On the Multi-Output Filtering Model and Its Applications

Teng Wu, Yin Tan, Kalikinkar Mandal and Guang Gong

Department of Electrical and Computer Engineering
University of Waterloo,
Waterloo, N2L 3G1, CANADA
Email:{teng.wu,y24tan,kmandal,ggong}@uwaterloo.ca

Abstract. In this paper, we propose a novel technique, called multi-output filtering model, to study the non-randomness property of a cryptographic algorithm such as message authentication codes and block ciphers. A multi-output filtering model consists of a linear feedback shift register and a multi-output filtering function. Our contribution in this paper is twofold. First, we propose an attack technique under IND-CPA using the multi-output filtering model. By introducing a distinguishing function, we theoretically determine the success rate of this attack. In particular, we construct a distinguishing function based on the distribution of the linear complexity of component sequences, and apply it on studying TUAK's f_1 algorithm, AES, KASUMI, PRESENT and PRINTcipher. We demonstrate that the success rate of the attack on KASUMI and PRESENT is non-negligible, but f_1 and AES are resistant to this attack. Second, we study the distribution of the cryptographic properties of component functions of a random primitive in the multi-output filtering model. Our experiments show some non-randomness in the distribution of algebraic degree and nonlinearity for KASUMI.

1 Introduction

Let $\mathcal{C}$ be a cryptographic scheme (keyed or non-keyed) with n-bit input and m-bit output. Clearly it can be regarded as a vectorial Boolean function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. When $\mathcal{C}$ involves a key K, we should write C_K for strictness, but we prefer to use $\mathcal{C}$ for simplicity if the context is clear. In most circumstances, the cryptographic properties of $\mathcal{C}$, such as algebraic degree and nonlinearity, are difficult to be exploited due to the large values of n and m. A natural idea to overcome this difficulty is to restrict the inputs of $\mathcal{C}$ on a subspace $\mathcal{S}$ of $\mathbb{F}_2^n$. For instance, the subspace $\mathcal{S}$ can be generated by an ℓ-stage linear feedback shift register. Then we obtain a function $\mathcal{C}'$ from $\mathcal{S}$ to its image set $\mathcal{C}(\mathcal{S})$. By adapting the size of $\mathcal{S}$, we can study the cryptographic properties of $\mathcal{C}'$. If $\mathcal{C}$ has good randomness properties, it should be difficult to find a subspace $\mathcal{S}$ such that $\mathcal{C}'$ has bad randomness properties. We must mention that the above method for analyzing the cryptographic scheme $\mathcal{C}$ lies in a more general notion called *subset cryptanalysis* [27], which tries to track the statistical evolution of a certain subset of values through various operations in the cryptographic schemes. One is referred to [17] for a successful application of the subset cryptanalysis to find a 5-round collision on Keccak [5].

We achieve the above idea by proposing a new technique, called a *multi-output filtering model*. This model aims to exploit the non-randomness property of a cryptographic algorithm $\mathcal{C}$

such as message authentication codes and block ciphers. A multi-output filtering model consists of a linear feedback shift register (LFSR) and a multi-output filtering function. The LFSR is used to generate an input subspace of $\mathcal{C}$ and $\mathcal{C}$ is used as a multi-output filtering function. This multi-output model is a generalization of the classic filtering model in stream ciphers [39] as it outputs multiple bits, instead of only one bit, for the set of inputs to $\mathcal{C}$ generated by an LFSR. Under this model, we can obtain a number of component sequences and component functions in the multi-output model. This paper is devoted to studying the randomness properties of $\mathcal{C}$ through investigating its component sequences and component functions. The detail of the multi-output filtering model can be found in Section 3. We should mention that in this paper we restrict $\mathcal{C}$ to MACs and block ciphers, but this model can also be generalized to study other cryptographic primitives.

Thanks to the fruitful research outcome on the theory of sequences and Boolean functions, we can study the distribution of certain properties of the component sequences and functions. Such properties include linear complexity of the component sequences, algebraic degree and nonlinearity of the component functions, etc. Before describing our contribution in further, let us first briefly introduce the cryptographic primitives on which we apply the multi-output filtering model, especially on the recently proposed f_1 algorithm of the TUAK algorithm set [44] for the 3^{rd} Generation Partnership Project.

Recently, TUAK is proposed to the 3^{rd} Generation Partnership Project (3GPP) for providing authenticity and key derivation functionalities in mobile communications. The design of TUAK is based on the Keccak permutation with 1600-bit internal state due to its good attack resistance property and its simple and efficient constructions of message authentication code and key derivation function. The TUAK algorithm set contains seven different algorithms, namely f_1 to f_5 and f_1^* and f_5^*. The f_1 (or f_1^* as re-synchronisation message authentication) algorithm ensures the authenticity of messages, f_2 is used for generating responses and f_3 to f_5 and f_5^* are used as key derivation functions. Since TUAK's design is closely based on Keccak's design, one may expect that the security property of TUAK may inherit from that of Keccak. The security evaluation for TUAK is essential for guaranteeing the authenticity in mobile communications. The details of TUAK can be found in Appendix A. The analysis of the resistance of TUAK to many known attacks has been presented in [21]. In this paper, we restrict ourselves to the analysis of the MAC generation algorithm f_1 in the multi-output filtering model. Our analysis also considers the block ciphers PRINTcipher [24], AES [11], KASUMI [43], and PRESENT [8].

In Section 5, we introduce a generic distinguishing attack framework on $\mathcal{C}$ under the indistinguishability under chosen-plaintext attack model (IND-CPA for short), which is a variant of indistinguishability of encryptions proposed by Goldwasser and Micali [19] in public-key cryptography settings. This attack makes use of a special object, called a *distinguishing function*. We theoretically determine the success rate of the attack. In particular, we construct a new type of distinguishing function based on the distribution of the linear complexity of the component sequences. Applying this new distinguishing function on f_1, AES, KASUMI and PRESENT, we can distinguish the output of both KASUMI and PRESENT with the output of a random primitive with non-negligible success rate. On the other hand, our study shows that f_1 and AES are immune to this attack.

Furthermore, in Section 7, we study the distribution of the algebraic degree and nonlinearity of the component functions. We first determine the distribution of these two properties for the component functions of a random multi-output filtering function. By performing experiments on f_1, AES, KASUMI and PRESENT, it can be seen that, for KASUMI, the density of its component

functions with algebraic degree less than $\ell - 2$ is greater than the random case, where ℓ is the length of the LFSR. While the degree distributions of the other primitives are similar to that of the random case. This can be a potential risk of the security of KASUMI when an adversary uses the decoding method of Reed-Muller code.

The rest part of the paper is organized as follows. In Section 2 we introduce some background on sequences and Boolean functions. In Section 3, we describe the multi-output filtering model in which a linear feedback shift register is used to generate the inputs of a multi-output function. Section 6 presents some observations on non-randomness about PRINTcipher when it is applied to multi-output filtering model. In Section 4, we describe the attack model of our distinguishing attack and in Section 5, we present the construction of a distinguishing function based on the linear complexity of component sequences. In Section 7, we present some non-randomness in the distribution of the algebraic degree and the nonlinearity of component functions of f_1 and other primitives. Section 8 concludes the paper.

2 Preliminaries

In this section, we provide a list of notations and some definitions that will be used throughout the paper.

Notations

- $\mathbb{F}_2$: the Galois field with two elements $\{0, 1\}$;
- $\mathbb{F}_{2^n}$: a finite field with 2^n elements that is defined by a primitive element α;
- $\mathbb{F}_2^n$: a vector space with 2^n elements and each element is a binary n-tuple;
- $d(f)$: the algebraic degree of a Boolean function f;
- NL(f): the nonlinearity of a Boolean function;
- LC($\mathbf{s}$): the linear complexity of a binary sequence $\mathbf{s}$ with period N;
- Π: Keccak-f[1600] permutation.
- $\mathcal{B}_n$: the set of all Boolean functions with n variables.

2.1 Basic definitions on sequences

We present some definitions on sequences. For a well-rounded treatment of sequences and Boolean functions, the reader is referred to [10, 20].

