k \notin \mathbf{K} \cup \mathbf{K}'$, then $k \notin \mathbf{K}$. From the definition of $\mathbf{p}$, what we can get from m with the help of $\mathbf{K}$ is just what we can get from m with the help of $\mathbf{K} \cup \mathbf{K}'$.

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 8 (Function $\mathcal{F}_m$). *Given a message m, a function $\mathcal{F}_m : \wp(\mathbf{Keys}) \to \wp(\mathbf{Keys})$ can be defined[7]. Precisely, given a set $\mathbf{K} \subseteq \mathbf{Keys}$, we have*

$$\mathcal{F}_m(\mathbf{K}) = \mathbf{rck}(\mathbf{p}(m, \mathbf{K})) \tag{8}$$

[6] To keep continuity, the message m used in this example will also be used in the followed examples.

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

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.*

Proof. Assume $K_1 \in \wp(\mathbf{Keys}), \mathbf{K}_2 \in \wp(\mathbf{Keys})$, and $\mathbf{K}_1 \subseteq \mathbf{K}_2$, we will show that $\mathcal{F}_m(\mathbf{K}_1) \subseteq \mathcal{F}_m(\mathbf{K}_2)$.

By equation (8), we have $\mathcal{F}_m(\mathbf{K}_1) = \mathbf{rck}(\mathbf{p}(m, \mathbf{K}_1))$, and $\mathcal{F}_m(\mathbf{K}_2) = \mathbf{rck}(\mathbf{p}(m, \mathbf{K}_2))$. So, to show $\mathcal{F}_m(\mathbf{K}_1) \subseteq \mathcal{F}_m(\mathbf{K}_2)$, we only need to prove that $\mathbf{rck}(\mathbf{p}(m, \mathbf{K}_1)) \subseteq \mathbf{rck}(\mathbf{p}(m, \mathbf{K}_2))$:

$$\begin{aligned}
&\mathbf{rck}(\mathbf{p}(m, \mathbf{K}_1)) && \\
&= \mathbf{rck}(\mathbf{p}(m, \mathbf{K}_2 \cap \mathbf{K}_1)) && \text{by assumption } \mathbf{K}_1 \subseteq \mathbf{K}_2 \\
&= \mathbf{rck}(\mathbf{p}(\mathbf{p}(m, \mathbf{K}_2), \mathbf{K}_1)) && \text{by (6)} \\
&\subseteq \mathbf{rck}(\mathbf{p}(m, \mathbf{K}_2)) && \text{by (7)}
\end{aligned}$$

The monotonicity of the function $\mathcal{F}_m$ makes it possible to define the greatest fix-point of $\mathcal{F}_m$.

Definition 9 (The greatest fix point of $\mathcal{F}_m$). *The greatest fix-point of $\mathcal{F}_m$, written $\mathbf{FIX}(\mathcal{F}_m)$, is defined as follows:*

$$\mathbf{FIX}(\mathcal{F}_m) = \bigcap\nolimits_{i=0}^{\ell} \mathcal{F}_m^i(\mathbf{keys}(m)) \tag{9}$$

where $\ell = |\mathbf{keys}(m)|$.

Obviously, by the definition of greatest fix-point and the monotonicity of $\mathcal{F}_m$, we have

$$\mathbf{FIX}(\mathcal{F}_m) = \mathcal{F}_m^{\ell}(\mathbf{keys}(m)) \tag{10}$$

Definition 10 (Patterns of messages). *The pattern of the message m, written as $\mathbf{pattern}(m)$, is define as(See Example3 in appendix):*

$$\mathbf{pattern}(m) = \mathbf{p}(m, \mathbf{FIX}(\mathcal{F}_m)) \tag{11}$$

Example 3. Let m be the same as in Example 2, and assume the number of a key's shares to be 2. Starting from the set $\mathbf{K}_0 = \mathbf{keys}(m)$, the greatest fix point of $\mathcal{F}_m$ can be computed recursively as follows:

$$\begin{aligned}
\mathbf{K}_0 &= \{k_1, k_2, k_3, k_4, k_5, k_6, k_7\} \\
\mathbf{p}(m, \mathbf{K}_0) &= (\{\!|k_1, k_2^1|\!\}_{k_1}, \{\!|k_3, \{\!|\{\!|k_4|\!\}_{k_5}|\!\}_{k_4}|\!\}_{k_2}, \{\!|k_2^2|\!\}_{k_3}, \{\!|k_4^2|\!\}_{k_6}, k_5^1, k_6^1, \{\!|k_4^1|\!\}_{k_7}) \\
\mathbf{K}_1 &= \mathcal{F}_m(\mathbf{K}_0) = \mathbf{rck}(\mathbf{p}(m, \mathbf{K}_0)) = \{k_1, k_2, k_3, k_4\} \\
\mathbf{p}(m, \mathbf{K}_1) &= (\{\!|k_1, k_2^1|\!\}_{k_1}, \{\!|k_3, \{\!|\{\!|\Diamond|\!\}_{k_5}|\!\}_{k_4}|\!\}_{k_2}, \{\!|k_2^2|\!\}_{k_3}, \{\!|\Diamond^2|\!\}_{k_6}, k_5^1, k_6^1, \{\!|\Diamond^1|\!\}_{k_7}) \\
\mathbf{K}_2 &= \mathcal{F}_m(\mathbf{K}_1) = \mathbf{rck}(\mathbf{p}(m, \mathbf{K}_1)) = \{k_1, k_2, k_3\} \\
\mathbf{p}(m, \mathbf{K}_2) &= (\{\!|k_1, k_2^1|\!\}_{k_1}, \{\!|k_3, \{\!|\{\!|\Diamond|\!\}|\!\}_{k_4}|\!\}_{k_2}, \{\!|k_2^2|\!\}_{k_3}, \{\!|\Diamond^2|\!\}_{k_6}, k_5^1, k_6^1, \{\!|\Diamond^1|\!\}_{k_7})
\end{aligned}$$

$$\mathbf{K}_3 = \mathcal{F}_m(\mathbf{K}_2) = \mathbf{rck}(\mathbf{p}(m, \mathbf{K}_2)) = \{k_1, k_2, k_3\}$$

Then, we have $\mathbf{FIX}(\mathcal{F}_m) = \{k_1, k_2, k_3\}$, and thus,

$$\begin{aligned}\mathbf{pattern}(m) &= \mathbf{p}(m, \mathbf{FIX}(\mathcal{F}_m)) \\ &= (\{\!|k_1, k_2^1|\!\}_{k_1}, \{\!|k_3, \{\!|\{\!|\Diamond|\!\}|\!\}_{k_4}|\!\}_{k_2}, \{\!|k_2^2|\!\}_{k_3}, \{\!|\Diamond^2|\!\}_{k_6}, k_5^1, k_6^1, \{\!|\Diamond^1|\!\}_{k_7})\end{aligned}$$

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 11 (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:*

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

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 11, KS-renaming and S-renaming are also bijections.

