Information Sciences 430–431 (2018) 228–239

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Improved known-plaintext attack to permutation-only multimedia ciphers

Leo Yu Zhang[a,b,1,*], Yuansheng Liu[c,1], Cong Wang[d], Jiantao Zhou[e], Yushu Zhang[a], Guanrong Chen[f]

[a] School of Information Technology, Deakin University, Victoria 3125, Australia
[b] School of Information Technology, Jinan University, Guangdong 510632, China
[c] Advanced Analytics Institute, FEIT, University of Technology Sydney, Broadway, Australia
[d] Department of Computer Science, City University of Hong Kong, Hong Kong
[e] Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macau
[f] Department of Electronic Engineering, City University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 4 January 2017
Revised 1 November 2017
Accepted 14 November 2017
Available online 15 November 2017

Keywords:
Cryptanalysis
Known-plaintext attack (KPA)
Multimedia encryption
Permutation
Transposition

ABSTRACT

Permutation is a commonly used operation in many secure multimedia systems. However, it is fragile against cryptanalysis when used alone. For instance, it is well-known that permutation-only multimedia encryption is insecure against known-plaintext attack (KPA). There exist algorithms that are able to (partially) retrieve the secret permutation sequences in polynomial time with logarithmic amount of plaintexts in the number of elements to be permuted. But existing works fail to answer how many known plaintexts are needed to fully recover a underlying secret permutation sequence and how to balance the storage cost and computational complexity in implementing the KPA attack. This paper addresses these two problems. With a new concept of composite representation, the underlying theoretical rules governing the KPA attack on a permutation-only cipher are revealed, and some attractive algorithms outperforming the state-of-the-art methods in terms of computational complexity are developed. As a case study, experiments are performed on permutation-only image encryption to verify the theoretic analysis. The performance gap of the proposed KPA between artificial noise-like images, which perfectly fits the theoretical model, and the corresponding natural images is identified and analyzed. Finally, experimental results are shown to demonstrate the efficiency improvement of the new schemes over the existing ones.

1. Introduction

Permutation (or transposition) is a very common primitive in the field of information security. Considering whether or not the permutation is secret-key related, it can be divided into two categories: public permutation and secret permutation. By combining with certain nonlinear primitives, public permutation is adopted by many block ciphers with substitution-permutation network [12], for example AES [7]. For secret permutation, it can be viewed as a symmetric

* Corresponding author.
E-mail address: leocityu@gmai.com (L.Y. Zhang).
[1] L. Zhang and Y. Liu contributed equally to this work.

https://doi.org/10.1016/j.ins.2017.11.021
0020-0255/

cipher, where the secret for determining the permutation sequence serves as the key. With the ever-increasing popularity of multimedia services and the development of modern network technology, this simple model is widely used for protecting multimedia data in many applications. For example, in video surveillance, to preserve the privacy of the people appearing in a monitored area, secret permutation may be applied to some region-of-interest in the video streams [23].

The security of permutation-only multimedia encryption algorithms has been studied for a long time. Ciphertext-only attack (COA) takes effects on many permutation-only broadcast-TV systems because the specific structures of analog video signals limit the flexibility of designing complicated permutations [11]. For example, by exploiting the relationship between consecutive frames, Bertilsson et al. [1] proposed a COA attack on the design of Matias and Shamir [22], where each frame of a video is scanned along different pesudo-random space-filling curves, to partially recover the contents of the video.

When secret permutations are used to protect digitized multimedia data, the situation seems much more optimistic. This is because, according to the specific format of the digitized multimedia, the number of atoms that can be permuted is larger than its analog counterpart. Such atoms could be bits [8,10,30,42], bit-planes [30,42], pixels [5,31,38], pixel blocks [5,15,23], transform coefficients [5,9,27,32], variable-length codewords [32], tree nodes [6], motion vectors [25,32], prediction errors [5,25,41], and their various combinations [25,32]. For example, the atoms in Ye's image cryptosystem [30] are bits and atoms of Zhou et al's image encryption-then-compression system [41] are pixels, while atoms of the video encryption system of Zeng and Lei [32] are motion vectors and bits/blocks of transform coefficients.

Notably, COA may still be effective if there exist some correlations among the atoms to be permuted. For example, Li *et al's* demonstrated a COA attack on row-column shuffled images in [21] by exploiting the correlation between different rows and columns. And this method is later extended to crack permutation-only image encryption of pixel bits in [18]. Nevertheless, it is understandable that if the number of atoms, L, is not too small and the entropy contained in each atom is high, exhaustively searching for all the $(L!)$ combinations, and hence COA, is computationally infeasible[2]. Due to this fact, some studies suggest designing more elaborate methods to produce secret permutations, aiming to obtain better security and to simultaneously satisfy some other application-dependent requirements [15,25,32,38].

Despite the efforts devoted to improving the resistance of permutation-only ciphers against COA, all ciphers of this kind are fragile to plaintext attacks. This is easy to understand since some elements of the underlying permutation sequence can be determined by identifying unique atoms in available plaintext/ciphertext pairs. In the scenario of chosen-plaintext attack (CPA), which is a cryptanalysis model presuming that the attacker can obtain the ciphertext of an arbitrary plaintext, such effect can be maximized by forcing all atoms of the plaintext to be different from each other. With a permutation-only image cryptosystem and based on the results given in [20], Jolfaei et al. [16] showed that the lower bound of the number of chosen plaintexts to fully retrieve the underlying permutation sequence is $\lceil \log_r L \rceil$, where r is the number of different intensities.

In the scenario of known-plaintext attack (KPA), which is a cryptanalysis model that differs from CPA only by the assumption that the attacker cannot arbitrarily choose plaintext, it is somewhat complicated to deduce how many known plaintexts are required to fully retrieve the underlying permutation sequence. Generally, $L \gg r$ in multimedia data. So, some values in $\{0, 1, \ldots, r-1\}$ must appear more than once according to the pigeonhole principle. For example, the pixel value 0 will appear roughly 256 times in the permutation-only encrypted ciphertext if one assumes a uniform distribution of a known plain-image of size 256×256. Thus, by observing this single plain-image and the corresponding cipher-image, there will be $(256!)$ candidates for one single element in the permutation sequence, whose associated pixel's value is 0. Intuitively, the ambiguity in such a phenomenon can be gradually eliminated by observing more pairs of known plain-images and cipher-images. So, the key question about KPA attack to permutation-only ciphers is how many known plaintexts are needed to partially or fully reveal the permutation sequence with minimal computation and storage costs.

By extending the work in [40], Li et al. presented a quantitative analysis of KPA attack to permutation-only multimedia ciphers in [20]. Their method is composed of two steps: dividing a permutation sequence into different sets according to the values of atoms in each plaintext/ciphertext pair, and computing the intersection of the sets among different pairs. This method was improved in [19] in terms of storage and computational complexity by constructing a tree structure. Both these two works concluded that the number of known plaintexts is at the order of $\lceil \log_r L \rceil$. Due to the generality of these two works, they are popular for the design and analysis of lightweight multimedia encryption schemes [14,16,26,34–37,39,42].

This paper re-analyzes the KPA attack to permutation-only ciphers from the viewpoint of composite representation. This notion is not totally new. It was reported in a series of works by Bianchi et al. [2–4], demonstrating that it can be used to reduce the size of an encrypted message and speed up linear operations on encrypted messages that are produced by additive homomorphic cryptosystem. In this paper, the same notion is revisited for the purpose of comprehensively analyzing the number of required known plaintexts, partially or fully deducing the permutation sequence with minimal computation and storage costs. Differing from existing works [19,20,40], the contribution of this paper is two-fold:

- With composite representation, it offers a comprehensive theoretical analysis of KPA attack to permutation-only ciphers;
- The composite representation inherently implies different KPA algorithms, one of which outperforms the known "optimal" method in the sense of faster computation with the same storage;

The rest of this paper is organized as follows. The formalized model of the considered KPA attack followed by a review of the methods of [19,20] is given in Section 2. Based on composite representation, Section 3 deduces a necessary and

[2] It is noted that $(L!) > O(2^{80})$ is out of the reach by the state-of-the-art computing capabilities and the foreseeable future.

sufficient condition for determining the underlying permutation sequence, and then heuristically designs three algorithms, all of which run faster than existing ones. To that end, all the proposed algorithms are verified by performing extensive simulations in Section 4 and the last section draws some conclusions. The codes for reproducing the results in Section 4 are available online [33].