Let $\mathbf{s} = \{s_i\}$ be a sequence generated by a linear feedback shift register (LFSR) whose recurrence relation is defined as

$$s_{\ell+i} = \sum_{j=0}^{\ell-1} c_j s_{i+j}, s_i, c_i \in \mathbb{F}_2,\ i = 0, 1, ... \tag{1}$$

where $p(x) = \sum_{i=1}^{\ell} c_i x^i \in \mathbb{F}_2[x]$ is the characteristic polynomial of degree ℓ of the LFSR. A binary sequence $\mathbf{s}$ in Eq. (1) with period $2^\ell - 1$ generated by an LFSR is called an *m-sequence*. Let

$\mathbf{s} = \{s_i\}$ be an m-sequence of period $2^\ell - 1$ and $f(x_0, ..., x_{\ell-1})$ be a Boolean function in ℓ variables. We define a sequence $\mathbf{a} = \{a_i\}$ as

$$a_i = f(s_{r_1+i}, s_{r_2+i}, ..., s_{r_t+i}),\ s_i, a_i \in \mathbb{F}_2,\ i \geq 0$$

where $r_1 < r_2 < \ldots < r_t < \ell$ are tap positions. Then the sequence $\mathbf{a}$ is called a *filtering sequence* and the period of $\mathbf{a}$ equals $2^\ell - 1$.

The *linear complexity* or *linear span* of a sequence is defined as the length of the shortest LFSR that generates the sequence. For an m-sequence, the linear complexity of an m-sequence is equal to the length of its LFSR [20]. On the other hand, the linear complexity of a nonlinear filtering sequence lies in the range of ℓ and $2^\ell - 1$ [23]. If a filtering sequence has linear complexity $2^\ell - 1$, then we call it has *optimal* linear complexity.

2.2 Basic definitions on Boolean functions

There is a one-to-one correspondence between a sequence and a Boolean function. The correspondence between a Boolean function and a sequence can be obtained by computing the trace representation of a given sequence using the Fourier transformations. For the details, see Chapter 6 of [20].

Definition 1. *Let f be a Boolean function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. Then f can be uniquely represented by its algebraic normal form (ANF) as*

$$f(x) = \sum_{I \in \mathcal{P}(\{0,...,n-1\})} a_I x^I,$$

where $a_I \in \mathbb{F}_2, x^I = \prod_{i \in I} x_i$ and $\mathcal{P}(\{0, \ldots, n-1\})$ is the power set of $\{0, \ldots, n-1\}$. The algebraic degree of f, denoted by $d(f)$, is the maximal size of I in the ANF of f such that $a_I \neq 0$.

One of the most important properties of Boolean functions is its nonlinearity, which was proposed to measure the distance of it to all affine functions. A cryptographic strong Boolean function is supposed to have high nonlinearity to resist linear attacks [31].

Definition 2. *The Walsh transform of a Boolean function f to a point $a \in \mathbb{F}_2^n$, denoted by $W_f(a)$, is defined by*

$$W_f(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+a \cdot x}$$

where $a \cdot x$ is the inner product of a and x.

The *nonlinearity* of f can be defined in terms of the Walsh transform as

$$\mathrm{NL}(f) = 2^{n-1} - \max_{a \in \mathbb{F}_2^n} \frac{|W_f(a)|}{2}.$$

When n is an even positive integer, it is known that the maximum value of the nonlinearity of a Boolean function f is $\mathrm{NL}(f) \geq 2^{n-1} - 2^{n/2-1}$ [10]. A Boolean functions achieving this bound is called a *bent function*.

Let m and n be two positive integers. A function F, from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$, defined by $F(x) = (f_1(x), f_2(x), ..., f_m(x))$ is called a (n, m)-function, multi-output Boolean functions, or vectorial Boolean functions, where f_i's are called coordinate functions [10].

3 Multi-Output Filtering Model

In this section, we provide a detailed description of the multi-output filtering model of a cryptographic primitive.

3.1 Description of the multi-output filtering model

Let $\mathbf{a} = \{a_i\}_{i\geq 0}$ be a binary sequence generated by an ℓ-stage linear feedback shift register (LFSR) whose recurrence relation is

$$a_{\ell+i} = \sum_{j=0}^{\ell-1} c_j a_{i+j},\ c_j \in \mathbb{F}_2,\ i \geq 0, \tag{2}$$

where $p(x) = x^\ell + \sum_{i=0}^{\ell-1} c_i x^i$ is a primitive polynomial of degree ℓ over $\mathbb{F}_2$ and $\mathrm{STATE}_j = (a_j, a_{j+1}, ..., a_{\ell-1+j})$ is called the j-th state of the LFSR. Using this LFSR, from the above sequence $\mathbf{a}$, we generate a set of messages of n bits as follows $\mathcal{R} = \{\mathrm{R}_j : 0 \leq j \leq 2^\ell - 2\}$ where

$$\mathrm{R}_j = (a_j, a_{j+1}, \cdots, a_{j+n-1}),\ j = 0, 1, ..., 2^\ell - 2, \tag{3}$$

where modulo $2^\ell - 1$ is taken over the indices of a_i's. Note that the elements in $\mathcal{R}$ are in the sequential order. We now define the multi-output filtering model on $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$. For a fixed key K and for each R_j with $0 \leq j \leq 2^\ell - 2$, we obtain

$$\begin{array}{rcl} \mathrm{C}_j & = & F(\mathrm{K}, \mathrm{R}_j) = (g_0(\mathrm{K}, \mathrm{R}_j), \ldots, g_{m-1}(\mathrm{K}, \mathrm{R}_j)) \\ & \triangleq & (y_{j,0}, y_{j,1}, \ldots, y_{j,m-1}). \end{array} \tag{4}$$

Using a matrix, we can represent the above C_j as

$$\begin{pmatrix} \mathrm{C}_0 \\ \mathrm{C}_1 \\ \vdots \\ \mathrm{C}_{2^\ell-2} \end{pmatrix} = \begin{pmatrix} y_{0,0} & y_{0,1} & \cdots & y_{0,m-1} \\ y_{1,0} & y_{1,1} & \cdots & y_{1,m-1} \\ \vdots & \vdots & & \vdots \\ y_{2^\ell-2,0} & y_{2^\ell-2,1} & \cdots & y_{2^\ell-2,m-1} \end{pmatrix}. \tag{5}$$

The matrix (5) provides us two methods to study cryptographic properties of F as described below.

I. **Sequence point of view:** Each column in the above can be considered as a sequence of period $2^\ell - 1$ for a nonzero initial state of the LFSR. Each sequence of period $2^\ell - 1$ is called a *component sequence*. We denote the i-th component sequence by $\mathbf{s}_i$ and $\mathbf{s}_i = \{y_{0,i}, y_{1,i}, ..., y_{2^\ell-2,i}\}$. $\mathbf{s}_i$ can also be considered as a filtering sequence with filter function g_i, $0 \leq i \leq m-1$.

II. **Boolean function point of view:** From (4) and (5), we see the following process

$$g_i : \left\{\mathrm{STATE}_j \in \mathbb{F}_2^\ell \text{ of the LFSR}\right\} \to \{\mathrm{R}_j \in \mathbb{F}_2^n\} \to i\text{-th component sequence.}$$

Therefore, each component sequence can also be regarded as a Boolean function on $\mathbb{F}_2^\ell$. Note that, for a nonzero initial state, the LFSR cannot generate all-zero state, we need

to query F to get the output value $F(\mathrm{K}, 0^n)$ for all-zero input for all component Boolean functions. With a fixed K in F, using an ℓ-stage LFSR, we obtain m Boolean functions on $\mathbb{F}_2^\ell$. Mathematically, m Boolean functions $g_i : \mathbb{F}_2^\ell \to \mathbb{F}_2$ $(0 \le i \le m-1)$ are defined as

$$g_i(K, \mathrm{STATE}_j) = y_{j,i}, \quad (0 \le j \le 2^\ell - 2). \tag{6}$$

We call each Boolean function g_i a *component* or *coordinate function* of F.

3.2 Application to TUAK's f_1, AES, KASUMI and PRESENT

For the sake of clarity on the input assignment, we briefly explain how we apply the multi-output filtering model on TUAK's f_1, and block ciphers AES, PRESENT and KASUMI.

3.2.1 TUAK's f_1:

Recall that f_1 takes K, RAND, and SQN as inputs. We fix the key K and the sequence number SQN. We use an ℓ-stage LFSR to generate random numbers RAND_j in f_1. Denoting by the i-th state of the ℓ-stage LFSR by $\mathrm{STATE}_i \in \mathbb{F}_2^\ell$. We obtain $2^\ell - 1$ different n-bit RAND numbers $\mathcal{R} = \{\mathrm{R}_j : 0 \le j \le 2^\ell - 2\}$ by Eq. (3) and the component sequences and component functions are obtained using Eq. (4) with $\mathrm{C}_j = f_1(\mathrm{K}, \mathrm{R}_j, \mathrm{SQN})$.