Here, KS-renaming is similar to the consistent renaming defined in [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. For example, in $m\bar{\sigma}$, $\bar{\sigma}(k^j)$ is a share of $\bar{\sigma}(k)$ if k^j is a share of k in m, while in $m\hat{\sigma}$, such relation may be broken.

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.

Let $m \in \mathbf{MSG}$, $\bar{\sigma}[\mathbf{K} \cup \mathbf{s}(\mathbf{K})]$ be a KS-renaming. We use $m\bar{\sigma}$ as applying $\bar{\sigma}$ to message m. That is, rename all the key $k_i \in \mathbf{K}$ and its secret shares k_i^j occurring in m with $\bar{\sigma}(k_i)$ and $\bar{\sigma}(k_i^j)$ respectively.

Similarly, let $m \in \mathbf{MSG}$, $\hat{\sigma}[\mathbf{s}(\mathbf{K})]$ be an S-renaming on $\mathbf{s}(\mathbf{K})$. We use $m\hat{\sigma}$ as applying $\hat{\sigma}$ to message m. That is, rename all secret shares $k^j \in \mathbf{s}(\mathbf{K})$ with $\hat{\sigma}(k^j)$ without renaming of k itself.

Now, it suffices to define the equivalence of messages.

Definition 12 (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 12, but not equivalent in [16]. A more complicated example can be found in Example4.

Example 4. This example is used to illustrate the three types of renaming and messages equivalence. Continuing Example 3, let $\mathbf{K} = \mathbf{keys}(\mathbf{pattern}(m))$, $\mathbf{K}' = \mathbf{psk}(\mathbf{pattern}(m)\bar{\sigma})$. We then define a KS-renaming $\bar{\sigma}$ based on a K-renaming $\sigma[\mathbf{K}]$, and an S-renaming $\hat{\theta}$ based on a K-renaming $\theta[\mathbf{K}']$(Refer to Fig. 2).

From Definition 12 and Fig. 2, if one of the following two conditions holds,

$$\mathbf{pattern}(m') = (\{\!|k_7, k_6^1|\!\}_{k_7}, \{\!|k_5, \{\!|\{\!|\Diamond|\!\}|\!\}_{k_4}|\!\}_{k_6}, \{\!|k_6^2|\!\}_{k_5}, \{\!|\Diamond^2|\!\}_{k_2}, k_3^1, k_2^1, \{\!|\Diamond^1|\!\}_{k_1})$$
$$\mathbf{pattern}(m') = (\{\!|k_7, k_6^1|\!\}_{k_7}, \{\!|k_5, \{\!|\{\!|\Diamond|\!\}|\!\}_{k_4}|\!\}_{k_6}, \{\!|k_6^2|\!\}_{k_5}, \{\!|\Diamond^2|\!\}_{k_2}, k_{3'}^1, k_{2'}^1, \{\!|\Diamond^1|\!\}_{k_1})$$

we have $m' \cong m$.

3 Computational model

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

Definition 13 (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.

$\mathbf{pattern}(m)$	$(\{\!\vert k_1, k_2^1\vert\!\}_{k_1}, \{\!\vert k_3, \{\!\vert\{\!\vert\Diamond\vert\!\}\vert\!\}_{k_4}\vert\!\}_{k_2}, \{\!\vert k_2^2\vert\!\}_{k_3}, \{\!\vert\Diamond^2\vert\!\}_{k_6}, k_5^1, k_6^1, \{\!\vert\Diamond^1\vert\!\}_{k_7})$
$\begin{array}{c}\mathbf{K}\\ \downarrow\\ \sigma(\mathbf{K})\end{array}$	$\begin{array}{ccccccc} k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & k_7\\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow\\ k_7 & k_6 & k_5 & k_4 & k_3 & k_2 & k_1\end{array}$
$\begin{array}{c}\mathbf{K}\cup\mathbf{s}(\mathbf{K})\\ \downarrow\\ \bar{\sigma}(\mathbf{K}\cup\mathbf{s}(\mathbf{K}))\end{array}$	$\begin{array}{cccccccccccccccc} k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & k_7 & k_1^1 & k_1^2 & \cdots & k_5^1 & k_5^2 & k_6^1 & k_6^2 & k_7^1 & k_7^2\\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \cdots & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow\\ k_7 & k_6 & k_5 & k_4 & k_3 & k_2 & k_1 & k_7^1 & k_7^2 & \cdots & k_3^1 & k_3^2 & k_2^1 & k_2^2 & k_1^1 & k_1^2\end{array}$
$\mathbf{pattern}(m)\bar{\sigma}$	$(\{\!\vert k_7, k_6^1\vert\!\}_{k_7}, \{\!\vert k_5, \{\!\vert\{\!\vert\Diamond\vert\!\}\vert\!\}_{k_4}\vert\!\}_{k_6}, \{\!\vert k_6^2\vert\!\}_{k_5}, \{\!\vert\Diamond^2\vert\!\}_{k_2}, k_3^1, k_2^1, \{\!\vert\Diamond^1\vert\!\}_{k_1})$
$\begin{array}{c}\mathbf{K}'\\ \downarrow\\ \theta(\mathbf{K}')\end{array}$	$\begin{array}{cc} k_3 & k_2\\ \downarrow & \downarrow\\ k_{3'} & k_{2'}\end{array}$
$\begin{array}{c}\mathbf{s}(\mathbf{K}')\\ \downarrow\\ \hat{\theta}(\mathbf{s}(\mathbf{K}'))\end{array}$	$\begin{array}{cccc} k_3^1 & k_3^2 & k_2^1 & k_2^2\\ \downarrow & \downarrow & \downarrow & \downarrow\\ k_{3'}^1 & k_{3'}^2 & k_{2'}^1 & k_{2'}^2\end{array}$
$(\mathbf{pattern}(m)\bar{\sigma})\hat{\theta}$	$(\{\!\vert k_7, k_6^1\vert\!\}_{k_7}, \{\!\vert k_5, \{\!\vert\{\!\vert\Diamond\vert\!\}\vert\!\}_{k_4}\vert\!\}_{k_6}, \{\!\vert k_6^2\vert\!\}_{k_5}, \{\!\vert\Diamond^2\vert\!\}_{k_2}, k_{3'}^1, k_{2'}^1, \{\!\vert\Diamond^1\vert\!\}_{k_1})$

Fig. 2. An example for KS-renaming and an S-renaming

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

$$\text{if } D \approx D' \text{ and } D' \approx D'', \text{ then } D \approx D'' \tag{12}$$

Definition 14 (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* $\mathbf{Gen}$ *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* $\mathbf{Enc}$ *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* $\mathbf{Dec}$ *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: indistinguishability against chosen plaintext attacks(CPA).