2. Problem formulation and related works

Before reviewing the details of the works [19,20] in the literature, some necessary notations and definitions for describing KPA attacks are first given.

2.1. Formalization of the KPA attack

As mentioned earlier, in digital multimedia data, the atoms to be permuted can be in various forms, such as bits, pixels and transform coefficients. Also, those atoms can be organized in different ways. For example, one can scan an image to 1D data and permute the 1D sequence, or directly perform permutation on the original form of the image (2D matrix), or even stack the image to a 3D cube before permutation. The notations of the KPA attack would vary with the considered form. In particular, the permutations considered by [16,19,20,40] are in 2D form. Note that permutation in one form can be converted to other forms, the permutation considered here is carried out on L atoms (1D), whose index set can be written as $\mathbb{L} = \{l \mid 1, 2, \ldots, L\}$ with $\#\mathbb{L} = L$. Each atom takes a value from the set $\{0, 1, \ldots, r-1\}$ and an atom is said to be unique if its value is different from all the other $L-1$ ones. Typically, $r \ll L$ for multimedia data. For example, when the permutation is carried out on pixels of a 256×256 gray-scale image, one has $L = 256^2$ (the 2D image is vectorized to 1D) and $r = 256$.

Definition 1. Secret permutation $\Pi_k : \mathbb{L} \to \mathbb{L}$ is a bijection, which is determined by a secret key k, that maps elements of $\mathbb{L}$ to itself.

Cauchy's two-line notation will be used to represent a secret permutation, i.e.,

$$\begin{pmatrix} 1 & 2 & 3 & \cdots & L \\ \pi_k(1) & \pi_k(2) & \pi_k(3) & \cdots & \pi_k(L) \end{pmatrix},$$

and it will be written as $[\pi_k(1), \pi_k(2), \pi_k(3), \ldots, \pi_k(L)]$ for simplicity. Typically, different k assigns different permutations and $\Pi_{\tilde{k}}$ is called a permutation instance when $k = \tilde{k}$. Π_k is pseudo-random if its instances are uniformly drawn from all the $(L!)$ possible candidates. For example, Π_k can be implemented via a combination of AES and the Knuth shuffle algorithm [17].

Definition 2. Permutation-only ciphers are a special kind of symmetric encryption algorithms, which encrypt a plaintext $\mathbf{P} = [p(1), p(2), p(3), \ldots, p(L)]$ by

$$\begin{aligned} \mathbf{C} &= [c(1), c(2), c(3), \ldots, c(L)] \\ &= E_k(\mathbf{P}) \\ &= [p(\pi_k(1)), p(\pi_k(2)), p(\pi_k(3)), \ldots, p(\pi_k(L))]. \end{aligned}$$

The decryption of $\mathbf{C}$ can be done by using the inverse of Π_k. Referring to Shannon's maxim, the security of a cryptosystem is solely dependent on the secrecy of the key. Since $[\pi_k(l)]_{l=1}^{L}$ plays the same role as k during encryption/decryption, once the method used for producing $[\pi_k(l)]_{l=1}^{L}$ is determined, only the task of deducing $[\pi_k(l)]_{l=1}^{L}$ is considered hereinafter[3].

2.2. Related works

The two state-of-the-art works [19,20] about KPA attack to permutation-only ciphers are reviewed here. Roughly speaking, Li et al.'s method [20] determines an approximate version of $[\pi(l)]_{l=1}^{L}$ by solving the intersection of the candidate sets that are determined by different plaintext/ciphertext pairs. Assume that n plaintext/ciphertext pairs, $(\mathbf{P}_i, \mathbf{C}_i) = ([p_i(1), p_i(2), p_i(3), \ldots, p_i(L)], [c_i(1), c_i(2), c_i(3), \ldots, c_i(L)])$ $(i = 1 \sim n)$, are available, their algorithm can be summarized as follows:

- Step 1. Determine $(n \cdot r)$ index sets as

  $$\Lambda_1(0), \ldots, \Lambda_1(r-1), \ldots, \Lambda_n(0), \ldots, \Lambda_n(r-1),$$

  where $\Lambda_i(j) \subset \mathbb{L}$ denotes a set containing all the indexes of the atoms in $\mathbf{P}_i$ whose associated value equals j, i.e., $p_i(l) = j$ for all $l \in \Lambda_i(j)$.

[3] Write $\pi_k(l)$ as $\pi(l)$ when it does not cause any ambiguity.

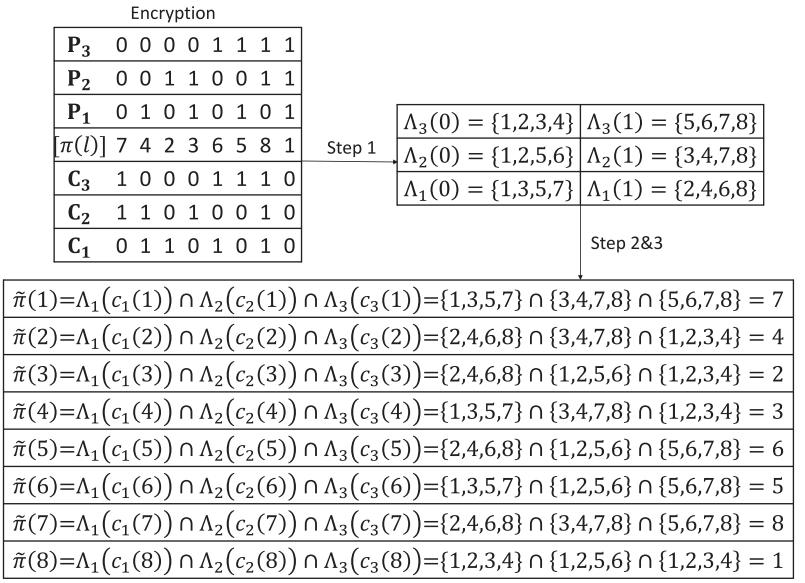


Fig. 1. Example of the KPA attack by Li et al. [20].

- Step 2. Produce a multi-valued permutation vector $[\hat{\pi}(l)]_{l=1}^{L}$ (each element in the vector represents a set), using

$$\hat{\pi}(l) = \bigcap_{i=1}^{n} \Lambda_i(c_i(l)).$$

- Step 3. Determine $[\tilde{\pi}(1), \ldots, \tilde{\pi}(L)]$, the approximate version of $[\pi(l)]_{l=1}^{L}$, from $[\hat{\pi}(l)]_{l=1}^{L}$ with the rule that $\tilde{\pi}(l) \in \hat{\pi}(l)$ and $\tilde{\pi}(l_1) \neq \tilde{\pi}(l_2)$ for all $l_1 \neq l_2$.

Note that $[\tilde{\pi}(l)]_{l=1}^{L}$ may not be the same as $[\pi(l)]_{l=1}^{L}$ since $\#\hat{\pi}(l) > 1$ may be true if n is not large enough[4]. As can be observed from the above, the storage of the KPA by Li et al. is $O(nL)$, which is created by Step 1, and its computational complexity is $O(nL^2)$, which is caused by computing the intersection of the candidate sets in Step 2. For illustration, an example with $\mathbb{L} = \{1, 2, \ldots, 8\}$, $r = 2$ and $[\pi(l)]_{l=1}^{8} = [7, 4, 2, 3, 6, 5, 8, 1]$ is depicted in Fig. 1.

In [19], Li and Lo improved the attack suggested in [20] in terms of computational complexity by taking advantage of an r-ary tree. Their method can be characterized by the following steps:

- Step 1. Set $i = 1$ and construct the root node, which contains two index sets, Λ_0 and Λ'_0, with $\Lambda_0 = \Lambda'_0 = \mathbb{L} = \{1, 2, \ldots, L\}$;
- Step 2. Construct r child nodes by visiting the ith known plaintext/ciphertext pair ($\mathbf{P}_i$, $\mathbf{C}_i$). Every node has two index sets, $\Lambda_i(j)$ and $\Lambda'_i(j)$, which satisfy

$$\begin{cases} \Lambda_i(j) = \Lambda_{i-1}(j) \bigcap \{l \mid p_i(l) = j\}, \\ \Lambda'_i(j) = \Lambda'_{i-1}(j) \bigcap \{l \mid c_i(l) = j\}, \end{cases}$$

  where $l \in [1, L]$, $\Lambda_0(j) = \Lambda_0$ and $\Lambda'_0(j) = \Lambda'_0$ for $j = 0 \sim r - 1$.
- Step 3. If $i < n$, increase i by 1 and go to Step 2.
- Step 4. Determine $[\tilde{\pi}(1), \ldots, \tilde{\pi}(L)]$, the approximate version of $[\pi(l)]_{l=1}^{L}$, from the leaf nodes with the rule that $\tilde{\pi}(l) \in \Lambda_n(j)$ for $l \in \Lambda'_n(j)$ and $\tilde{\pi}(l_1) \neq \tilde{\pi}(l_2)$ for all $l_1 \neq l_2$.