Remark 1. *For TUAK's f_1 function, in Eq. (5), recovering the last bit $y_{2^\ell-2,i}$ for each component sequence $\boldsymbol{s}_i$ from the previous $2^\ell-2$ bits is equivalent to recovering $C_{2^\ell-2}$ from $\{C_0, ..., C_{2^\ell-3}\}$. This leads to a MAC forgery attack on f_1.*

3.2.2 PRINTcipher, AES, PRESENT and KASUMI:

Recall that PRINTcipher is a lightweight cipher published on CHES 2010. The key of PRINTcipher has 80 bits, which include a 48-bit XOR key and a 32-bit permute key. The block size of PRINTcipher is 48-bit. AES_128 accepts a 128-bit key and a 128-bit input and produces an output of 128 bits, and AES_256 accepts a 256-bit key and a 128-bit input and produces an output of 128 bits [11]. KASUMI has a 64-bit input, a 128-bit key, and a 64-bit output. For AES_128 and AES_256, the inputs messages of 128 bits are generated using an LFSR of length ℓ and by Eq. (3), and the component sequences and functions are obtained using Eq. (4) with $\mathrm{C}_j = \mathsf{AES\_128}(\mathrm{K}, \mathrm{R}_j)$ and $\mathrm{C}_j = \mathsf{AES\_256}(\mathrm{K}, \mathrm{R}_j)$. PRESENT [8] is a 64-bit block cipher with a 80-bit key. The component sequences and functions of PRESENT are obtained using Eq. (4) with $\mathrm{C}_j = \mathsf{PRESENT}(\mathrm{K}, \mathrm{R}_j)$. KASUMI [43] is a 64-bit block cipher with a 128-bit key. The 64-bit inputs messages are generated by Eq. (3) with $n = 64$ and the component sequences and functions are obtained using Eq. (4) with $\mathrm{C}_j = \mathsf{KASUMI}(\mathrm{K}, \mathrm{R}_j)$.

4 Distinguishing Attack Model

In this section, we describe the attack model of our distinguishing attack on a message authentication code and a block cipher. The attack model is based on indistinguishability (IND) of encryptions under chosen-plaintext attack (CPA) (IND-CPA), which was first proposed by Goldwasser and Micali [19] in public-key settings. In [3], Bellare *et al.* studied the indistinguishability of encryptions under chosen-plaintext attack in the symmetric key setting. Here,

we use the same attack model to distinguish MACs (or ciphertexts) in the symmetric-key setting. However, we develop a new distinguishing technique based on linear complexity of component sequences in the multi-output filtering model for deciding the MAC (or ciphertext). For the message authentication code, the aim of an adversary is to distinguish two MACs for two messages P_0 and P_1 with a high probability where messages P_0 and P_1 were chosen by the adversary. On the other hand, for an encryption, the adversary aims at distinguishing two ciphertexts for two chosen messages P_0 and P_1 with a high probability.

Let $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$ be a cryptographic algorithm which accepts two inputs, a key of length k and a message of length n and produces an output of length m. Assume that P_0 and P_1 are two messages of length n chosen by the adversary, the length of the key K is k and $c_i = F(K, P_i), i = 0, 1$. The aim of the distinguishing attack is to distinguish c_0 and c_1 for the messages P_0 and P_1 with a high probability. We denote the random oracle by $\mathcal{O}$ and the adversary by $\mathcal{A}$. The indistinguishability game [2, 19] between the random oracle and the adversary is played as follows.

(1) Fixing a key K and generating the set of messages $\mathcal{R} = \{R_0, R_1, ..., R_{N-1}\}$ using an LFSR with a primitive polynomial of degree ℓ, $N = 2^\ell - 1$;

(2) The adversary $\mathcal{A}$ randomly picks up $P_0 \in \mathcal{R}$ and $P_1 \notin \mathcal{R}$ and sends both $\{P_0, P_1\}$ to $\mathcal{O}$.

(3) The random oracle picks up $P_b \xleftarrow{\$} \{P_0, P_1\}$, $b = 0$ or 1 and computes $c = F(K, P_b)$. $\mathcal{O}$ sends c to the adversary $\mathcal{A}$.

(4) Once $\mathcal{A}$ receives c as a challenge, the adversary performs a technique and decides b' and returns b' to $\mathcal{O}$ where $b' = 0$ or 1;

(5) If $b = b'$, then adversary $\mathcal{A}$ succeeds; otherwise she fails.

We also summarize the game in Figure 1.

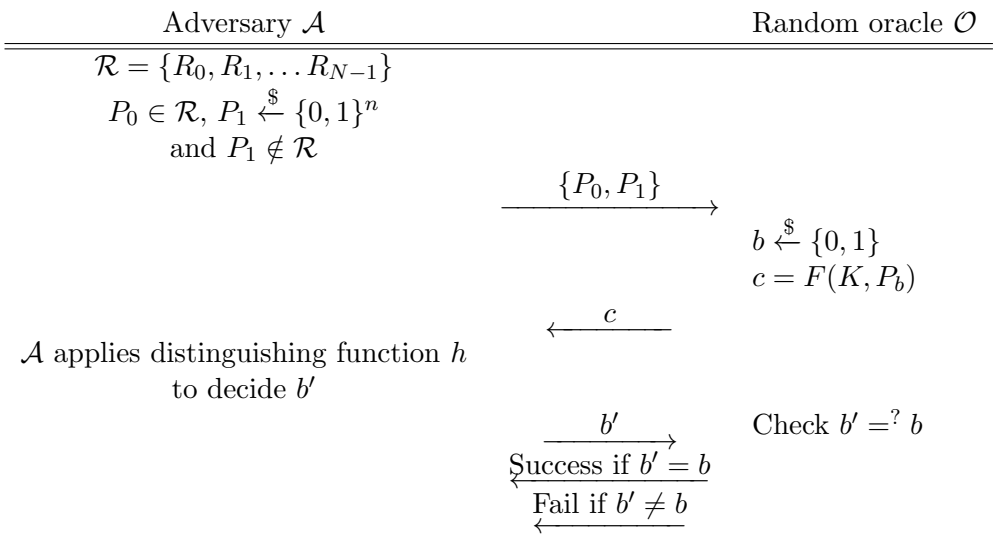

Figure 1: Indistinguishability game

It is easy to see that, for a random cipher $\mathcal{B}$, the success rate of winning the game for an adversary is 1/2. In the following section, we present a new method to distinguish the MACs

produced by f_1 for P_0 and P_1 with probability greater than $1/2$. Therefore, the new method provides a construction of a distinguisher on f_1.

5 Distinguishing Attack Based on Linear Complexity

In this section, we first present a general technique to build a distinguisher of a cryptographic primitive, followed by the theoretical determination of the success probability of the distinguishing attack. In particular, we make use of the distribution of the linear complexity of component sequences of a primitive to develop a new distinguisher. Finally we apply this technique on f_1, AES, KASUMI, and PRESENT.

5.1 A generic framework to build a distinguisher

We start this section by the following definition.

Definition 3. *Let $\mathcal{R}$ and $\mathcal{S}$ be two subsets of U, where $\mathcal{S} = U \setminus \mathcal{R}$. Let Ω be a subset of $\mathcal{R} \times \mathcal{S}$. Let $\mathcal{C}$ be a cryptographic scheme from U to some set V. For any $P_0 \in \mathcal{R}$ and $P_1 \in \mathcal{S}$, define a distinguishing function $h : \{\mathcal{C}(P_0), \mathcal{C}(P_1)\} \to \{0, 1\}$. We say that $\mathcal{C}$ is distinguishable with respect to $\mathcal{R}, \mathcal{S}, h, \Omega$ if the average probability*

$$\sum_{i \in \{0,1\}} \Pr\Big(h(c) = i \wedge c = \mathcal{C}(P_i)\Big)$$

is non-negligible compared with $1/2$, when (P_0, P_1) is randomly chosen from Ω.

Now we state the main theorem below.

Theorem 1. *Let the notations be the same as above. Now we define a subset $\mathcal{CS}$ of U, which is called the condition set. Let $\mathcal{S}' \subset \mathcal{S}$ and $\Omega = \mathcal{R} \times \mathcal{S}'$. For any $P_0 \in \mathcal{R}, P_1 \in \mathcal{S}'$, let us define the distinguishing function $h : \{\mathcal{C}(P_0), \mathcal{C}(P_1)\} \to \{0, 1\}$ as*

$$h(y) = \begin{cases} 0 & \text{if } y = \mathcal{C}(x) \text{ and } x \in \mathcal{CS}, \\ 1 & \text{otherwise.} \end{cases} \tag{7}$$

Define the following two probabilities

$$\begin{array}{rcl} q_0 & = & \Pr\left(x_0 \in \mathcal{R} \wedge x_0 \in \mathcal{CS}\right), \\ q_1 & = & \Pr\left(x_1 \in \mathcal{S}' \wedge x_1 \in \mathcal{CS}\right) \end{array} \tag{8}$$

where $(x_0, x_1) \xleftarrow{\$} \Omega$. Then the average probability is

$$\sum_{i \in \{0,1\}} \Pr\left(h(c) = i \wedge c = \mathcal{C}(P_i)\ \right) = \frac{1 + (q_0 - q_1)}{2}. \tag{9}$$

Proof. It is not difficult to see that there are four independent cases of the event $h(c) = i \wedge c = \mathcal{C}(P_i)$ when $i \in \{0, 1\}$, therefore we may compute its probability one by one and sum them together:

(1) $h(c) = 0 \wedge c = \mathcal{C}(P_0) \wedge P_0 \in \mathcal{CS}$. The probability of this case equals

$$\Pr\left(h(c) = 0 \mid c = \mathcal{C}(P_0) \wedge P_0 \in \mathcal{CS}\right) \Pr\left(c = \mathcal{C}(P_0) \mid P_0 \in P\right) \Pr\left(P_0 \in \mathcal{CS}\right) = \frac{1}{2} q_0;$$

(2) $h(c) = 0 \wedge c = \mathcal{C}(P_0) \wedge P_0 \notin \mathcal{CS}$. The probability of this case is clear 0.