Definition 15 (CPA security). *For any probabilistic polynomial time adversaries* $\mathcal{A}$ *and polynomial* $\mathbf{poly}$*, 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* $\mathbf{Gen}$*,* $O_b(i, m)$ *be an encryption oracle that outputs* $\mathbf{Enc}_{k_i}(m)$ *if* $b = 1$*, or* $\mathbf{Enc}_{k_i}(0^{\vert m\vert})$ *if* $b = 0$*. The encryption scheme* $\mathbf{\Pi}$ *is indistinguishable under chosen plaintext attack(or is* $\mathbf{CPA}$*-secure) if there exists a negligible function* $\mathbf{negl}$ *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 16 (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 17 (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*

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

Definition 18 (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.*
- $\mathbf{\Lambda}$ *is a secret sharing scheme.*
- $\omega : \mathbf{Data} \to \{0,1\}^*$ *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.*

Definition 19 (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* $\kappa[i](1 \leq i \leq \ell)$ *and* $\varsigma[i, j](1 \leq i \leq \ell, 1 \leq j \leq n)$ *by the following procedure:*

```
        for $i$ = 1 to $\ell$ 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* $1 \leq i \leq \ell$.
   - $[\![k_i^j]\!]_{\mathbf{M}} = \varsigma[i,j]$*, for* $k_i^j \in \mathbf{Shares}$ *and* $1 \leq j \leq n$.
   - $[\![(m_1, m_2)]\!]_{\mathbf{M}} = \gamma\left([\![m_1]\!]_{\mathbf{M}}, [\![m_2]\!]_{\mathbf{M}}\right)$.
   - $[\![\{\!|m|\!\}_{k_i}]\!]_{\mathbf{M}} = \mathbf{Enc}_{[\![k_i]\!]_{\mathbf{M}}}[\![m]\!]_{\mathbf{M}}$.
   - $[\![\mathbf{struct}(m)]\!]_{\mathbf{M}} = 0^{|[\![m]\!]_{\mathbf{M}}|}$*, where* $|[\![m]\!]_{\mathbf{M}}|$ *denotes the length of* $[\![m]\!]_{\mathbf{M}}$[8].

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 clarify the proof, we use Fig.3 to list the invoking structure of these lemmas and the propositions in proving the computational soundness theorem, where $a \rightarrow b$ means that a is invoked in proving b.

$$\begin{tikzcd}
 & & \text{Proposition 1} \arrow[d] & & \text{Proposition 2} \arrow[d] \\
\text{Lemma 1} \arrow[drr] & \text{Lemma 2} \arrow[r] \arrow[dr] & \text{Lemma 3} \arrow[r] & \text{Lemma 4} \arrow[r] & \text{Lemma 5} \arrow[dll] \\
 & & \text{Computational soundness theorem} & &
\end{tikzcd}$$

Fig. 3. The invoking structure in proving the computational soundness theorem

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*

$$[\![m]\!]_{\mathbf{M}} \approx [\![m\bar{\sigma}]\!]_{\mathbf{M}}$$

Proof. According to the definition of KS-renaming in Definition 11, $m\bar{\sigma}$ is got from m by consistently renaming its keys and key shares according to $\bar{\sigma}$. But the distribution associated with a message is decided only by their meaning, not by the symbols used in the message. Thus, this lemma holds.

[8] Here, as mentioned in [17], we assume that all functions operating on messages are length-regular. So, $|[\![m]\!]_{\mathbf{M}}|$ depends only on $[\![\mathbf{struct}(m)]\!]_{\mathbf{M}}$, which makes this definition well-defined.

In fact, Lemma 1 is the same as Lemma 8 in [16]. Here, KS-renaming is the consistent renaming in [16].

The following lemma is similar to Lemma 1 except that S-renaming is used. However, Lemma 1 cannot be naturally applied on S-renaming, simply because S-renaming is actually not a consistent renaming.

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*

$$[\![m]\!]_{\mathbf{M}} \approx [\![m\hat{\theta}]\!]_{\mathbf{M}}$$

Proof. Assume $|\mathbf{psk}(m)| = \rho$ is polynomially bounded in the length of message m, and thus $\mathbf{psk}(m) = \{k_{a_1}, k_{a_2}, \cdots, k_{a_\rho}\}$. Let $m_0 = m$, and $m_i = m_{i-1}\hat{\theta}[\{k_{a_i}\}]$ where $1 \leq i \leq \rho$. We have $m_\rho = m\hat{\theta}[\mathbf{psk}(m)]$, i.e.,$m_\rho = m\hat{\theta}$. By the hybrid argument, to show $[\![m]\!]_{\mathbf{M}} \approx [\![m\hat{\theta}]\!]_{\mathbf{M}}$, we only need to show $[\![m_{i-1}]\!]_{\mathbf{M}} \approx [\![m_i]\!]_{\mathbf{M}}$, where $1 \leq i \leq \rho$.

Assume $\theta(k_{a_i}) = k_{a'_i}$. Let us evaluate messages m_{i-1} and m_i according to Definition 19. Intuitively, the only difference between m_{i-1} and m_i is that, in m_i, the secret shares of k_{a_i} is replaced by the secret shares of $k_{a'_i}$. So, we can use Definition 19 to get computational interpretations of each symbol in m_{i-1}, and complete evaluating message m_{i-1}. To evaluate message m_i, we use the same computational interpretation to m_{i-1} except the secret shares of $k_{a'_i}$. To give the computational interpretation of shares of $k_{a'_i}$, we firstly generate a new key by **Gen** of $\mathbf{\Pi}$; then create n secret shares of this key by **Crt** of $\mathbf{\Lambda}$, and save them in $\varsigma[i,1]$ to $\varsigma[i,n]$ respectively. By doing such, we get $[\![m_{i-1}]\!]_{\mathbf{M}}$ and $[\![m_i]\!]_{\mathbf{M}}$.