Several remarks on this method are made in order as follows:

1. From Step 2, it can be observed that adding new nodes to the ith level relies only on nodes at the $(i-1)$th level. So, the algorithm consumes $O(4L)$ storage to maintain the index sets, $\Lambda_i(j)$, $\Lambda'_i(j)$, $\Lambda_{i-1}(j)$ and $\Lambda'_{i-1}(j)$.
2. From Step 2, the calculation of $\Lambda_i(j)$ is actually a refinement of the previous result $\Lambda_{i-1}(j)$. The computational complexity of Step 2 is $O(2L)$, and thus the total computational complexity of the algorithm is $O(2nL)$.

[4] If $\#\hat{\pi}(l) > 1$, one can simply take the first unused element from $\hat{\pi}(l)$ as $\tilde{\pi}(l)$. This "taking-the-first" method is employed in [20, Section 3.1] and in the following algorithms of this paper.

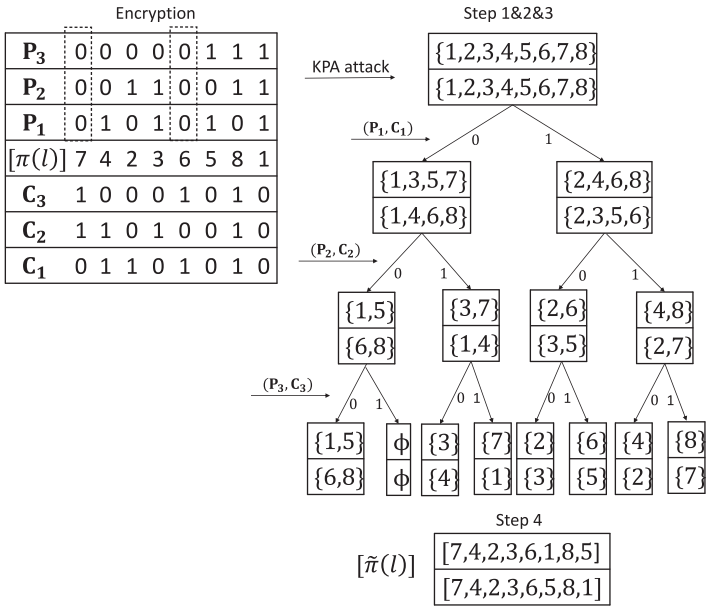


Fig. 2. Example of the KPA attack by Li and Lo [19].

For illustration, an example of this attack with $\mathbb{L} = \{1, 2, \ldots, 8\}$, $r = 2$ and $[\pi(l)]_{l=1}^{8} = [7, 4, 2, 3, 6, 5, 8, 1]$ is depicted in Fig. 2.

Under the assumption that the known plaintexts are uniformly distributed, the authors of [19,20] estimated that the number of known plaintexts for the correct retrieval of $[\pi(l)]_{l=1}^{L}$ should be $O(\lceil \log_r L \rceil)$. This claim is easy to understand from Step 2 of Li and Lo's algorithm and the binary tree in Fig. 2, where $\#\Lambda_{i-1}(j)$ will shrink down to $1/r$ of its original size with the information supplied by the ith plaintext/ciphertext pair. So, with $n = O(\lceil \log_r L \rceil)$ pairs of plaintext/ciphertext[5], one has $\#\Lambda_n(j) = L/r^n \to 1$.

3. Known-plaintext attack based on composite representation

Based on composite representation, this section first establishes a necessary and sufficient condition for determining $[\pi(l)]_{l=1}^{L}$. Then, it will be shown how this condition can be employed to devise concrete KPA algorithms by incorporating different data structures (e.g., array, tree and hash table) to gain better computational efficiency.

3.1. A necessary and sufficient condition

The proposed analysis is built on top of a concept called composite representation, which is defined as follows.

Definition 3. Given an index sequence $[\rho(1), \rho(2), \ldots, \rho(n)]$ ($\rho(i) \in [1, n]$ and $\rho(i) \neq \rho(j)$ for all $i \neq j$), the vector $\tilde{\mathbf{P}}^n = [\tilde{p}^n(1), \tilde{p}^n(2), \tilde{p}^n(3), \ldots, \tilde{p}^n(L)]$ is called the composite representation of n plaintexts $\{\mathbf{P}_i = [p_i(1), p_i(2), p_i(3), \ldots, p_i(L)]\}_{i=1}^{n}$ ($p_i(l) \in [0, r-1]$) with respect to $[\rho(i)]_{i=1}^{n}$ if

$$\tilde{p}^n(l) = \sum_{i=1}^{n} p_{\rho(i)}(l) \cdot r^{i-1}, \tag{1}$$

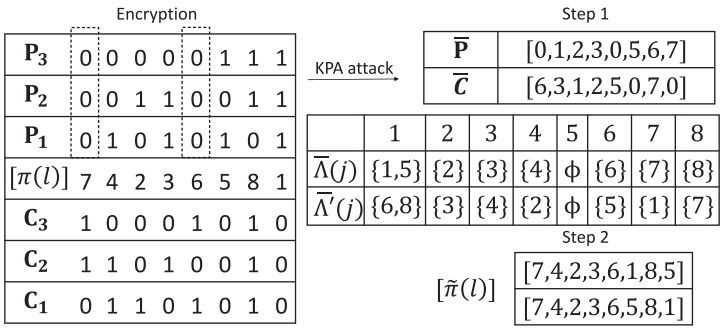
where $l = 1 \sim L$.

From Eq. (1), it can be concluded that the composite atom $\tilde{p}^n(l) \in [0, r^n - 1]$. Using this definition, the following two propositions reveal a necessary and sufficient condition for determining $[\pi(l)]_{l=1}^{L}$.

Proposition 1. *For a given index sequence $[\rho(1), \rho(2), \ldots, \rho(n)]$ and n known plaintexts $\{\mathbf{P}_i = [p_i(1), p_i(2), p_i(3), \ldots, p_i(L)]\}_{i=1}^{n}$, the composite representation $\tilde{\mathbf{C}}^n = [\tilde{c}^n(1), \tilde{c}^n(2), \tilde{c}^n(3), \ldots, \tilde{c}^n(L)]$ of the ciphertexts corresponding to these n known plaintexts under the permutation sequence $[\pi(l)]_{l=1}^{L}$ is given by*

$$\tilde{c}^n(l) = \tilde{p}^n(\pi(l))$$

[5] The analysis given in [19,20] is different from what is used here, but their rationales are the same. Moreover, it will be shown in Section 3 that this analysis is not comprehensive since it only focuses on the case of determining a single element of $[\pi(l)]_{l=1}^{L}$.



$\mathbf{P}_3$	0	0	0	0	0	1	1	1
$\mathbf{P}_2$	0	0	1	1	0	0	1	1
$\mathbf{P}_1$	0	1	0	1	0	1	0	1
$[\pi(l)]$	7	4	2	3	6	5	8	1
$\mathbf{C}_3$	1	0	0	0	1	0	1	0
$\mathbf{C}_2$	1	1	0	1	0	0	1	0
$\mathbf{C}_1$	0	1	1	0	1	0	1	0

$\bar{\mathbf{P}}$	[0,1,2,3,0,5,6,7]
$\bar{\boldsymbol{C}}$	[6,3,1,2,5,0,7,0]

	1	2	3	4	5	6	7	8
$\bar{\Lambda}(j)$	{1,5}	{2}	{3}	{4}	ϕ	{6}	{7}	{8}
$\bar{\Lambda}'(j)$	{6,8}	{3}	{4}	{2}	ϕ	{5}	{1}	{7}

$[\tilde{\pi}(l)]$	[7,4,2,3,6,1,8,5]
	[7,4,2,3,6,5,8,1]

Fig. 3. Example of the proposed trading-space-for-time algorithm (Algorithm 1).

$$= \sum_{i=1}^{n} p_{\rho(i)}(\pi(l)) \cdot r^{i-1}$$
$$= \sum_{i=1}^{n} c_{\rho(i)}(l) \cdot r^{i-1}. \tag{2}$$

Proof. The proof of this proposition follows directly from Definition 1 and Definition 3. □

Proposition 2. *For any index sequence* $[\rho(1), \rho(2), \ldots, \rho(n)]$, *given n known plaintext/ciphertext pairs* $(\mathbf{P}_1, \mathbf{C}_1), \ldots, (\mathbf{P}_n, \mathbf{C}_n)$, *the* l_0*th entry of the permutation sequence* $[\pi(l)]_{l=1}^{L}$, $\pi(l_0)$, *can be uniquely determined if and only if the* l_0*th composite atom of* $\bar{\mathbf{C}}^n$ *is unique.*

Proof. Suppose that the index sequence is fixed to be $[\hat{\rho}(1), \hat{\rho}(2), \ldots, \hat{\rho}(n)]$. Referring to Eq. (2), the l_0th composite atom of $\bar{\mathbf{C}}^n$ is unique if and only if the $\pi(l_0)$th composite atom of $\bar{\mathbf{P}}^n$ is unique. So, the proposition is true for a fixed index sequence.