(3) $h(c) = 1 \wedge c = \mathcal{C}(P_1) \wedge P_0 \in \mathcal{CS}$. The probability of this case equals

$$\Pr\left(h(c) = 1 \mid c = \mathcal{C}(P_1) \wedge P_0 \in \mathcal{CS}\right) \Pr\left(c = \mathcal{C}(P_1) \mid P_0 \in P\right) \Pr\left(P_0 \in \mathcal{CS}\right) = \frac{1}{2} q_0 (1 - q_1).$$

(4) $h(c) = 1 \wedge c = \mathcal{C}(P_1) \wedge P_0 \notin \mathcal{CS}$. The probability of this case equals

$$\Pr\left(h(c) = 1 \mid c = \mathcal{C}(P_1) \wedge P_0 \notin \mathcal{CS}\right) \Pr\left(c = \mathcal{C}(P_1) \mid P_0 \notin P\right) \Pr\left(P_0 \notin \mathcal{CS}\right) = \frac{1}{2} (1 - q_0)(1 - q_1).$$

Summing the above probability we have the desired result

$$\sum_{i \in \{0,1\}} \Pr\left(h(c) = i \wedge c = \mathcal{C}(P_i)\right) = \frac{1 + (q_0 - q_1)}{2}.$$

The proof is completed. □

Several remarks on Theorem 1 are as follows:

(i) An attacker will expect the probability value in (9) to be as large as possible so that she can distinguish the cryptographic scheme $\mathcal{C}$ with a high probability.

(ii) The difficulty of finding the distinguishing attack described in Theorem 1 is to find a proper condition set $\mathcal{CS}$ such that $q_0 - q_1$ is large.

(iii) The value of $q_0 - q_1$ could be negative. If the attacker uses $\overline{\mathcal{CS}}$ to replace $\mathcal{CS}$, $q_0 - q_1$ will be positive, and the probability will be greater than 0.5. Thus, the problem of finding a condition set such that $q_0 - q_1$ is large becomes the problem of finding the condition set such that $|q_0 - q_1|$ is large.

(iv) In the rest of this section, we will show how to construct such set $\mathcal{CS}$, which leads to the distinguishing attacks on KASUMI and PRESENT with a non-negligible success rate.

5.2 Distribution of the linear complexity of component sequences

We use f_1, AES, KASUMI and PRESENT as multi-output filtering functions and study the distribution of the linear complexity of their component sequences. Meidl and Niederreiter studied the expectation of the linear complexity of random binary periodic sequences in [32]. According to our experimental results, the average values of the linear complexities of the component sequences of AES, f_1, KASUMI, PRESENT are very close to the theoretical value determined in [32]. This motivates us to look at the whole distribution of the linear complexity of the component sequences instead of considering only the average value. We perform the following test for the linear complexity and have an interesting observation on the component sequences of KASUMI and PRESENT.

5.2.1 Test of the distribution of linear complexity

Usually, for a primitive $\mathcal{C}$, it is difficult to determine the distribution of linear complexity of its component sequences. Of course, one can choose a subset of inputs to the primitive to estimate the linear complexity distribution. However, since the input space is very large, it is hard to measure the accuracy of the estimated distribution. To avoid such problem, we propose a new method to test the distribution. This goal is achieved by choosing two (large) subsets of inputs and by comparing the distributions of the linear complexity of their component sequences. In particular, we choose one subset $\mathcal{LI}$ of the inputs generated by an ℓ-stage LFSR and the other subset $\mathcal{RI} = (\mathcal{LI} \setminus \{P_0\}) \cup \{P_1\}$, where $P_0 \xleftarrow{\$} \mathcal{LI}$ and $P_1 \xleftarrow{\$} \overline{\mathcal{LI}}$. Note that the elements in $\mathcal{LI}$ are ordered according to Eq. (3). It is clear that if the $\mathcal{C}$ has very good random property, it should not be easy to distinguish two distributions for $\mathcal{LI}$ and $\mathcal{RI}$. Our method consists of the following three steps.

Now fixing a primitive $\mathcal{C}$ and an ℓ-stage LFSR:

Step 1 (Generating component sequences)

We randomly choose N_{key} keys.

1. For all keys, using $\mathcal{LI}$ as the set of inputs and $\mathcal{C}$ as a multi-output filter, we obtain $m \cdot N_{key}$ component sequences. This set of component sequences is denoted by Q_1.
2. Similarly, using $\mathcal{RI}$ as the inputs, we generate another set of $m \cdot N_{key}$ component sequences, which is denoted by Q_2.

Step 2 (Computing linear complexity)

We compute the linear complexities of the sequences in Q_1 and Q_2 and count the number of component sequences in Q_i with the linear complexity $2^\ell - 2$ and $2^\ell - 1$, denoted by $N^i_{2^\ell-1}$ and $N^i_{2^\ell-2}$, where $i = 1$ or 2.

Step 3 (Comparing the distributions)

Now we compare two distributions by computing the slopes sl_i of the line between two points $(2^\ell - 2, N^i_{2^\ell-2})$ and $(2^\ell - 1, N^i_{2^\ell-1})$, where

$$sl_i = \frac{N^i_{2^\ell-1} - N^i_{2^\ell-2}}{(2^\ell - 1) - (2^\ell - 2)} = N^i_{2^\ell-1} - N^i_{2^\ell-2}.$$

If the difference between sl_1 and sl_2 is non-negligible, we can make use of it to build a distinguisher of $\mathcal{C}$, which is described in the next section. The worst case computational complexity for exhausting all ℓ-stage LFSRs of the above three steps is

$$\frac{\phi(2^\ell - 1)}{\ell} \times N_{\text{key}} \times 2\ell \times (2^\ell - 1) \times m, \tag{10}$$

where ϕ is the Euler phi function. Below we perform the experiment using these parameters on f_1, AES, KASUMI and PRESENT.

5.2.2 Distribution of f_1, AES, KASUMI and PRESENT

In our experiment, we choose $\ell = 8$ and $N_{key} = 10^8$. By Eq. (10), the worst case complexity for the primitive f_1 is $2^{50.27}$ (some computation can be performed in a parallel way). We present the result in the following figures in which the solid (resp. dashed) line represents the distribution of sequences in Q_1 (resp. Q_2).