Let $\mathcal{D}_1$ be a probabilistic polynomial-time distinguisher, and set

$$\begin{aligned}\varepsilon_1(\eta) \triangleq \mathbf{Pr}\left[v_1 \leftarrow [\![m_{i-1}]\!]_{\mathbf{M}(\eta)} : \mathcal{D}_1(v_1, 1^\eta) = 1\right] - \\ \mathbf{Pr}\left[v_1 \leftarrow [\![m_i]\!]_{\mathbf{M}(\eta)} : \mathcal{D}_1(v_1, 1^\eta) = 1\right].\end{aligned}$$

Now, assume for contradiction that $\mathcal{D}_1$ distinguishes $[\![m_{i-1}]\!]_{\mathbf{M}}$ from $[\![m_i]\!]_{\mathbf{M}}$ with non-negligible probability, i.e., $\varepsilon_1(\eta)$ is non-negligible. Then we construct an adversary $\mathcal{A}_1$ to break the security of sharing scheme $\mathbf{\Lambda}$ with the help of distinguisher $\mathcal{D}_1$.

Let n be the number of shares created by $\mathbf{\Lambda}$, $S_i = \{j | k^j_{a_i} \lessdot m\}$ be the set of indexes j such that $k^j_{a_i}$ occurs in m. Since $k_{a_i} \in \mathbf{psk}(m)$, the shares of k_{a_i} only partially occur in m, that is $|S_i| < n$. From the definition of m_{i-1}, we know that the shares of k_{a_i} occurring in m are exactly the shares of k_{a_i} occurring in m_{i-1}. So, the share numbers of k_{a_i} occurring in m_{i-1} is also $|S_i|$.

Adversary 1 ($\mathcal{A}_1$) *The adversary is given two keys $\hat{k}_0, \hat{k}_1 \leftarrow$* **Gen**$(1^\eta)$ *and a set of shares*[9] $\{\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_p\}$ *either sampled from* **Crt**$(\hat{k}_0, 1^\eta)|_{S_i}$*, or sampled from* **Crt**$(\hat{k}_1, 1^\eta)|_{S_i}$*, where $p = |S_i| < n$.*

[9] Here, we use $\hat{k}$ or $\hat{s}$ instead of k or s to distinguish the bit-string keys or shares from the formal symbols of keys or shares.

1. *$\mathcal{A}_1$ evaluates m_{i-1} to get a value v_1:*
   (a) *Let $|\mathbf{keys}(m_{i-1})| = \ell$. Construct an ℓ vector κ and an $(\ell \times n)$ array ς;*
   (b) *$\kappa[j](j \neq i)$ is initialized by sampling from $\mathbf{Gen}(1^\eta)$;*
   (c) *$\varsigma[j,1], \varsigma[j,2], \cdots, \varsigma[j,n](j \neq i)$ are initialized by sampling from $\mathbf{Crt}(\kappa[j], 1^\eta)$;*
   (d) *m_{i-1} is evaluated to value v_1 according to Definition 19 except k_{a_i} and the shares of k_{a_i}. More precisely, k_{a_i} is interpreted by $\hat{k}_0$, and the shares of k_{a_i} is interpreted by $\{\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_p\}$.*
2. *$\mathcal{A}_1$ runs $\mathcal{D}_1(v_1, 1^\eta)$, and outputs whatever $\mathcal{D}_1(v_1, 1^\eta)$ outputs.*

Note that both $\hat{k}_0$ and $\hat{k}_1$ are generated by $\mathbf{Gen}$. So, if $\{\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_p\}$ are sampled from $\mathbf{Crt}(\hat{k}_0, 1^\eta)|_{S_i}$, then v_1 is just sampled from $[\![m_{i-1}]\!]_{\mathbf{M}}$. If $\{\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_p\}$ are sampled from $\mathbf{Crt}(\hat{k}_1, 1^\eta)|_{S_i}$, then k_{a_i} is interpreted by $\hat{k}_0$, while the shares of k_{a_i} are interpreted by the shares of $\hat{k}_1$. By the definition of m_i, in this situation, v_1 is just sampled from $[\![m_i]\!]_{\mathbf{M}}$. Considering that $\mathcal{A}_1$ outputs whatever $\mathcal{D}_1(v_1, 1^\eta)$ outputs, we have

$$\begin{aligned}
&\mathbf{Pr}\left[\hat{k}_0, \hat{k}_1 \leftarrow \mathbf{Gen}(1^\eta), \mathbf{sh}(\hat{k}_0) \leftarrow \mathbf{Crt}(\hat{k}_0, 1^\eta) : \mathcal{A}_1(\hat{k}_0, \hat{k}_1, \mathbf{sh}(\hat{k}_0)|_{S_i}) = 1\right] - \\
&\mathbf{Pr}\left[\hat{k}_0, \hat{k}_1 \leftarrow \mathbf{Gen}(1^\eta), \mathbf{sh}(\hat{k}_1) \leftarrow \mathbf{Crt}(\hat{k}_1, 1^\eta) : \mathcal{A}_1(\hat{k}_0, \hat{k}_1, \mathbf{sh}(\hat{k}_1)|_{S_i}) = 1\right] \\
=& \mathbf{Pr}\left[v_1 \leftarrow [\![m_{i-1}]\!]_{\mathbf{M}(\eta)} : \mathcal{D}_1(v_1, 1^\eta) = 1\right] - \mathbf{Pr}\left[v_1 \leftarrow [\![m_i]\!]_{\mathbf{M}(\eta)} : \mathcal{D}_1(v_1, 1^\eta) = 1\right] \\
=& \varepsilon_1(\eta)
\end{aligned}$$

This shows that $\mathcal{A}_1$ can break $\mathbf{\Lambda}$ with non-negligible probability, which is in contradiction with the security of $\mathbf{\Lambda}$, and thus Lemma 2 holds.

Example 5. Recall message m in Example 2, we have $\mathbf{psk}(m) = \{k_5, k_6\}$, Fig. 4 shows an S-renaming and the messages m_0, m_1, and m_2 constructed according to the approach in proof of Lemma 2. Given a computational model $\mathbf{M}$, from Lemma 2, we know that $[\![m]\!]_{\mathbf{M}} \approx [\![m\hat{\theta}]\!]_{\mathbf{M}}$.

$\mathbf{s}(\mathbf{psk}(m))$ $\downarrow$ $\hat{\theta}(\mathbf{s}(\mathbf{psk}(m)))$	$k_5^1\ k_5^2\ k_6^1\ k_6^2$ $\downarrow\ \downarrow\ \downarrow\ \downarrow$ $k_{5'}^1\ k_{5'}^2\ k_{6'}^1\ k_{6'}^2$