Next, using the idea of proof-by-contradiction, it can be shown that when the index sequence is changed, the new composite atom at the l_0th entry is still unique. Suppose a new index sequence $[\tilde{\rho}(1), \tilde{\rho}(2), \ldots, \tilde{\rho}(n)]$ is used and $\bar{c}^n(l_0) = \sum_{i=1}^{n} c_{\tilde{\rho}(i)}(l_0) \cdot r^{n-1}$ is not unique. Then, there exists an $l_1 \in [1, L]$ with $l_1 \neq l_0$ such that $\bar{c}^n(l_0) = \bar{c}^n(l_1)$. This implies that

$$c_i(l_0) = c_i(l_1)$$

for all $i \in [1, n]$. This contradicts the assumption that, under the original index sequence $[\hat{\rho}(1), \hat{\rho}(2), \ldots, \hat{\rho}(n)]$, the l_0th composite atom of $\bar{\mathbf{C}}^n$ is unique. □

3.2. Algrithm 1: A trading-space-for-time approach

As implied by the above analysis, the determination of the underlying permutation sequence $[\pi(l)]_{l=1}^{L}$ under KPA can be converted to the problem of identifying unique (composite) atoms in a composite known plaintext/ciphertext pair. Moreover, as indicated by Proposition 2, the order of how the composite representation is generated does not affect the performance. With this result, for n known plaintext/ciphertext pairs $(\mathbf{P}_i, \mathbf{C}_i) = ([p_i(1), p_i(2), p_i(3), \ldots, p_i(L)], [c_i(1), c_i(2), c_i(3), \ldots, c_i(L)])$ $(i = 1 \sim n)$, it is possible to propose a simple algorithm (call Algorithm 1 hereinafter) by trading-space-for-time, as follows:

- Step 1. Construct an array of length r^n, whose element is composed of two index sets, $\bar{\Lambda}(j) = \emptyset$ and $\bar{\Lambda}'(j) = \emptyset$, for $j = 1 \sim r^n$. Update $[\bar{\Lambda}(j)]_{j=1}^{r^n}$ and $[\bar{\Lambda}'(j)]_{j=1}^{r^n}$ using the rule

$$\begin{cases} \bar{\Lambda}(j) = \{l \mid \bar{p}^n(l) = j - 1\} \bigcup \bar{\Lambda}(j), \\ \bar{\Lambda}'(j) = \{l \mid \bar{c}^n(l) = j - 1\} \bigcup \bar{\Lambda}'(j), \end{cases}$$

  where $\bar{p}^n(l)$ and $\bar{c}^n(l)$ $(l = 1 \sim L)$ are the lth composite atoms of $\bar{\mathbf{P}}^n$ and $\bar{\mathbf{C}}^n$ that are determined by

$$\begin{cases} \bar{p}^n(l) = \sum_{i=1}^{n} p_i(l) \cdot r^{i-1}, \\ \bar{c}^n(l) = \sum_{i=1}^{n} c_i(l) \cdot r^{i-1}. \end{cases}$$

- Step 2. Determine $[\tilde{\pi}(1), \ldots, \tilde{\pi}(L)]$, the approximate version of $[\pi(l)]_{l=1}^{L}$, under the rule that $\tilde{\pi}(l) \in \bar{\Lambda}(j)$ for $l \in \bar{\Lambda}'(j)$ and $\tilde{\pi}(l_1) \neq \tilde{\pi}(l_2)$ for all $l_1 \neq l_2$.

For illustration, an example of this algorithm is depicted in Fig. 3 with the same setting as that used in Fig. 2. It is clear that Algorithm 1 requires $O(r^n)$ space to store the array and $O(nL)$ computation to produce the composite plaintext $\bar{\mathbf{P}}^n$ and ciphertext $\bar{\mathbf{C}}^n$. It is also understandable that Algorithm 1 is faster than all existing schemes [19,20,40] since the determination of $[\pi(l)]_{l=1}^{L}$ reduces to a simple traversal of $\bar{\mathbf{P}}^n$ and $\bar{\mathbf{C}}^n$ after consuming a storage exponential to the number

of known plaintexts. Despite being storage expensive, it provides a capability to intuitively interpret the theoretic foundation for how KPA works on permutation-only ciphers, while all previous studies [19,20,40] cannot.

Assume that the plaintexts are uniformly distributed and[6] $N = r^n \gg L$. Referring to Proposition 2, the permutation sequence $[\pi(l)]_{l=1}^{L}$ can be fully revealed if and only if all the L atoms of $\bar{\mathbf{C}}^n$ are unique. It is worth mentioning that this phenomenon is exclusive to the well-known birthday paradox problem [28], which considers the probability that, in a set of L randomly chosen atoms, at least two of them have the same value. So, following deduction of the birthday paradox problem, the probability that the permutation sequence $[\pi(l)]_{l=1}^{L}$ is fully revealed can be calculated as

$$\begin{aligned}\text{Prob}(full\ recovery) &= \frac{N}{N}\cdot\frac{N-1}{N}\cdots\frac{N-(L-1)}{N}\\ &= 1\cdot\left(1-\frac{1}{N}\right)\cdots\left(1-\frac{L-1}{N}\right) \qquad (3)\\ &\approx e^{-\frac{(1+2+\cdots+(L-1))}{N}}\\ &= e^{\frac{(L-L^2)}{2N}}, \qquad (4)\end{aligned}$$

where Eq. (4) has applied the approximation $e^x = 1 + x$ when $|x| \ll 1$. Set $\text{Prob}(full\ recovery) \geq 50\%$ and deduce that $N = r^n \geq \frac{L^2-L}{\ln 4}$. In other words, under KPA, to successfully recover $[\pi(l)]_{l=1}^{L}$ with a probability more than a half, the number of known plaintext/ciphertext pairs should be in the order of $n = O(\lceil 2\cdot\log_r L\rceil)$.

Obviously, this result is different from $O(\lceil\log_r L\rceil)$, which is given in [19,20] and reviewed in Section 2 above. As mentioned before, such an analysis is not comprehensive since it only focuses on the event of recovering a single element of $[\pi(l)]_{l=1}^{L}$. With $n = O(\lceil\log_r L\rceil)$, it is easy to conclude that $\text{Prob}(\#\Lambda_n(j) \to 1) \approx 1/r^n \approx 1/L$ and, thus, $\text{Prob}(full\ recovery) \approx 1/L^L \approx 1/r^{nL} \to 0$. For example, letting $r = 2$ (which corresponds to binary image) and $L = 128\times 128$, one has $\log_r L = 14$, and it is easy to observe that $\text{Prob}(full\ recovery) \approx 1/2^{14\times 128\times 128}$ is negligible.

The above discussion focuses on the case that the permutation sequence is totally recovered. However, for multimedia signals considered in this work, partial recovery of $[\pi(l)]_{l=1}^{L}$ may still lead to a satisfactory visual effect since human eyes have very strong capability of suppressing noises and extracting features [24,29] (referring to the experiment results in Sections 4.1 and 4.2 for visual verifications). So, the average recovery of $[\pi(l)]_{l=1}^{L}$ is further studied in the following. Let X represent the possible number of elements of $[\pi(l)]_{l=1}^{L}$ that are correctly retrieved under the KPA attack. Then, the expected value of X is

$$\text{E}(X) = L\cdot\left(1-\frac{1}{N}\right)^{L-1}. \qquad (5)$$

For example, when $r = 2$, $L = 128\times 128$ and $n = 14$, one has $\text{E}(X)/L\times 100\% \approx 37\%$.