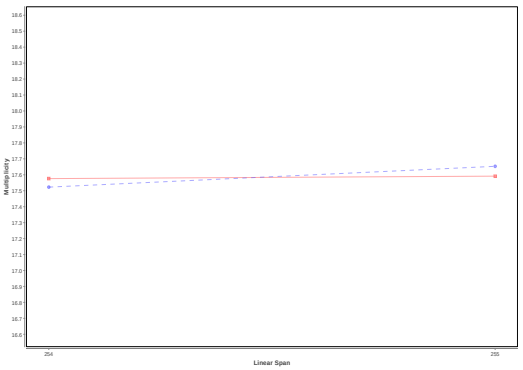

Figure 2: KASUMI

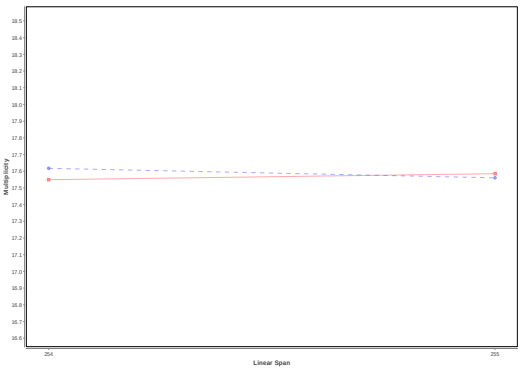

Figure 3: PRESENT

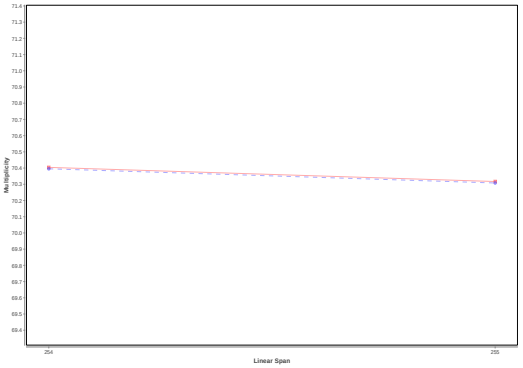

Figure 4: AES

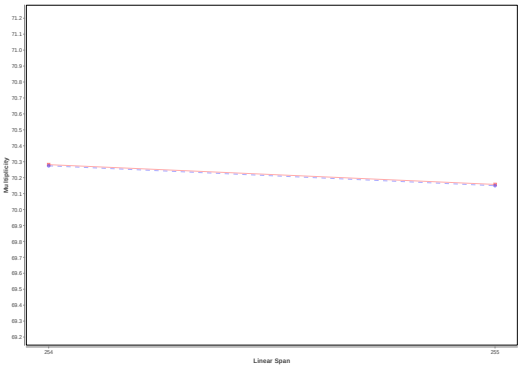

Figure 5: f_1

From Figs. 2 and 3, one can observe that, for KASUMI and PRESENT, the difference of the distribution of the linear complexities for sequences in Q_1 and Q_2 is non-negligible. While Figs. 4 and 5 show this is not the case for AES and f_1.

5.3 The new distinguishing attack

We now present the details of our distinguishing attack, which is achieved through constructing a distinguishing function h. The construction of the distinguishing function is based on the linear complexity distribution of the component sequences of a primitive in the multi-output filtering model.

5.3.1 Constructing the distinguishing function.

Recall that the distinguishing function is defined in Definition 3. We use the notations in Theorem 1 and the attack model is depicted in Fig. 1.

Table 1: Average success rate of our attack on f_1, AES, KASUMI and PRESENT

Primitive	t	q_0	q_1	Avg. Succ. Rate
f_1	2	0.20398	0.194458	50.476%
AES	2	0.193848	0.20044	50.329%
KASUMI	4	0.421875	0.454103	51.612%
PRESENT	5	0.5686	0.540285	51.416%

1. Choosing an ℓ-stage LFSR with a primitive polynomial to generate the inputs of length n in $\mathcal{R}$ (see Eq. (3)). For f_1 and AES, $n = 128$; for KASUMI and PRESENT, $n = 64$.
2. Constructing $\mathcal{S} = \mathbb{F}_2^n \setminus \mathcal{R}$;
3. Randomly choosing a message $P_0 \in \mathcal{R}$ and $P_1 \in \mathcal{S}$;
4. Let N_{LC} be the number of component sequences with linear complexity LC where $\ell \leq LC \leq 2^\ell - 1$;
5. Defining the condition set

$$\mathcal{CS} = \left\{ y \in \mathbb{F}_2^n \;\middle|\; \begin{array}{l} \text{using } (\mathcal{R} \setminus \{P_0\}) \cup \{y\} \text{ as the inputs of a primitive in the} \\ \text{multi-output filtering model, the slope of the line between} \\ \text{the points } (2^\ell - 2, N_{2^\ell - 2}) \text{ and } (2^\ell - 1, N_{2^\ell - 1}) \text{ is less than } t. \end{array} \right\};$$

6. The distinguishing function h is defined in Eq. (7) using the condition set $\mathcal{CS}$;
7. q_0, q_1 are the probability values defined in Definition 3.

5.4 An example of the attack

In this section, we apply the attack with our distinguishing function defined in Section 5.3 on f_1, AES, KASUMI, and PRESENT. Theorem 1 and the observations in Figs. 2 and 3 enable us to gain a non-negligible success rate of the attack on KASUMI and PRESENT. For simplicity, we use an 8-stage LFSR to conduct our attack. However, one can use an arbitrary length LFSR.

The set $\mathcal{R}$ is constructed by the 8-stage LFSR with a primitive polynomial. We then randomly chose 2^{10} keys. For each key, a message $P_0 \in \mathcal{R}$ and a message $P_1 \in \mathcal{S}$ are chosen randomly. We use the distinguishing function h to execute the attack. It is worth to mention that, to test that the average success rate is stable, we repeated the experiment 20 times by choosing different groups of 2^{10} keys and we found similar results for all experiments. We present the average success rate for an experiment in Table 1, where we use the upper bound of the slope t and the 8-stage LFSR the same as those in Table 7.

One can observe from the average success rate in Table 1 that the outputs of both KASUMI and PRESENT can be distinguished from a random primitive with a non-negligible probability. On the other hand, the performance of f_1 and AES is very similar to the random one.

6 Distribution of the Linear Complexity of PRINTcipher under the Multi-Output Filtering Model

In Leander *et al.*'s paper [28], the authors pointed out a weakness of PRINTcipher that when the input and key have some particular patterns the output has the same pattern as the input.

Since the input and output have the same pattern, the subspace of $\mathbb{F}_2^{48}$ formed by the inputs and outputs is called *invariant subspace*. Table 2 shows one example of such patterns and the invariant subspace. Each asterisk in Table 2 represents one variant bit, which can be either 0 or 1. The bits that are fixed to 0 or 1 are called invariant bits. We use $\mathcal{I}$ to denote the invariant subspace shown in Table 2, and $\mathcal{K}$ to denote the key (composed by XOR and permute key) space shown in Table 2. The *invariant subspace attack* is addressed as follows. If the input and key are selected from $\mathcal{I}$ and $\mathcal{K}$ respectively, the output must be in $\mathcal{I}$.

Table 2: Patterns of input, key, and output

Input	00*	*10	***	***	00*	*10	***	***	00*	*10	***	***	00*	*10	***	***
XOR Key	01*	*01	***	***	01*	*01	***	***	01*	*01	***	***	01*	*01	***	***
Permute Key	0*	11	**	**	10	01	**	**	11	*0	**	**	*0	11	**	**
Output	00*	*10	***	***	00*	*10	***	***	00*	*10	***	***	00*	*10	***	***

6.1 Directly Applying Multi-Output Filtering Model to PRINTcipher

One immediate application of our multi-output filtering model is applying it model to PRINTcipher with the weak keys in $\mathcal{K}$. We choose an 8-stage LFSR with primitive feedback polynomial, and initialize the LFSR with 0x01. Let $v_0, \cdots, v_7$ denote the first 8 varian bits of the input, and LFSR_i denote the i-th bit of the LFSR's internal state. The experiment is done as follows.

(1) Fix all invariant bits of the input to the input pattern shown in Table 2.

(2) Randomly generate the variant bits except for the first 8 variant bits.

(3) Run the LFSR once.

(4) Assign the internal state of the LFSR to the first 8 variant bits of the input, i.e. $v_i = \text{LFSR}_i$, for $0 \leq i < 8$.

(5) Randomly generate a key, with the key pattern shown in Table 2.

(6) Compute the output of the input, and put each bit of the output into the corresponding component sequence.

(7) Repeat from Step (3) 254 times.

(8) Compute linear complexity for each component sequence.

By repeating this procedure N times, we can get the linear complexity distribution of each component sequence. Obviously, the component sequences that are composed by the invariant bits are either all 0 sequences or all 1 sequences. Thus, the linear complexity of such component sequence is either 0 or 1. Ideally, the linear complexity of each component sequence is distributed as partially shown in Table 3. For each component sequence composed by invariant bit of the output, the linear complexity is a constant. Thus, the distribution is far away from the ideal one. Such difference shows the non randomness of PRINTcipher.

6.2 Applying Multi-Output Filtering Model to PRINTcipher with Recurrent Input

A more challenging experiment is letting one invariant bit change. Denote the i-th bit of the input by b_i. The experiment is done as follows.

Table 3: Ideal linear complexity distribution of sequences with period 255

Linear Complexity	Distribution
255	0.2747587618182607
254	0.2747587618182607
...	...
1	$1.7272337110188872 \times 10^{-77}$
0	$1.7272337110188872 \times 10^{-77}$

(1) Fix all invariant bits of the input to the input pattern shown in Table 2 except for b_5.

(2) Randomly generate the variant bits except for $b_2, b_3, b_6, b_7, b_8,$ and b_9.

(3) Run the LFSR once.