$m_0 = m$	$(\{\!\vert k_1, k_2^1\vert\!\}_{k_1}, \{\!\vert k_3, \{\!\vert\{\!\vert k_4\vert\!\}_{k_5}\vert\!\}_{k_4}\vert\!\}_{k_2}, \{\!\vert k_2^2\vert\!\}_{k_3}, \{\!\vert k_4^2\vert\!\}_{k_6}, k_5^1, k_6^1, \{\!\vert k_4^1\vert\!\}_{k_7})$
$m_1 = m_0\hat{\theta}[\mathbf{s}(k_5)]$	$(\{\!\vert k_1, k_2^1\vert\!\}_{k_1}, \{\!\vert k_3, \{\!\vert\{\!\vert k_4\vert\!\}_{k_5}\vert\!\}_{k_4}\vert\!\}_{k_2}, \{\!\vert k_2^2\vert\!\}_{k_3}, \{\!\vert k_4^2\vert\!\}_{k_6}, k_{5'}^1, k_6^1, \{\!\vert k_4^1\vert\!\}_{k_7})$
$m_2 = m_1\hat{\theta}[\mathbf{s}(k_6)] = m\hat{\theta}$	$(\{\!\vert k_1, k_2^1\vert\!\}_{k_1}, \{\!\vert k_3, \{\!\vert\{\!\vert k_4\vert\!\}_{k_5}\vert\!\}_{k_4}\vert\!\}_{k_2}, \{\!\vert k_2^2\vert\!\}_{k_3}, \{\!\vert k_4^2\vert\!\}_{k_6}, k_{5'}^1, k_{6'}^1, \{\!\vert k_4^1\vert\!\}_{k_7})$

Fig. 4. An example for applying S-renaming in proof of Lemma 2.

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*

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

Proof. If we can show

$$\mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta})) = \mathbf{p}(m, \mathbf{rck}(m))\hat{\theta} \tag{13}$$

then, by Lemma 2, we can directly show that Lemma 3 holds. We then show (13) by the following two steps:

$$\mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta})) = \mathbf{p}(m\hat{\theta}, \mathbf{rck}(m)) \tag{14}$$

$$\mathbf{p}(m\hat{\theta}, \mathbf{rck}(m)) = \mathbf{p}(m, \mathbf{rck}(m))\hat{\theta} \tag{15}$$

Proof of (14). From the definition of $\mathbf{sbk}$ and S-renaming, we have $\mathbf{sbk}(m\hat{\theta}) = \mathbf{rck}(m) \cup \theta(\mathbf{psk}(m))$. Considering that $\theta(\mathbf{psk}(m)) \cap \mathbf{keys}(m) = \emptyset$, and $\mathbf{rck}(m) \subseteq \mathbf{keys}(m)$ by (3) and (1), we have $\theta(\mathbf{psk}(m)) \cap \mathbf{rck}(m) = \emptyset$. Together with Proposition 1 showed in section 2.2, we get that

$$\begin{aligned}\mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta})) &= \mathbf{p}(m\hat{\theta}, \mathbf{rck}(m) \cup \theta(\mathbf{psk}(m))) \\ &= \mathbf{p}(m\hat{\theta}, \mathbf{rck}(m))\end{aligned}$$

Proof of (15). From (4), we know that $\mathbf{psk}(m) \cap \mathbf{rck}(m) = \emptyset$. So, $\mathbf{p}(m\hat{\theta}, \mathbf{rck}(m))$ is only different from $\mathbf{p}(m, \mathbf{rck}(m))$ in that the shares of keys in $\mathbf{psk}(m)$ is renamed according to $\hat{\theta}$. Therefore, by using the same $\hat{\theta}$ on $\mathbf{p}(m, \mathbf{rck}(m))$, we can get $\mathbf{p}(m\hat{\theta}, \mathbf{rck}(m))$.

Example 6. Continue the Example 5, we have

$$\begin{aligned}
m &= \left(\{\!|k_1, k_2^1|\!\}_{k_1}, \{\!|k_3, \{\!|\{\!|k_4|\!\}_{k_5}|\!\}_{k_4}|\!\}_{k_2}, \{\!|k_2^2|\!\}_{k_3}, \{\!|k_4^2|\!\}_{k_6}, k_5^1, k_6^1, \{\!|k_4^1|\!\}_{k_7}\right) \\
\mathbf{rck}(m) &= \{k_1, k_2, k_3, k_4\} \\
\mathbf{p}(m, \mathbf{rck}(m)) &= (\{\!|k_1, k_2^1|\!\}_{k_1}, \{\!|k_3, \{\!|\{\!|\Diamond|\!\}_{k_5}|\!\}_{k_4}|\!\}_{k_2}, \{\!|k_2^2|\!\}_{k_3}, \{\!|\Diamond^2|\!\}_{k_6}, k_5^1, k_6^1, \{\!|\Diamond^1|\!\}_{k_7}) \\
\mathbf{p}(m, \mathbf{rck}(m))\hat{\theta} &= \left(\{\!|k_1, k_2^1|\!\}_{k_1}, \{\!|k_3, \{\!|\{\!|\Diamond|\!\}_{k_5}|\!\}_{k_4}|\!\}_{k_2}, \{\!|k_2^2|\!\}_{k_3}, \{\!|\Diamond^2|\!\}_{k_6}, k_{5'}^1, k_{6'}^1, \{\!|\Diamond^1|\!\}_{k_7}\right) \\
m\hat{\theta} &= \left(\{\!|k_1, k_2^1|\!\}_{k_1}, \{\!|k_3, \{\!|\{\!|k_4|\!\}_{k_5}|\!\}_{k_4}|\!\}_{k_2}, \{\!|k_2^2|\!\}_{k_3}, \{\!|k_4^2|\!\}_{k_6}, k_{5'}^1, k_{6'}^1, \{\!|k_4^1|\!\}_{k_7}\right) \\
\mathbf{sbk}(m\hat{\theta}) &= \{k_1, k_2, k_3, k_4, k_{5'}, k_{6'}\} \\
\mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta})) &= \left(\{\!|k_1, k_2^1|\!\}_{k_1}, \{\!|k_3, \{\!|\{\!|\Diamond|\!\}_{k_5}|\!\}_{k_4}|\!\}_{k_2}, \{\!|k_2^2|\!\}_{k_3}, \{\!|\Diamond^2|\!\}_{k_6}, k_{5'}^1, k_{6'}^1, \{\!|\Diamond^1|\!\}_{k_7}\right) \\
\mathbf{p}(m\hat{\theta}, \mathbf{rck}(m)) &= \left(\{\!|k_1, k_2^1|\!\}_{k_1}, \{\!|k_3, \{\!|\{\!|\Diamond|\!\}_{k_5}|\!\}_{k_4}|\!\}_{k_2}, \{\!|k_2^2|\!\}_{k_3}, \{\!|\Diamond^2|\!\}_{k_6}, k_{5'}^1, k_{6'}^1, \{\!|\Diamond^1|\!\}_{k_7}\right)
\end{aligned}$$

Obviously, $\mathbf{p}(m, \mathbf{rck}(m))\hat{\theta} = \mathbf{p}(m\hat{\theta}, \mathbf{rck}(m))$. Then, from Lemma 2, we get

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

as expected.

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 $[\![m]\!]_{\mathbf{M}} \approx [\![\mathbf{p}(m, \mathbf{rck}(m))]\!]_{\mathbf{M}}$.*

Proof. Given a message m, we can easily construct an S-renaming $\hat{\theta}$ based on K-renaming $\theta[\mathbf{psk}(m)]$ such that $\theta(\mathbf{psk}(m)) \cap \mathbf{keys}(m) = \emptyset$. Then, we have