3.3. Improved algorithms: Better space/time trade-off approaches

Algorithm 1 is with low computational complexity at the cost of an exponentially increasing storage. It is not good for practical usage when n, the number of plaintext/ciphertext pairs, is not small. However, the concept of composite representation can be used for designing other KPA approaches of permutation-only multimedia ciphers with better space/time trade-off, as follows:

- Applying composite representation to Li and Lo's tree structure [19], thereby better balancing the trade-off between storage and computation (call improved Li and Lo's algorithm hereinafter);
- Grouping the composite atoms into different categories according to their hash values so as to reduce the storage cost with little extra computations (call Algorithm 2 hereinafter).

From a high level point of view, the improved Li and Lo's algorithm gradually constructs an r^{n_0}-ary tree by using composite plaintext/ciphertext that is composed of n_0 pairs of original plaintext/ciphertext. In particular, with n plaintext/ciphertext pairs $(\mathbf{P}_1, \mathbf{C}_1), \ldots, (\mathbf{P}_n, \mathbf{C}_n)$, it produces a tree of depth $m = n/n_0$ as follows[7]:

- Step 1. Set $i = 1$ and construct the root node, which contains two index sets, $\bar{\Lambda}_0$ and $\bar{\Lambda}'_0$, with $\bar{\Lambda}_0 = \bar{\Lambda}'_0 = \mathbb{L} = \{1, 2, \ldots, L\}$;
- Step 2. Construct r^{n_0} child nodes by visiting the i-th composite plaintext/ciphertext pair $(\bar{\mathbf{P}}_i^{n_0}, \bar{\mathbf{C}}_i^{n_0}) = ([\bar{p}_i^{n_0}(1), \ldots, \bar{p}_i^{n_0}(L)], [\bar{c}_i^{n_0}(1), \ldots, \bar{c}_i^{n_0}(L)])$, where

$$\begin{cases}\bar{p}_i^{n_0}(l) = \sum_{i'=1}^{n_0} p_{(i-1)n_0+i'}(l)\cdot r^{i'-1},\\ \bar{c}_i^{n_0}(l) = \sum_{i'=1}^{n_0} c_{(i-1)n_0+i'}(l)\cdot r^{i'-1},\end{cases}$$

[6] For full recovery of $[\pi(l)]_{l=1}^{L}$ with a non-negligible probability, the requirement of $N \gg L$ is necessary.

[7] For simplicity, assume n_0 divides n.

Table 1
Storage and computational complexity of all the KPA algorithms ($m = n/n_0$).

	Li et al's algorithm [20]	Li and Lo's algorithm [19]	Algorithm 1	Improved Li and Lo's algorithm	Algorithm 2
Storage	$O(nL)$	$O(4L)$	$O(r^n)$	$O(6L)^*$	$O(4L)$
Computation	$O(nL^2)$	$O(2nL)$	$O(nL)$	$O(nL+2mL)^*$	$O(nL+\varepsilon)$

* When $m = n$, the storage and computational complexity of this improved algorithm and Li and Lo's algorithm is the same since composition becomes unnecessary.

for $l = 1 \sim L$. Every node is composed of two index sets, $\bar{\Lambda}_i(j)$ and $\bar{\Lambda}'_i(j)$, which satisfy

$$\begin{cases} \bar{\Lambda}_i(j) = \bar{\Lambda}_{i-1}(j) \bigcap \{l \mid \bar{p}_i^{n_0}(l) = j\}, \\ \bar{\Lambda}'_i(j) = \bar{\Lambda}'_{i-1}(j) \bigcap \{l \mid \bar{c}_i^{n_0}(l) = j\}, \end{cases}$$

where $j = 0 \sim r^{n_0} - 1$.

- Step 3. If $i < m$, increase i by 1 and go to Step 2.
- Step 4. Determine $[\tilde{\pi}(1), \ldots, \tilde{\pi}(L)]$, the approximate version of $[\pi(l)]_{l=1}^{L}$, from the leaf nodes, under the rule that $\tilde{\pi}(l) \in \bar{\Lambda}_m(j)$ for $l \in \bar{\Lambda}'_m(j)$ and $\tilde{\pi}(l_1) \neq \tilde{\pi}(l_2)$ for all $l_1 \neq l_2$.

Besides $O(4L)$ storage to maintain $\bar{\Lambda}_i(j)$, $\bar{\Lambda}'_i(j)$, $\bar{\Lambda}_{i-1}(j)$ and $\bar{\Lambda}'_{i-1}(j)$, this improved scheme also consumes $O(2L)$ storage for the composite plaintext/ciphertext pair $(\bar{\mathbf{P}}_i^{n_0}, \bar{\mathbf{C}}_i^{n_0})$. So, the total storage of this algorithm is $O(6L)$. Similar to Algorithm 1, the computational complexity for computing $(\bar{\mathbf{P}}_i^{n_0}, \bar{\mathbf{C}}_i^{n_0})$ is $(n_0 L)$, and similar to Step 2 of Li and Lo's method, the computational complexity for updating $\bar{\Lambda}_i(j)$ and $\bar{\Lambda}'_i(j)$ is $O(2L)$. So, the total computational complexity of this algorithm is $O(m \times (n_0 L + 2L)) = O(nL + 2mL)$.

Differing from the above discussed improved version of Li and Lo's method, which still involves constructing a tree structure iteratively, one may construct a hash table by dividing all the composite plaintext/ciphertext atoms according to their hash values. In this way, finding the location information associated with a unique (composite) atom can be done as fast as Algorithm 1. Since the hash function used here is only for classification, the simple mapping $f(x) = x \bmod L$ is sufficient for this purpose [13]. With this mapping, Algorithm 2 can be characterized by the following steps:

- Step 1. Construct a sequence of length L, in which each element contains four sets $\bar{\Lambda}(j)$, $V(j)$, $\bar{\Lambda}'(j)$ and $V'(j)$ (initialized to be empty for $j = 1 \sim L$). Update these sets using the information provided by the composite plaintext/ciphertext, via

$$\begin{cases} \bar{\Lambda}(j) = \bar{\Lambda}(j) \bigcup \{l \mid \bar{p}^n(l) \bmod L = j - 1\}, \\ \bar{\Lambda}'(j) = \bar{\Lambda}'(j) \bigcup \{l \mid \bar{c}^n(l) \bmod L = j - 1\}, \\ V(j) = V(j) \bigcup \{p^n(l) \mid \bar{p}^n(l) \bmod L = j - 1\}, \\ V'(j) = V'(j) \bigcup \{c^n(l) \mid \bar{c}^n(l) \bmod L = j - 1\}, \end{cases}$$

where $\bar{p}^n(l)$ and $\bar{c}^n(l)$ $(l = 1 \sim L)$ are the lth composite atoms of $\bar{\mathbf{P}}^n$ and $\bar{\mathbf{C}}^n$, given by

$$\begin{cases} \bar{p}^n(l) = \sum_{i=1}^{n} p_i(l) \cdot r^{i-1}, \\ \bar{c}^n(l) = \sum_{i=1}^{n} c_i(l) \cdot r^{i-1}. \end{cases}$$

- Step 2. Determine $[\tilde{\pi}(1), \ldots, \tilde{\pi}(L)]$, the approximate version of $[\pi(l)]_{l=1}^{L}$, under the following rules:
  - if $\#\bar{\Lambda}(j) = 1$ (or equivalently $\#\bar{\Lambda}'(j) = 1$), let $\tilde{\pi}(l) \in \bar{\Lambda}(j)$ for $l \in \bar{\Lambda}'(j)$;
  - if $\#\bar{\Lambda}(j) > 1$ (or equivalently $\#\bar{\Lambda}'(j) > 1$), set $\tilde{\pi}(l) \in \bar{\Lambda}(j)$ for $l \in \bar{\Lambda}'(j)$ under the constraint that $\bar{p}^n(\tilde{\pi}(l)) = \bar{c}^n(l)$ $(\bar{p}^n(\tilde{\pi}(l)) \in V(j)$, $\bar{c}^n(l) \in V'(j))$ and $\tilde{\pi}(l_1) \neq \tilde{\pi}(l_2)$ for all $l_1 \neq l_2$.