(4) Assign the internal state as follows.

$$\begin{array}{lll} b_2 = \mathrm{LFSR}_0, & b_3 = \mathrm{LFSR}_1, & b_5 = \mathrm{LFSR}_3, \\ b_6 = \mathrm{LFSR}_4, & b_7 = \mathrm{LFSR}_5, & b_8 = \mathrm{LFSR}_6, \\ b_9 = \mathrm{LFSR}_7 & & \end{array}$$

Note that since b_4 is fixed to 1, in one period of the LFSR, each possible input occurs twice. We call this *recurrent input*.

(5) Randomly generate a key, with the key pattern shown in Table 2.

(6) Compute the output of the input, and put each bit of the output into the corresponding component sequence.

(7) Repeat from Step (3) 254 times.

(8) Compute linear complexity for each component sequence.

After running the above procedure for N times, we have the linear complexity distribution of each component sequence. Since the ideal distribution of each component sequence with recurrent input is hard to compute, we run the above procedure on both PRINTcipher and AES, and use the distribution of AES as a reference. The results are shown in Figure 6. The blue curve marked with cross represents the distribution of AES' 74th component sequence; the red curve marked with horizontal bars represents PRINTcipher's 12th component sequence, and the brown curve marked with vertical bars represents the 20th component sequence. For all AES's component sequences, the distributions of linear complexity are almost the same. However, we find the distribution of the 12th component sequence of PRINTcipher is quite different from AES. We can see that the red curve jumps dramatically

This interesting observation motivates us to apply our model to other primitives to test the randomness of those primitives.

7 Distribution of the Algebraic Degree and Nonlinearity of the Component Functions

In this section, we investigate the distribution of the algebraic degree and the nonlinearity of the component functions of f_1, AES, KASUMI, and PRESENT in the multi-output filtering model. To measure the randomness property, we first determine the distribution of the algebraic degree

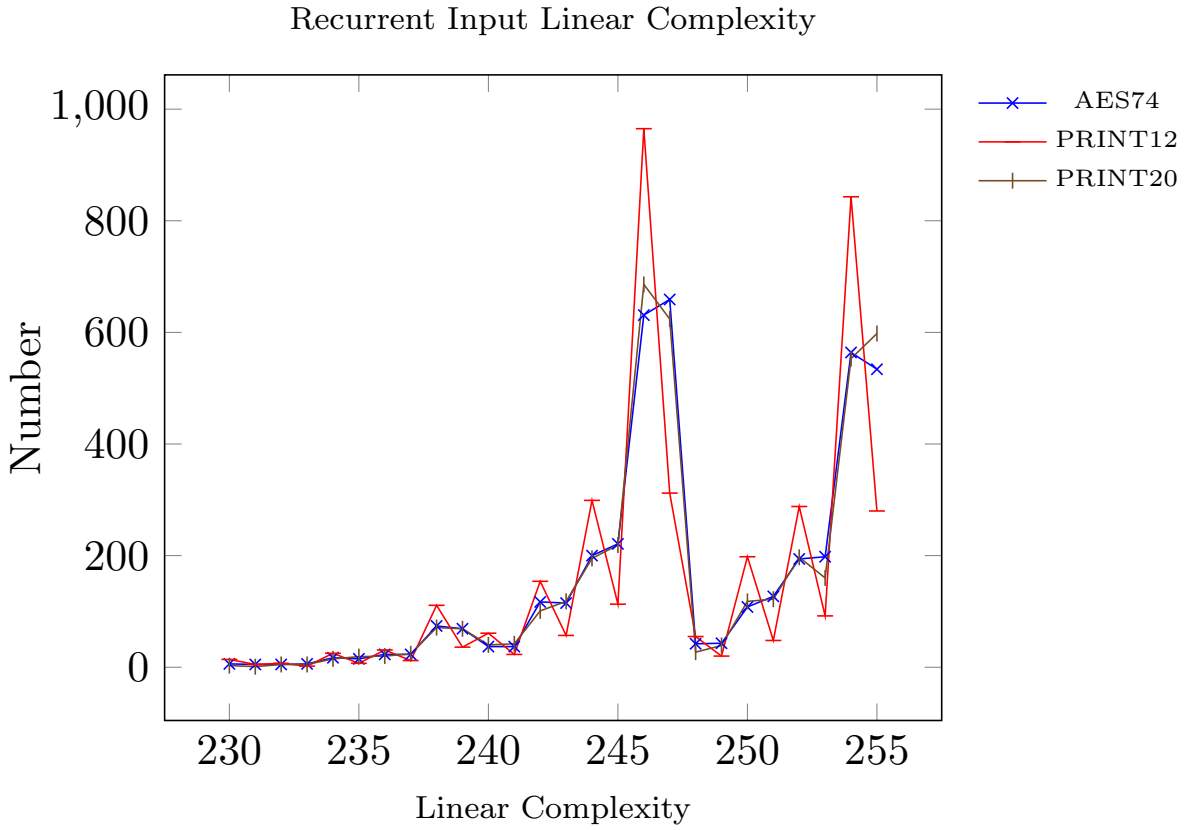

Figure 6: Distributions of component sequence's linear complexity with recurrent inputs: AES and PRINTcipher

and the nonlinearity of component functions using a random primitive as the multi-output filter. Comparing this ideal distribution with those of f_1, AES, KASUMI and PRESENT obtained by performing experiments, some non-randomness property of KASUMI is discovered. On the other hand, our experimental results show that f_1, AES and PRESENT perform very similar to the ideal case in the sense of the distributions of the algebraic degree and nonlinearity.

7.1 Algebraic degree distribution

Recall that the algebraic degree of a Boolean function is defined in Section 2. The following result states the number of Boolean functions with a given algebraic degree. The first part of the result can also be found in [10]. We provide a simple proof below for the completeness.

Theorem 2. *Let f be a Boolean function on $\mathbb{F}_2^n$. Then the number of Boolean functions with algebraic degree at most d is $2^{\sum_{i=0}^{d}\binom{n}{i}}$, and the number of Boolean functions with algebraic degree exactly d is $\left(2^{\binom{n}{d}}-1\right)2^{\sum_{i=0}^{d-1}\binom{n}{i}}$.*

Proof. Denoting the set $\Omega = \{0, 1, \ldots, n-1\}$. Let the ANF of f be $f(x) = \sum_{I\in\mathcal{P}(\Omega)} a_I x^I$. If the degree of f is at most d, then all $a_I = 0$ for $|I| > d$. Clearly there are $\sum_{i=0}^{d}\binom{n}{i}$ terms in the ANF of f with $|I| \le d$, and their coefficients can be either 0 or 1. Therefore there are $2^{\sum_{i=0}^{d}\binom{n}{i}}$

Boolean functions with degree at most d. For simplicity, let us denote by A_d the number of Boolean functions with degree at most d. Then by noting the number of Boolean functions with degree exactly d is $A_d - A_{d-1}$ we obtain the result. □

Corollary 1. *Let $\mathcal{C}$ be a random cryptographic primitive and $\mathcal{L}$ be an n-stage LFSR whose characteristic polynomial is a primitive polynomial of degree n. We use $\mathcal{C}$ as a multi-output filtering function and $\mathcal{L}$ to generate the inputs of $\mathcal{C}$. Then the probability of the component functions having degree at most d is $\frac{2^{\sum_{i=0}^{d}\binom{n}{i}}}{2^{2^n}}$. In particular, $Pr(d \leq n-3) = \frac{1}{2^{n+1}}$.*

Several remarks on the application of Theorem 2 are in the sequel:

(1) Assume the primitive $\mathcal{C}$ is used to generate MACs (for instance the function f_1 in TUAK). If the percentage of component functions with degree less than $n-2$ is large, then we may use the decoding method of the Reed-Muller code $R(n, n-3)$ to forge the MACs. See [30] for the Reed-Muller decoding. Note that the code $R(n, n-3)$ is the set of Boolean functions on $\mathbb{F}_{2^n}$ with algebraic degree at most $n-3$. Therefore, we need the probability $\Pr(d \leq n-3)$ to be as small as possible.

(2) On the other way, as shown in Corollary 1, for a random primitive, the probability $\Pr(d \leq n-3) = \frac{1}{2^{n+1}}$. So, for the primitive $\mathcal{C}$, if this probability is very different with $\frac{1}{2^{n+1}}$, some non-randomness properties may be exploited.