$$\llbracket m \rrbracket_{\mathbf{M}} \approx \llbracket m\hat{\theta} \rrbracket_{\mathbf{M}} \qquad \text{by Lemma 2}$$
$$\llbracket \mathbf{p}(m, \mathbf{rck}(m)) \rrbracket_{\mathbf{M}} \approx \llbracket \mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta})) \rrbracket_{\mathbf{M}} \qquad \text{by Lemma 3}$$

Therefore, to prove $\llbracket m \rrbracket_{\mathbf{M}} \approx \llbracket \mathbf{p}(m, \mathbf{rck}(m)) \rrbracket_{\mathbf{M}}$, we only need to show

$$\llbracket m\hat{\theta} \rrbracket_{\mathbf{M}} \approx \llbracket \mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta})) \rrbracket_{\mathbf{M}}$$

Let us evaluate message $m\hat{\theta}$ and $\mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta}))$ according to Definition 19.

Intuitively, the difference between $m\hat{\theta}$ and m is that, in $m\hat{\theta}$, the secret shares of k in $\mathbf{psk}(m)$ are replaced by the secret shares of a new key. Thus, we can evaluate $m\hat{\theta}$ by generating $|\mathbf{psk}(m)|$ more keys and their secret shares.

The difference between $\mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta}))$ and $m\hat{\theta}$ is that, all the sub-messages of $m\hat{\theta}$ in form of $\{\!|m'|\!\}_{k_i}$, where $k_i \in \mathbf{eok}(m\hat{\theta}) = \mathbf{keys}(m\hat{\theta}) \setminus \mathbf{sbk}(m\hat{\theta})$, are replaced by $\{\!|\mathbf{struct}(m')|\!\}_{k_i}$. So, according to Definition 19, we can evaluate $\mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta}))$ by using the same computational interpretation of $m\hat{\theta}$ except the sub-message in form of $\{\!|m'|\!\}_{k_i}$ where $k_i \in \mathbf{eok}(m\hat{\theta})$. The computational interpretation of $\{\!|m'|\!\}_{k_i}$ is simply interpreted by $0^{|\llbracket m' \rrbracket_{\mathbf{M}}|}$.

By doing such, we get $\llbracket m\hat{\theta} \rrbracket_{\mathbf{M}}$ and $\llbracket \mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta})) \rrbracket_{\mathbf{M}}$.

Let $\mathcal{D}_2$ be a probabilistic polynomial-time distinguisher, and set

$$\begin{aligned}\varepsilon_2(\eta) \triangleq & \mathbf{Pr}\left[v_2 \leftarrow \llbracket m\hat{\theta} \rrbracket_{\mathbf{M}(\eta)} : \mathcal{D}_2(v_2, 1^\eta) = 1\right] - \\ & \mathbf{Pr}\left[v_2 \leftarrow \llbracket \mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta})) \rrbracket_{\mathbf{M}(\eta)} : \mathcal{D}_2(v_2, 1^\eta) = 1\right].\end{aligned}$$

Assume for contradiction that there is a distinguisher $\mathcal{D}_2$ which can distinguish $\llbracket m\hat{\theta} \rrbracket_{\mathbf{M}}$ from $\llbracket \mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta})) \rrbracket_{\mathbf{M}}$ with non-negligible probability. We then construct an adversary $\mathcal{A}_2$ to break the encryption scheme $\mathbf{\Pi}$.

Adversary 2 ($\mathcal{A}_2$) *The adversary is given the security parameter 1^η and an encryption oracle $O_b(\cdot,\cdot)$ about $\mathbf{eok}(m\hat{\theta})$. Given a query (i, m') where $k_i \in \mathbf{eok}(m\hat{\theta})$, $O_b(\cdot,\cdot)$ outputs $\mathbf{Enc}_{k_i}(m')$ if $b = 1$, or $\mathbf{Enc}_{k_i}(0^{|m'|})$ if $b = 0$.*

1. *$\mathcal{A}_2$ evaluates $m\hat{\theta}$ to get a value v_2:*
   (a) *Let $|\mathbf{sbk}(m\hat{\theta})| = \ell$. Construct an ℓ vector κ and an $(\ell \times n)$ array ς;*
   (b) *$\kappa[j](k_j \in \mathbf{sbk}(m\hat{\theta}))$ is initialized by sampling from $\mathbf{Gen}(1^\eta)$;*
   (c) *$\varsigma[j,1], \varsigma[j,2], \cdots, \varsigma[j,n](k_j \in \mathbf{sbk}(m\hat{\theta}))$ are initialized by sampling from $\mathbf{Crt}(\kappa[j], 1^\eta)$;*
   (d) *$m\hat{\theta}$ is evaluated to value v_2 according to Definition 19 except keys in $\mathbf{eok}(m\hat{\theta}) = \mathbf{keys}(m\hat{\theta}) \setminus \mathbf{sbk}(m\hat{\theta})$*[10]*. More precisely, a message in form of $\{\!|m'|\!\}_{k_i}$, where $k_i \in \mathbf{eok}(m\hat{\theta})$, is evaluated by submitting (i, m') to $O_b(\cdot,\cdot)$.*

[10] Since $\mathbf{eok}(m\hat{\theta}) \cap \mathbf{sbk}(m\hat{\theta}) = \emptyset$, the shares of keys in $\mathbf{eok}(m\hat{\theta})$ never occur in $m\hat{\theta}$.

2. $\mathcal{A}_2$ runs $\mathcal{D}_2(v_2, 1^\eta)$, and outputs whatever $\mathcal{D}_2(v_2, 1^\eta)$ outputs.

Since we deal with messages in the presence of key cycles and secret shares, one may wonder that if it is always feasible for the adversary $\mathcal{A}_2$ to construct a query submitted to oracle. After all, for any $m \in \mathbf{MSG}$, it seems that such a query may contain some keys or secret shares that $\mathcal{A}_2$ does not know. In fact, by using $m\hat{\theta}$ instead of m itself, considering that $\mathcal{A}_2$ knows all the keys in $\mathbf{sbk}(m\hat{\theta})$ and their shares, it is definitely feasible for $\mathcal{A}_2$ to construct such query without knowing $\mathbf{psk}(m)$. This is the reason why we need to construct an S-renaming $\hat{\theta}$ where $\theta(\mathbf{psk}(m)) \cap \mathbf{keys}(m) = \emptyset$..