The storage complexity of Algorithm 2 is $O(4L)$, which is caused by storing all the indexes and values of the composite atoms. The computational complexity of Algoritm 2 is $O(nL)$, which is resulted from grouping the composite plaintext/ciphertext to L different residue classes in Step 1. Nevertheless, the second item of Step 2 introduces some extra computations since it requires comparing the values of the composite atoms in $V(j)$ and $V'(j)$. However, by the hypothesis that composite plaintexts/ciphertexts are uniformly distributed, the size of $V(j)$ (or $V'(j)$) is in the order of $O(\frac{L}{r^n} \times \lceil \frac{r^n}{L} \rceil) = O(1)$. Also, the larger the n is, the smaller the $\#\bar{\Lambda}(j)$ will be. Finally, it can be concluded that the computational complexity of Algorithm 2 is[8] $O(nL + \varepsilon)$.

As a concluding remark of this section, the storage and computational complexity of all the mentioned KPA algorithms for permutation-only ciphers are summarized in Table 1. Since big-O notation hides large constant, from Table 1, one can only observe an $O(1)$-time computational efficiency improvement of the proposed algorithms over Li and Lo's algorithm [19]. Nevertheless, referring to the details of their algorithm, it requires the comparison of all the $2nL$ atoms of the plaintexts/ciphertexts with r values to construct the tree structure (Steps $2 \sim 3$). In Algorithm 1 here, no comparison is

[8] Here, ε is a small constant that differentiates Algorithm 2 from Algorithm 1.

Fig. 4. KPA attack of encrypted binary image using artificial known plaintexts: (a) the most significant bit plane of the Lena image; (b) the encryption of Fig. 4(a); (c) a decryption instance with $n = 14$; (d) a decryption instance with $n = 22$.

needed thanks to the strategy of trade-space-for-time. In the improved Li and Lo's algorithm, such comparisons are reduced from $2nL$ to $2mL$ ($m = n/n_0$). And, in Algorithm 2 here, comparisons are only performed in residue classes, which contain only a few atoms. As the further experimental results in Section 4.3 demonstrate, for different values of L, the improved Li and Lo's algorithm ($m = 2$) runs roughly $4 \sim 5$ times faster than the method in [19] and Algorithm 2 runs roughly 100 times faster than that of [19].

4. Experiments on permutation-only image ciphers

Experiments are performed on permutation-only image ciphers to verify the performance of the KPA analysis presented above. Firstly, the effectiveness of the algorithms is tested using known artificial plain-images, so that the uniform distribution assumption can be fulfilled to fit the theoretic analysis in Section 3.2. Then, attacks are performed on some natural images, whose distributions are not uniform, used as the known plaintexts. The performance gap of the KPA attacks between known artificial plain-images and natural plain-images is then identified and analyzed. Finally, the efficiency of all the methods is evaluated by experiments on images of different sizes. Without loss of generality, Π_k is implemented using AES-CTR and the Knuth shuffle in all the following tests.

4.1. Verification of the theoretical results with known artificial plaintexts

For the 128×128 binary image shown in Fig. 4(a), it is encrypted with Π_k and the result is shown in Fig. 4(b). Under this setting, $2 \cdot \lceil \log_r L \rceil = 2 \cdot \lceil \log_2 128^2 \rceil = 28$.

Randomly generate n ($n \in [21, 32]$) binary plain-images and encrypt them with the same key k. Then, perform KPA attack using these n plaintext/ciphertext pairs. For different values of n, the KPA attack is repeated for 100 times with random known plaintexts/ciphertexts and the occurrence of full recovery (of $[\pi(l)]_{l=1}^{L}$) is counted. The obtained results is depicted in Fig. 5(a) together with the theoretical result given by Eq. (3). From Fig. 5(a), it is clear that the above theoretical analysis correctly characterizes the principle as how $[\pi(l)]_{l=1}^{L}$ is fully identified, while previous analyses [19,20] fail. It is also clear from Fig. 5(a) that $N = 2^n \gg L$ is necessary for full recovery.

Moreover, consider the average number of elements of $[\pi(l)]_{l=1}^{128\times128}$ that are correctly retrieved with some smaller n (fewer number of plaintext/ciphertext pairs). Similarly, repeat each KPA instance for 100 times for all $n \in [10,25]$. This result is depicted in Fig. 5(b) together with the theoretical value given by Eq. (5). As shown by this figure, the overall performance of the experimental result is generally better than that of the theoretic result. Such a phenomenon may be attributed to the fact that, to approximate $\pi(l_0)$ in the case of $\#\bar{\Lambda}(j) > 1$, $\tilde{\pi}(l_0)$ will coincides with $\pi(l_0)$ at a probability of $1/\#\bar{\Lambda}(j)$.

With $n = 14$ and $n = 22$, the obtained approximate sequence $[\tilde{\pi}(l)]_{l=1}^{128\times128}$ is used to decrypt the image shown in Fig. 4(b) and the two decryption results are depicted in Figs. 4(c) and (d), respectively. The contour of the Lena image is still identifiable from Fig. 4(c) even though only 63% (larger than the theoretical value 37%) of the elements of $[\pi(l)]_{l=1}^{128\times128}$ are correctly recovered under the KPA attack, which agrees with the discussion in Section 3.2.

4.2. KPA attacks with known natural plaintexts

So far, based on the assumption that known plaintexts are uniformly distributed, the theoretical analysis on KPA attacks to permutation-only image ciphers has been evaluated. In the following, the proposed KPA attack is implemented using natural images as the source of the known plaintexts. The first row of Fig. 6 depicts four 128×128 gray-scale test images that are used in this experiment and the second row of Fig. 6 depicts their respective encryption results using a fixed key k. Here, $\lceil \log_r L \rceil = \lceil \log_{256} 128^2 \rceil = 2$. The attacker performs KPA attack on the Cipher-image #4 with the first i ($1 \le i \le 3$) plain-images and their corresponding cipher-images and the result is depicted in Fig. 7. For comparison, the KPA result using i random plain-image/cipher-image pairs is also depicted.

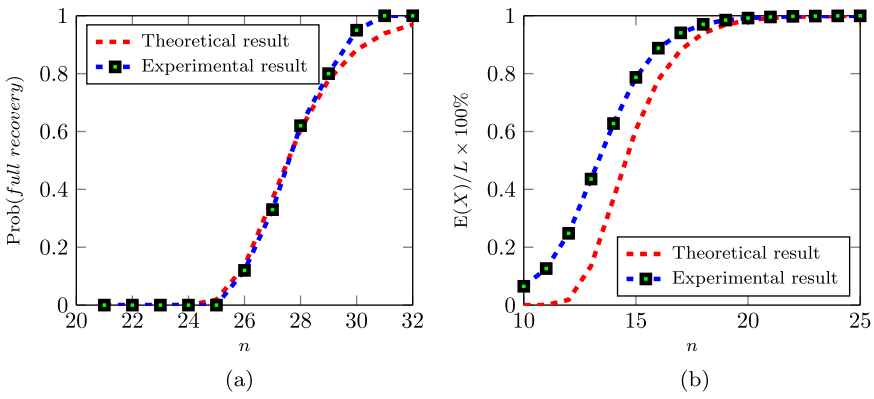


Fig. 5. For $r = 2$ and $L = 128 \times 128$: (a) The value of Prob(*full recovery*) against different n; (b) The value of $E(X)/L \times 100\%$ against different n.