(3) The probability $\Pr(d \leq n-3)$ is actually affected by the diffusion property of the primitive $\mathcal{C}$. Assumed $\mathcal{C}$ is a keyed primitive from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^m}$. In the modern design of ciphers, by increasing the number of iteration rounds, normally $\mathcal{C}$ could attain the maximal possible degree for any key K. For a keyed primitive $\mathcal{C}_K$, in the multi-output model, we restrict the inputs of $\mathcal{C}_K$ to a subspace $\mathcal{S}$ generated by an LFSR. For a fixed key K, $\mathcal{C}_K$ can be regarded as a vectorial function and the ANF of $\mathcal{C}_K$ has the form $\mathcal{C}_K(x) = \sum_{I \in \mathcal{P}(\Omega)} a_I(K) x^I$, where $\Omega = \{0, \ldots, n-1\}$ and $\mathcal{P}(\Omega)$ is the power set and $a_I(K) \in \mathbb{F}_{2^m}$ are the coefficients of x^I (a_I is a function with K as the variable) [10]. Then the restrictions of $\mathcal{C}_K|_{\mathcal{S}} = \sum_{I \in \mathcal{P}(\Omega), I \subset \mathcal{S}} a_I(K) x^I$. The degree d of the component functions is then determined by a_I with $|I| = d$. If the diffusion property of $\mathcal{C}$ and the key generating algorithm are good, it should be very rare that all $a_I = 0$ for $|I| \geq \dim(\mathcal{S}) - 2$.

To better understand Theorem 2 and the above comments, for f_1, AES, KASUMI and PRESENT, we perform the following test on the distribution of the algebraic degree of their component functions.

Statistical Test 1. *By Corollary 1, using an LFSR with a primitive polynomial of degree 8, the probability that the degree of the component functions is smaller than 7 is $\frac{1}{2^9} = 19.53125 \times 10^{-4}$. For f_1, AES, KASUMI and PRESENT, we apply the multi-output filtering model as in Section 3.1. We choose $50,000$ keys for these primitives and compute the degree of the component functions. The probability of the degree is smaller than 7 is listed in the following table.*

Table 4: Distribution of the degree smaller than 7

Cryptographic primitive	$\Pr(d \leq 6)$
Random function	19.53125×10^{-4}
f_1	19.87×10^{-4}
AES	19.77×10^{-4}
KASUMI	20.16×10^{-4}
PRESENT	19.58×10^{-4}

From Table 4, we can see that for KASUMI, the probability $Pr(d \leq 6)$ is much higher than the one for other ciphers. To confirm this, we test another 50000 keys and found the probability is very close to it. This points out a distinguisher of KASUMI and other ciphers in Table 4.

Remark 2. *In the multi-output filtering model, fixing some inputs to constants and varying remaining bits of the inputs according to the LFSR and viewing the m-bit outputs to find non-randomness is similar to the saturation attack settings [29]. We viewed the 2^ℓ m-bit outputs as component Boolean functions and study their algebraic degrees. For this case, the order of the inputs does not matter. However, the order of the inputs generated by the LFSR must be preserved when the properties of component sequences are considered.*

7.2 Nonlinearity distribution

The nonlinearity of a Boolean function is one of the most important cryptographic properties. A highly nonlinear function is used to avoid the linear attack and its variants. Let f be a Boolean function on $\mathbb{F}_2^n$. The *nonlinearity* of f is defined in Section 2. One can see easily from its definition that, in other words,

$$\mathrm{NL}(f) = \max_{g \in \mathrm{RM}(1,n)} d(f, g),$$

where $\mathrm{RM}(1, n)$ denotes all Boolean functions with degree at most 1, and $d(f, g)$ is the weight of the sequence $(f(x) + g(x) : x \in \mathbb{F}_2^n)$. It is well known that when n is even the best nonlinearity a Boolean function may achieve is $2^{n-1} - 2^{n/2-1}$ and such functions are called bent functions (see [10] for more details). However, such functions are very rare. For a random Boolean function, we have the following result on the distribution of its nonlinearity.

Theorem 3 ([38, 10])**.** *Let c be any strictly positive real number. The density of the set*

$$\left\{ f \in \mathcal{B}_n,\ NL(f) \geq 2^{n-1} - c\sqrt{n}2^{\frac{n-1}{2}} \right\}$$

is greater than $1 - 2^{n+1-c^2 n \log_2 e}$. If $c^2 \log_2 e > 1$, then this density tends to 1 when n tends to infinity.

Applying the above theorem on Boolean functions with 8 variables, we have the following table. Note that the best nonlinearity we expect for Boolean functions with 8 variables is $2^7 - 2^3 = 120$.

Table 5: Lower bound of the density of Boolean functions in $\mathcal{B}_8$ with nonlinearity greater than W

Lower Bound W of NL	Lower bound of the density of Boolean functions with $\mathrm{NL}(f) \geq W$
98	0.54747879061478987802997919600 8
97	0.71902310151075402984781111703 1
96	0.82824221024964787460062882576 5
95	0.89663430607249953273680824529 9
94	0.93875783156791138635120360571 3
93	0.96427774651450020080734327382 1
92	0.97948642802537161847775244745 5
91	0.98840267349024055409268340534 3
90	0.99354511316750952827725848552 4

From the above table, one can see that if the component functions of f_1 are random, the probability that the component Boolean functions have nonlinearity smaller than 90 is very small, which is $1 - 0.99354511316750952827725848552 4 \approx 0.00645$. In view of this, we perform the following statistical test for f_1, AES, KASUMI and PRESENT.

Statistical Test 2. *Let the LFSR and the other settings be the same as in* **Statistical Test 1**. *We list the distribution of the nonlinearity of the component functions of f_1 and AES in the following table. Since only the component functions with smallest nonlinearity are important to us (as an attacker), we only list the probability that a Boolean function has nonlinearity smaller than 90 or 91. The notation $\Pr_{<W}$ denotes the probability that the nonlinearity is smaller than W.*

Table 6: The distribution of the nonlinearity of component sequences of f_1, AES, KASUMI and PRESENT

Cryptographic primitive	$\Pr_{<90}$	$\Pr_{<91}$
Random Function	0.006455	0.011597
f_1	0.000299	0.000690
AES	0.000306	0.000592
KASUMI	0.000299	0.000565
PRESENT	0.000308	0.000589

Unlike the distribution of the algebraic degree, from the above table we can not see obvious difference among these four ciphers. However, one can still see that the probability values $Pr_{<90}$ and $Pr_{<91}$ is still very different with the random case (although they are only the ***upper bounds*** *of the probability).*

Although now we cannot derive attacks from Statistical Test 1 and Statistical Test 2, it is interesting to observe some non-randomness in the aspect of the distribution of cryptographic properties.

8 Concluding Remarks and Future Work

In this paper, we introduced the multi-output filtering model for analyzing the security of a cryptographic primitive. In this model, a cryptographic primitive is used as a multi-output filtering function and a number of component sequences and component functions of the primitive are obtained. We aimed at exploiting the security properties of the primitive through studying its component sequences and functions.

Thanks to the fruitful research outcome in the theory of sequences and Boolean functions, we propose a general distinguish attack technique under IND-CPA. We developed a new object, called a distinguishing function, to characterize the success rate of our new attack method. Interestingly enough, for a primitive $\mathcal{C}$, by comparing the distribution of the linear complexity of the component sequences generated by two sets of inputs, we can construct a new distinguishing function. The importance of this new distinguishing function is demonstrated by launching an attack on KASUMI and PRESENT with non-negligible success rates.

Furthermore, we studied the cryptographic properties of the component functions. By comparing the distribution of the algebraic degree and nonlinearity properties with that of a random one, we discovered that, for KASUMI, its distribution of the algebraic degree is very different, while the distribution of f_1, AES and PRESENT are not. We could not propose any immediate attack based on this observation, but it is worth to pointing it out for future research.

As a future work, we believe it is important to study which inner structure of a primitive affects the distribution of the linear complexity, algebraic degree, nonlinearity, and other properties of component sequences and functions. This study may lead to a new attacking method, and present new criteria on designing a cryptographic primitive.

Acknowledgement. The authors would like to thank the reviewers of the C2SI-Carlet 2017 conference for their insightful comments to improving the quality of the paper. The authors sincerely thank Reviewer 3 for pointing out an error in Corollary 1 and also mentioning a connection between Statistical test 1 and the saturation attack.

References

[1] Aumasson, J.-P., Meier, W.: Zero-sum distinguishers for deduced Keccak-f and for the core functions of Luffa and Hamsi. Presented at the rump session of CHES 2009 (2009)

[2] Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. Advances in Cryptology CRYPTO '98, LNCS, vol. 1462, pp. 26 – 45. Springer Berlin Heidelberg (1998)

[3] Bellare, M., Desai, A., Jokipii E., Rogaway, P.: A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. Proceedings of the 38th Symposium on Foundations of Computer Science, IEEE (1997)

[4] Bernstein, D.J.: Second preimages for 6 (7? (8??)) rounds of Keccak?. `http://ehash.iaik.tugraz.at/uploads/6/65/NIST-mailing-list_Bernstein-Daemen.txt` (2010)

[5] Bertoni, G., Daemen, J., Peeters, M., Assche G.V.: The Keccak reference. `http://keccak.noekeon.org/Keccak-reference-3.0.pdf` (2011)

[6] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Cryptographic sponge functions, January 2011, http://sponge.noekeon.org/.