Moreover, if $b = 1$, we can see that v_2 is just sampled from $[\![m\hat{\theta}]\!]_{\mathbf{M}}$, and if $b = 0$, v_2 is just sampled from $[\![\mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta}))]\!]_{\mathbf{M}}$. Considering that $\mathcal{A}_2$ outputs whatever $\mathcal{D}_2(v_2, 1^\eta)$ outputs, we have

$$\begin{aligned}
&\mathbf{Pr}[{\mathcal{A}_2}^{O_1}(1^\eta) = 1] - \mathbf{Pr}[{\mathcal{A}_2}^{O_0}(1^\eta) = 1] \\
=&\mathbf{Pr}\left[v_2 \leftarrow [\![m\hat{\theta}]\!]_{\mathbf{M}(\eta)} : \mathcal{D}_2(v_2, 1^\eta) = 1\right] - \\
&\mathbf{Pr}\left[v_2 \leftarrow [\![\mathbf{p}(m\hat{\theta}, \mathbf{sbk}(m\hat{\theta}))]\!]_{\mathbf{M}(\eta)} : \mathcal{D}_2(v_2, 1^\eta) = 1\right] \\
=&\varepsilon_2(\eta)
\end{aligned}$$

This shows that $\mathcal{A}_2$ can break $\mathbf{\Pi}$ with non-negligible probability, which is in contradiction with the **CPA** security of $\mathbf{\Pi}$. Therefore, $\varepsilon_2(\eta)$ is negligible, and this completes the lemma.

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 $[\![m]\!]_{\mathbf{M}} \approx [\![\mathbf{pattern}(m)]\!]_{\mathbf{M}}$.*

Proof. Let $\ell = |\mathbf{Keys}|$ be polynomially bounded in the security parameter η, from (5) and Definition 10, we have

$$\begin{aligned}
&[\![m]\!]_{\mathbf{M}} && [\![\mathbf{pattern}(m)]\!]_{\mathbf{M}} \\
=&[\![\mathbf{p}(m, \mathbf{keys}(m))]\!]_{\mathbf{M}} &=& [\![\mathbf{p}(m, \mathbf{FIX}(\mathcal{F}_m))]\!]_{\mathbf{M}} \\
=&[\![\mathbf{p}(m, \mathcal{F}_m^0(\mathbf{keys}(m)))]\!]_{\mathbf{M}} &=& [\![\mathbf{p}(m, \mathcal{F}_m^\ell(\mathbf{keys}(m)))]\!]_{\mathbf{M}}.
\end{aligned}$$

If we can show $[\![\mathbf{p}(m, \mathcal{F}_m^i(\mathbf{Keys}))]\!]_{\mathbf{M}} \approx [\![\mathbf{p}(m, \mathcal{F}_m^{i+1}(\mathbf{Keys}))]\!]_{\mathbf{M}}$, where $0 \leq i \leq \ell - 1$, then, by the transitivity of indistinguishability(see (12)), we can show $[\![\mathbf{p}(m, \mathcal{F}_m^0(\mathbf{Keys}))]\!]_{\mathbf{M}}$ is distinguishable from $[\![\mathbf{p}(m, \mathcal{F}_m^\ell(\mathbf{keys}(m)))]\!]_{\mathbf{M}}$, i.e., $[\![m]\!]_{\mathbf{M}} \approx [\![\mathbf{pattern}(m)]\!]_{\mathbf{M}}$. Let $\mathbf{K} = \mathcal{F}_m^i(\mathbf{keys}(m))$, $m' = \mathbf{p}(m, \mathbf{K})$. We have

$$\mathbf{p}(m, \mathcal{F}_m^i(\mathbf{keys}(m))) = \mathbf{p}(m, \mathbf{K}) = m' \tag{16}$$

$$\mathbf{p}(m, \mathcal{F}_m^{i+1}(\mathbf{keys}(m))) = \mathbf{p}(m, \mathcal{F}_m(\mathbf{K})) \tag{17}$$

Moreover, because $\mathbf{keys}(m)$ is the set of all keys occurring in m, we can get that $\mathcal{F}_m(\mathbf{keys}(m)) \subseteq \mathbf{keys}(m)$. According to Proposition 2 showed in section

2.2, $\mathcal{F}_m$ is monotone. So, we have $\mathcal{F}_m^{i+1}(\mathbf{keys}(m)) \subseteq \mathcal{F}_m^{i}(\mathbf{keys}(m))$, particularly, $\mathcal{F}_m(\mathbf{K}) \subseteq \mathbf{K}$, and thus

$$\mathcal{F}_m(\mathbf{K}) \cap \mathbf{K} = \mathcal{F}_m(\mathbf{K}) \tag{18}$$

Then, we have

$$\begin{aligned}
\mathbf{p}(m', \mathbf{rck}(m')) &= \mathbf{p}(\mathbf{p}(m, \mathbf{K}), \mathcal{F}_m(\mathbf{K})) && \text{by (8)}\\
&= \mathbf{p}(m, \mathbf{K} \cap \mathcal{F}_m(\mathbf{K})) && \text{by (6)}\\
&= \mathbf{p}(m, \mathcal{F}_m(\mathbf{K})) && \text{by (18)}
\end{aligned}$$

From lemma 4, we know that $[\![\mathbf{p}(m', \mathbf{rck}(m'))]\!]_{\mathbf{M}} \approx [\![m']\!]_{\mathbf{M}}$, and thus $[\![m']\!]_{\mathbf{M}} \approx [\![\mathbf{p}(m, \mathcal{F}_m(M))]\!]_{\mathbf{M}}$. Then, with (16) and (17), we get

$$[\![\mathbf{p}(m, \mathcal{F}_m^{i}(\mathbf{keys}(m)))]\!]_{\mathbf{M}} \approx [\![\mathbf{p}(m, \mathcal{F}_m^{i+1}(\mathbf{keys}(m)))]\!]_{\mathbf{M}}$$

and thus Lemma 5 holds.

Now, it is time for us to prove our main result, i.e., the computational soundness theorem.

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 12, 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 one of the following holds:

$$\mathbf{pattern}(m) = \mathbf{pattern}(m')\bar{\sigma} \tag{19}$$

$$\mathbf{pattern}(m) = (\mathbf{pattern}(m')\bar{\sigma})\hat{\theta} \tag{20}$$

From (19) and Lemma 1, we can get $[\![\mathbf{pattern}(m)]\!]_{\mathbf{M}} \approx [\![\mathbf{pattern}(m')]\!]_{\mathbf{M}}$. From (20), Lemma 2, and Lemma 1, we can also get $[\![\mathbf{pattern}(m)]\!]_{\mathbf{M}} \approx [\![\mathbf{pattern}(m')]\!]_{\mathbf{M}}$. So, we can conclude that, if $m \cong m'$, then

$$[\![\mathbf{pattern}(m)]\!]_{\mathbf{M}} \approx [\![\mathbf{pattern}(m')]\!]_{\mathbf{M}}. \tag{21}$$

Moreover, from Lemma 5, we have