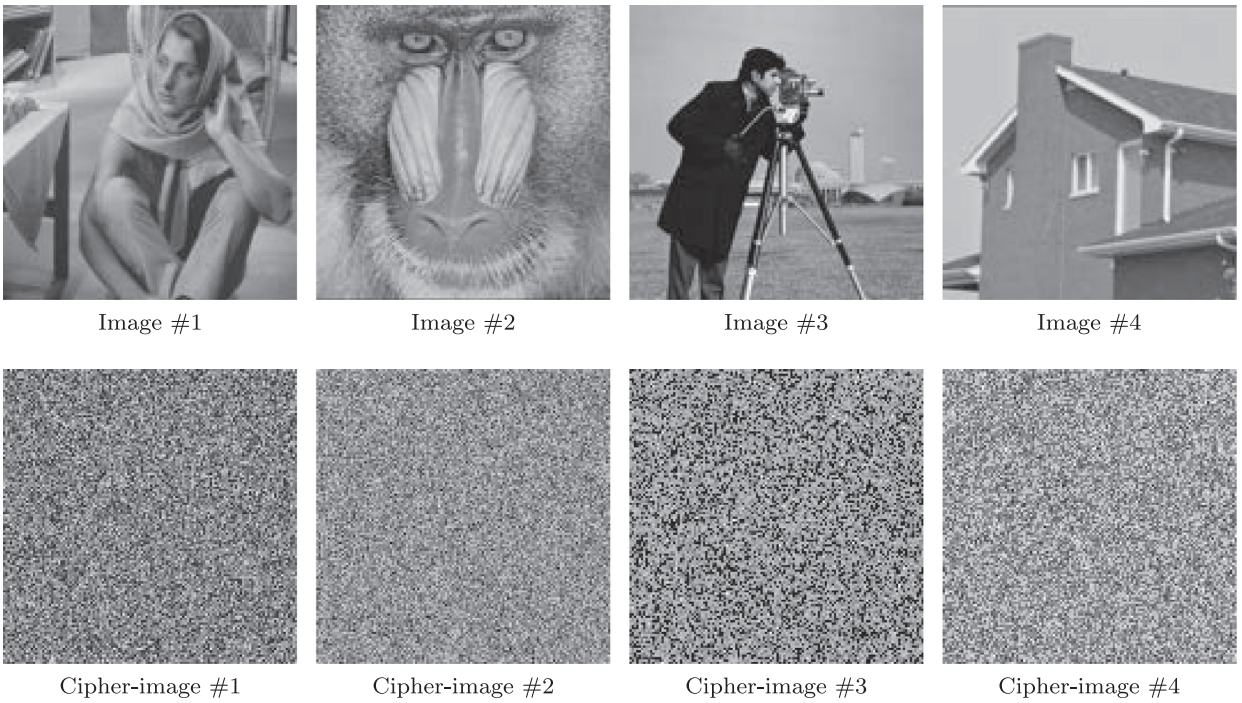


Fig. 6. Four test images and their corresponding cipher-images.

It can be observed from Fig. 7, when 3 plain-image/cipher-image pairs are used, the visual quality of the KPA results obtained from both randomly generated known-images and natural known-images are pretty good. For randomly generated known plain-images, referring to Eq. (5), the percentage of the average number of correctly recovered elements of $[\pi(l)]_{l=1}^{128\times128}$ reaches $E(X)/L \times 100\% = (1 - \frac{1}{256^3})^{128\cdot128-1} \times 100\% = 99.9\%$. For the natural images, according to Proposition 2, the number of correctly recovered pixels is proportional to the number of unique atoms in the composite plaintext constructed by images #1, #2 and #3. Generally, for n natural plain-images, $\mathbf{P}_1$, $\mathbf{P}_2$, $\cdots$, $\mathbf{P}_n$, the quality of the KPA results using these n natural plain-images should be (more or less) as good as its randomized counterpart, if the conditional entropy $H(\mathbf{P}_i|\mathbf{P}_{i-1}, \ldots, \mathbf{P}_1)$ $(i = 2 \sim n)$ is high.

4.3. Efficiency comparison

In this section, rather than estimating the computational complexity, the efficiency of the discussed KPA algorithms (i.e., algorithm in [20], algorithm in [19] and our two proposals in Section 3.3) is evaluated by simulations using C++ 4.2.1

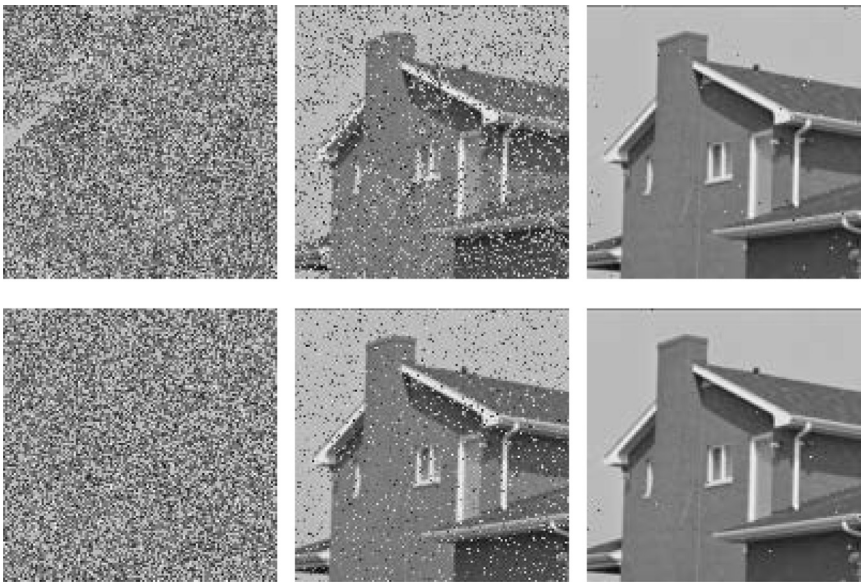


Fig. 7. Recover Cipher-image #4 with the proposed KPA using: the first i ($1 \le i \le 3$) plain-images and their corresponding cipher-images shown in Fig. 6 (the first row); i random plain-image/cipher-image pairs (the second row).

Table 2
Run time (in seconds) comparisons of our proposal and the state-of-the-art works.

Image Size L	Li et. al's algorithm [20]		Li and Lo's algorithm [19]		Improved Li and Lo's Algorithm		Algorithm 2	
	$n = \log_2 L$	$n = 2\log_2 L$	$n = \log_2 L$	$n = 2\log_2 L$	$n = \log_2 L,\ n_0 = \log_2 L/2$	$n = 2\log_2 L,\ n_0 = \log_2 L$	$n = \log_2 L$	$n = 2\log_2 L$
128×128	1938	3097	0.873	1.660	**0.263**	**0.342**	$\mathbf{6.303 \times 10^{-3}}$	$\mathbf{1.260 \times 10^{-3}}$
256×256	\	\	4.145	7.861	**1.085**	**1.500**	$\mathbf{4.387^{-2}}$	$\mathbf{7.139 \times 10^{-2}}$
512×512	\	\	19.2004	35.208	**4.751**	**6.413**	**0.263**	**0.404**
1024×1024	\	\	88.637	163.345	**20.507**	**27.287**	**1.040**	**2.076**

* We do not go through all the case for the algorithm in [20] since it is time consuming.

on a Macbook Pro with 2.6 GHz Intel Core i5 processor and 8 GB DDR3 memory[9]. All the experiments focus on random binary images, which yielded $r = 2$. The sizes of the tested images, i.e., the values of L, vary from 128×128 to 1024×1024. For different values of L, the test was repeated for 10 times and the average consumed time of the four KPA algorithms is shown in Table 2.

It can be observed from this table that the algorithm in [20] typically takes hours to conduct the KPA attack even for images of small sizes, while the algorithm in [19] dramatically reduces the time consumption. With the concept of composite representation, the improved Li and Lo's Algorithm runs $4 \sim 5$ times faster than the method in [19], and Algorithm 2 typically runs 100 times faster than the method in [19]. This phenomenon implies that the big-O notation in Table 1 hides large constants, which agrees with the analysis presented in Section 3.3.

5. Conclusion

This paper has revisited the resistance of permutation-only multimedia encryption algoritms against known-plaintext attack. Although it seems to be a well understood and studied problem, the works in the literature did not answer two basic questions: how many plaintexts are required to probabilistically recover the full secret permutation sequence and how to balance the storage and computation. Based on a simple concept, called composite representation, a necessary and sufficient condition has been established for determining the secret permutation sequence together with an estimation of the required number of plaintexts to probabilistically recover it. Numerous experiments have been performed to confirm the analysis. Moreover, simulations results have demonstrated that one of the accompanied algorithms, although it is not the fastest one, is generally faster than the state-of-the-art schemes by two orders of magnitude when working with images of regular sizes.

[9] The result of Algorithm 1 is not listed here as it is out of memory when L and n are large, for example, $L = 1024 \times 1024$ and $n = 40$. But, it is quite understandable that Algorithm 1 is faster than all the other algorithms.

Acknowledgments

The authors would like to thank the editor and anonymous reviewers for their valuable comments and suggestions to improve the quality of this manuscript. The work was supported in part by the National Natural Science Foundation of China under Grant 61702221, 61502399, 61402547, 61572412, in part by the Macau Science and Technology Development Fund under Grant FDCT/046/2014/A1, FDCT/022/2017/A1, in part by the Research Committee at the University of Macau under Grant MYRG2015-00056-FST, MYRG2016-00137-FST, and in part by the NVIDIA Corporation.

References

[1] M. Bertilsson, E.F. Brickell, I. Ingemarsson, Cryptanalysis of video encryption based on space-filling curves, in: Proc. Advances in Cryptology (EUROCRYPT), Springer, 1989, pp. 403–411.
[2] T. Bianchi, A. Piva, M. Barni, Efficient linear filtering of encrypted signals via composite representation, in: Proc. of the 16th IEEE International Conference on Digital Signal Processing (DSP), 2009, pp. 1–6.
[3] T. Bianchi, A. Piva, M. Barni, Composite signal representation for fast and storage-efficient processing of encrypted signals, IEEE Trans. Inf. Forensics Secur. 5 (1) (2010) 180–187.
[4] T. Bianchi, T. Veugen, A. Piva, M. Barni, Processing in the encrypted domain using a composite signal representation: pros and cons, in: Proc. of the 1st IEEE International Workshop on Information Forensics and Security (WIFS), 2009, pp. 176–180.
[5] N. Bourbakis, A. Dollas, SCAN-Based compression-encryption-hiding for video on demand, IEEE Multimed. 10 (3) (2003) 79–87.
[6] H. Cheng, X. Li, Partial encryption of compressed images and videos, IEEE Trans. Signal Process. 48 (8) (2000) 2439–2451.
[7] J. Daemen, V. Rijmen, The Design of Rijndael: AES-The Advanced Encryption Standard, Springer Science & Business Media, 2013.
[8] A.-V. Diaconu, Circular inter–intra pixels bit-level permutation and chaos-based image encryption, Inf. Sci. 355 (2016) 314–327.
[9] F. Dufaux, T. Ebrahimi, Scrambling for privacy protection in video surveillance systems, IEEE Trans. Circuits Syst. Video Technol. 18 (8) (2008) 1168–1174.
[10] C. Fu, B.-B. Lin, Y.-S. Miao, X. Liu, J.-J. Chen, A novel chaos-based bit-level permutation scheme for digital image encryption, Opt. Commun. 284 (23) (2011) 5415–5423.
[11] R. Graf, W. Sheets, Video Scrambling & Descrambling for Satellite & Cable TV, Newnes, 1998.
[12] H.M. Heys, S.E. Tavares, Avalanche characteristics of substitution-permutation encryption networks, IEEE Trans. Comput. 44 (9) (1995) 1131–1139.
[13] J.E. Hopcroft, J.D. Ullman, Data Structures and Algorithms, Addison-Wesley, 1987.
[14] Z. Hua, Y. Zhou, Image encryption using 2d logistic-adjusted-sine map, Inf. Sci. 339 (2016) 237–253.
[15] F. Huang, J. Huang, Y.Q. Shi, New framework for reversible data hiding in encrypted domain, IEEE Trans. Inf. Forensics Secur. 11 (12) (2016) 2777–2789.
[16] A. Jolfaei, X.-W. Wu, V. Muthukkumarasamy, On the security of permutation-only image encryption schemes, IEEE Trans. Inf. Forensics Secur. 11 (2) (2016) 235–246.
[17] D.E. Knuth, The Art of Computer Programming, Pearson Education, 1998.
[18] C. Li, D. Lin, J. Lü, Cryptanalyzing an image scrambling encryption algorithm of pixel bits, IEEE Multimed. 24 (3) (2017) 64–71.
[19] C. Li, K.-T. Lo, Optimal quantitative cryptanalysis of permutation-only multimedia ciphers against plaintext attacks, Signal Process. 91 (4) (2011) 949–954.
[20] S. Li, C. Li, G. Chen, N.G. Bourbakis, K.-T. Lo, A general quantitative cryptanalysis of permutation-only multimedia ciphers against plaintext attacks, Signal Process. Image Commun. 23 (3) (2008) 212–223.
[21] W. Li, Y. Yan, N. Yu, Breaking row-column shuffle based image cipher, in: Proc. of the 20th ACM international conference on Multimedia (ACMMM), 2012, pp. 1097–1100.
[22] Y. Matias, A. Shamir, A video scrambling technique based on space filling curves, in: Proc. Advances in Cryptology (CRYPTO), Springer, 1987, pp. 398–417.
[23] S.M.M. Rahman, M.A. Hossain, H. Mouftah, A. El Saddik, E. Okamoto, Chaos-cryptography based privacy preservation technique for video surveillance, Multimed. Syst. 18 (2) (2012) 145–155.
[24] V.S. Ramachandran, R.L. Gregory, Perceptual filling in of artificially induced scotomas in human vision, Nature 350 (6320) (1991) 699–702.
[25] H. Sohn, W. De Neve, Y.M. Ro, Privacy protection in video surveillance systems: analysis of subband-adaptive scrambling in JPEG XR, IEEE Trans. Circuits Syst. Video Technol. 21 (2) (2011) 170–177.
[26] T. Stutz, A. Uhl, A survey of H. 264 AVC/SVC encryption, IEEE Trans. Circuits Syst. Video Technol. 22 (3) (2012) 325–339.
[27] C. Wang, H.-B. Yu, M. Zheng, A DCT-based MPEG-2 transparent scrambling algorithm, IEEE Trans. Consum. Electron. 49 (4) (2003) 1208–1213.
[28] Wikipedia, Birthday problem – wikipedia, the free encyclopedia, 2016, (https://en.wikipedia.org/w/index.php?title=Birthday_problem&oldid=738761317). [accessed 14-September-2016].
[29] L.R. Williams, A.R. Hanson, Perceptual completion of occluded surfaces, Comput. Vision Image Understand. 64 (1) (1996) 1–20.
[30] G. Ye, Image scrambling encryption algorithm of pixel bit based on chaos map, Pattern Recognit. Lett. 31 (5) (2010) 347–354.
[31] M. Zanin, A.N. Pisarchik, Gray code permutation algorithm for high-dimensional data encryption, Inf. Sci. 270 (2014) 288–297.
[32] W. Zeng, S. Lei, Efficient frequency domain selective scrambling of digital video, IEEE Trans. Multimed. 5 (1) (2003) 118–129.
[33] L.Y. Zhang, 2017, (https://sites.google.com/site/leoyuzhang/). [accessed 27-June-2017].
[34] L.Y. Zhang, X. Hu, Y. Liu, K.-W. Wong, J. Gan, A chaotic image encryption scheme owning temp-value feedback, Commun. Nonlinear Sci. Numer. Simul. 19 (10) (2014) 3653–3659.
[35] L.Y. Zhang, Y. Liu, F. Pareschi, Y. Zhang, K.-W. Wong, R. Rovatti, G. Setti, On the security of a class of diffusion mechanisms for image encryption, IEEE Trans Cybern (2017), doi:10.1109/TCYB.2017.2682561.
[36] Q. Zhang, L.T. Yang, Z. Chen, Privacy preserving deep computation model on cloud for big data feature learning, IEEE Trans. Comput. 65 (5) (2016) 1351–1362.
[37] Q. Zhang, H. Zhong, L.T. Yang, Z. Chen, F. Bu, PPHOCFS: Privacy preserving high-order cfs algorithm on the cloud for clustering multimedia data, ACM Trans. Multimed. Comput. Commun. Appl. 12 (4s) (2016) 1–16.
[38] X. Zhang, Lossy compression and iterative reconstruction for encrypted image, IEEE Trans. Inf. Forensics Secur. 6 (1) (2011) 53–58.
[39] L. Zhao, A. Adhikari, D. Xiao, K. Sakurai, On the security analysis of an image scrambling encryption of pixel bit and its improved scheme based on self-correlation encryption, Commun. Nonlinear Sci. Numer. Simul. 17 (8) (2012) 3303–3327.
[40] X.-Y. Zhao, G. Cheng, D. Zhang, X.-H. Wang, G.-C. Dong, Decryption of pure-position permutation algorithms, J. Zhejiang University Sci. 5 (7) (2004) 803–809.
[41] J. Zhou, X. Liu, O.C. Au, Y.Y. Tang, Designing an efficient image encryption-then-compression system via prediction error clustering and random permutation, IEEE Trans. Inf. Forensics Secur. 9 (1) (2014) 39–50.
[42] Z.-L. Zhu, W. Zhang, K.-W. Wong, H. Yu, A chaos-based symmetric image encryption scheme using a bit-level permutation, Inf. Sci. 181 (6) (2011) 1171–1186.