$$[\![m]\!]_{\mathbf{M}} \approx [\![\mathbf{pattern}(m)]\!]_{\mathbf{M}}$$
$$[\![m']\!]_{\mathbf{M}} \approx [\![\mathbf{pattern}(m')]\!]_{\mathbf{M}}$$

Together with (12) and (21), we get

$$[\![m]\!]_{\mathbf{M}} \approx [\![m']\!]_{\mathbf{M}}$$

This completes our proof.

5 Conclusion

We proved 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.

These problems also bring significant challenges when proving computational soundness. Specifically, in proving computational soundness, we often need an adversary to evaluate a message with the help of encryption oracles(like $\mathcal{A}_2$ in proof of Lemma 4). 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 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. All these make our work significant and different from the previous work.

For future research, 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: TCS '00: Proceedings of the International Conference IFIP on Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics, London, UK, Springer-Verlag (2000) 3–22
2. Dolev, D., Yao, A.C.: On the security of public-key protocols. IEEE Transactions on Information Theory **30**(2) (1983) 198–208
3. Burrows, M., Abadi, M., Needham, R.: A logic of authentication. ACM Transactions on Computer Systems **8**(1) (February 1990) 18–36
4. Paulson, L.C.: The inductive approach to verifying cryptographic protocols. Journal of Computer Security **6** (1998) 85–128
5. Abadi, M., Gordon, A.D.: A calculus for cryptographic protocols: The spi calculus. Information and Computation **148**(1) (10 January 1999) 1–70
6. Goldwasser, S., Micali, S.: Probabilistic encryption. JCSS **28**(2) (April 1984) 270–299

7. Yao, A.C.: Theory and application of trapdoor functions. In: Proc. 23rd IEEE Symp. on Foundations of Comp. Science, Chicago (1982) 80–91
8. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In Stinson, D.R., ed.: Advances in Cryptology, Proc. CRYPTO' 93, *LNCS 773*, Springer (1994) 232–249
9. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42th IEEE Symposium on Foundations of Computers Science. (2001) 136–145
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.: Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings. Volume 2951 of Lecture Notes in Computer Science., Springer (2004) 133–151
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. Volume 3679 of Lecture Notes in Computer Science., Springer (2005) 374–396
14. Laud, P.: Symmetric encryption in automatic analyses for confidentiality against active adversaries. In: IEEE Symposium on Security and Privacy, IEEE Computer Society (2004) 71–85
15. Blanchet, B., Pointcheval, D.: Automated security proofs with sequences of games. In Dwork, C., ed.: CRYPTO. Volume 4117 of Lecture Notes in Computer Science., Springer (2006) 537–554
16. Abadi, M., Warinschi, B.: Security analysis of cryptographically controlled access to XML documents. Journal of the ACM **55**(2) (May 2008) 6:1–6:29
17. Micciancio, D.: Computational soundness, co-induction, and encryption cycles. In Gilbert, H., ed.: EUROCRYPT. Volume 6110 of Lecture Notes in Computer Science., Springer (2010) 362–380
18. Shamir, A.: How to share a secret. Communications of the ACM **22** (November 1979) 612–613
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, September 9–12, 2003, Berlin, Germany, Los Altos, CA 94022, USA, Morgan Kaufmann Publishers (2003) 898–909
20. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Pfitzmann, B., ed.: Advances in Cryptology – EUROCRYPT ' 2001. Volume 2045 of Lecture Notes in Computer Science., Innsbruck, Austria, Springer-Verlag, Berlin Germany (2001) 93–117
21. Laud, P.: Encryption cycles and two views of cryptography. In: Proceedings of the 7th Nordic Workshop on Secure IT Systems – NORDSEC ' 2002. 85–100
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) (2009) 737–797

23. Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In Nyberg, K., Heys, H.M., eds.: Selected Areas in Cryptography. Volume 2595 of Lecture Notes in Computer Science., Springer (2002) 62–75
24. Hofheinz, D., Unruh, D.: Towards key-dependent message security in the standard model. In Smart, N.P., ed.: EUROCRYPT. Volume 4965 of Lecture Notes in Computer Science., Springer (2008) 108–126
25. Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision diffie-hellman. In Wagner, D., ed.: CRYPTO. Volume 5157 of Lecture Notes in Computer Science., Springer (2008) 108–125
26. Haitner, I., Holenstein, T.: On the (im)possibility of key dependent encryption. In Reingold, O., ed.: TCC. Volume 5444 of Lecture Notes in Computer Science., Springer (2009) 202–219
27. Micciancio, D.: Pseudo-randomness and partial information in symbolic security analysis. Cryptology ePrint Archive, Report 2009/249 (2009) `http://eprint.iacr.org/`.