[7] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak sponge function family main document, submission to NIST (updated), Version 1.2, 2009.

[8] Bogdanov, A., Knudsen, L.R., Leander, G. Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher, Cryptographic Hardware and Embedded Systems - CHES 2007, LNCS, vol. 4727, pp. 450 – 466. Springer Berlin Heidelberg (2007)

[9] Boura, C., Canteaut, A.: Zero-sum distinguishers for iterated permutations and application to Keccak-f and Hamsi-256. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2011. LNCS, vol. 6544, pp. 1 – 17. Springer-Heidelberg (2011)

[10] Carlet, C.: Boolean functions for cryptography and error correcting codes, Chapter of the monography boolean models and methods in mathematics, computer science, and engineering, Cambridge University Press, Yves Crama and Peter L. Hammer (eds.), pp. 257-397. (2010)

[11] Daemen, J., Rijmen, V.: The Design of Rijndael, AES – The Advanced Encryption Standard. Springer (2002)

[12] Daemen, J., Van Assche, G.: Differential propagation analysis of Keccak. Fast Software Encryption, FSE 2012. LNCS, vol. 7549, pp. 422 – 441. Springer Berlin Heidelberg (2012)

[13] Daemen, J.: Permutation-based encryption, authentication and authenticated encryption. DIAC (2012)

[14] Dinur, I., Morawiecki, P., Pieprzyk, J., Srebrny, M., Straus, M.: Practical complexity cube attacks on round-reduced Keccak sponge function. Cryptology ePrint Archive, Report 2014/259 (2014) `http://eprint.iacr.org/`

[15] Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. Advances in Cryptology-EUROCRYPT '09, LNCS, pp. 278–299. Springer-Verlag (2009)

[16] Dinur, I., Dunkelman, O., Shamir, A.: New attacks on Keccak-224 and Keccak-256. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 442-461, Springer, Heidelberg (2012)

[17] Dinur, I., Dunkelman, O., Shamir, A.: Collision attacks on up to 5 rounds of SHA-3 using generalized internal differentials. Cryptology ePrint Archive, Report 2012/627. (2012) `http://eprint.iacr.org/`

[18] Duc, A., Guo, J., Peyrin, T., Wei, L., Unaligned rebound attack: Application to Keccak. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 402-421, Springer, Heidelberg (2012)

[19] Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28, 270 – 299 (1984)

[20] Golomb, S.W., Gong, G.: Signal design for good correlation – for wireless communication, cryptography and radar. Cambridge Press, 2005.

[21] Gong, G., Mandal, K., Tan, Y., Wu, T.: Security Assessment of TUAK Algorithm Set, 2014.

[22] Homsirikamol, E., Morawiecki, P., Rogawski, M., Srebrny, M.: Security margin evaluation of SHA-3 contest finalists through sat-based attacks. In A. Cortesi, N. Chaki, K. Saeed, and S.T. Wierzchon, (eds.). CISIM, LNCS, vol. 7564, pp. 56 – 67. Springer (2012)

[23] Key, E.L.: An analysis of the structure and complexity of nonlinear binary sequence generators. IEEE Transactions on Information Theory 22, 732 – 736. (1976)

[24] Knudsen, Lars, Gregor Leander, Axel Poschmann, and Matthew JB Robshaw. PRINTcipher: a block cipher for IC-printing. In Cryptographic Hardware and Embedded Systems, CHES 2010, pp. 16-32. Springer Berlin Heidelberg, 2010.

[25] Lai, X., Duan, M.: Improved zero-sum distinguisher for full round Keccak-f permutation. Cryptology ePrint Archive, Report 2011/023 (2011), http://eprint.iacr.org/2011/023

[26] Lathrop, J.: Cube attacks on cryptographic hash functions [EB/OL], Master's Thesis. (2009) http://www.cs.rit.edu/~jal6806/thesis/.

[27] Leander, G., Abdelraheem, M.A., AlKhzaimi, H., Zenner, E.: A cryptanalysis of PRINTcipher: The invariant subspace attack. In Rogaway, P. (ed.), CRYPTO 2011. LNCS, vol. 6841, pp. 206 – 221. Springer (2011)

[28] Leander, Gregor, Mohamed Ahmed Abdelraheem, Hoda AlKhzaimi, and Erik Zenner. "A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack." In CRYPTO, vol. 6841, pp. 206-221. (2011)

[29] Lucks, S.: The Saturation Attack — A Bait for Twofish. In Fast Software Encryption 2001, LNCS, vol 2355. pp. 1–15. Springer, Berlin, Heidelberg (2002)

[30] MacWilliams, F.J., Sloane, N.J.A.: The theory of error-correcting codes. North-Holland Mathematical Library. (1977)

[31] Matsui, M.: Linear cryptanalysis method for DES cipher. EUROCRYPT '93, LNCS vol. 765, pp. 55–64. (1994)

[32] Meidl, W., Niederreiter, H.: On the expected value of the linear complexity and the k-error linear complexity of periodic sequences. IEEE Transaction on Information Theory, 48(11) 2817–2825. (2002)

[33] Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography, CRC Press (1997)

[34] Morawiecki, P., Pieprzyk, J., Srebrny M., Straus, M.: Preimage attacks on the round-reduced Keccak with the aid of differential cryptanalysis, Cryptology ePrint Archive, Report 2013/561 (2013) http://eprint.iacr.org/

[35] Morawiecki, P., Pieprzyk, J., Srebrny, M.: Rotational cryptanalysis of round-reduced KECCAK, Cryptology ePrint Archive, Report 2012/546. (2012) http://eprint.iacr.org/.

[36] Naya-Plasencia, Røck, M.A., Meier, W.: Practical analysis of reduced-round Keccak. In: Bernstein, D.J., Chatterjee, S. (eds.) INDOCRYPT 2011. LNCS, vol. 7107, pp. 236-254. Springer, Heidelberg (2011)

[37] NIST, the SHA-3 competition (2007-2012). http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

[38] Olejar, D., Stanek, M.: On Cryptographic properties of random boolean functions. Journal of Universal Computer Science, 4(8), 705 – 717. (1998)

[39] Rueppel, R.A.: Analysis and design of stream ciphers. Springer-Verlag, Berlin (1986)

[40] Uspensky, J.V.: Introduction to mathematical probability. New York McGraw-Hill (1937)

[41] Tan, Y., Mandal, K., Gong, G.: Characterization of column parity kernel and differential cryptanalysis of Keecak. CACR 2014-01, University of Waterloo. (2014) http://cacr.uwaterloo.ca/.

[42] Todo, Y.: Integral Cryptanalysis on Full MISTY1. In CRYPTO 2015. LNCS, vol. 9215, pp. 413-432. (2015)

[43] 3^{rd} generation partnership project, Technical specification group services and system aspects, 3G security, specification of the 3GPP confidentiality and integrity algorithms; Document 2: KASUMI specification, V.3.1.1, 2001.

[44] Specification of the TUAK algorithm set: A second example algorithm set for the 3GPP authentication and key generation functions $f_1, f_1^*, f_2, f_3, f_4, f_5$ and f_5^*, SP-130602, ETSI/SAGE, Dec 13, 2013. `http://www.3gpp.org/ftp/tsg_sa/TSG_SA/TSGS_62/ftp-TdocsByTdoc_SP-62.htm`

A APPENDIX

Here we present a description of TUAK's f_1 algorithm, the proofs of Theorem 1, and the slope of the linear complexity distribution of f_1 and AES, KASUMI and PRESENT.

A.1 Overview of TUAK Algorithm Set

The TUAK algorithm set is designed to generate message authentication codes (MAC) and various keys such as cipher keys and integrity keys in mobile communications. The TUAK algorithm set consists of seven algorithms, namely f_1, f_1^*, f_2, f_3, f_4, f_5, f_5^*, which are built upon the Keccak permutation Keccak-f[1600] [44]. Each algorithm in TUAK is used to perform some specific task, for instance f_1 and f_1^* are used to generate MACs, f_2 to f_5 are used to output signed response (RES), confidentiality key (CK), integrity key (IK), anonymity key (AK), respectively. The MAC and various keys are expected to guarantee the security in mobile communications.

A.2 Description of f_1

Generally speaking, all the algorithms in TUAK are built by assigning some specific inputs at some predefined input positions of the Keccak permutation, and then by extracting M bits from predefined output positions of the Keccak permutation. For a detailed description of all TUAK algorithms, the reader is referred to [44]. In particular, we introduce the algorithm f_1 as it plays the role of generating MACs. We provide an overview of the f_1 algorithm in Fig. 7.

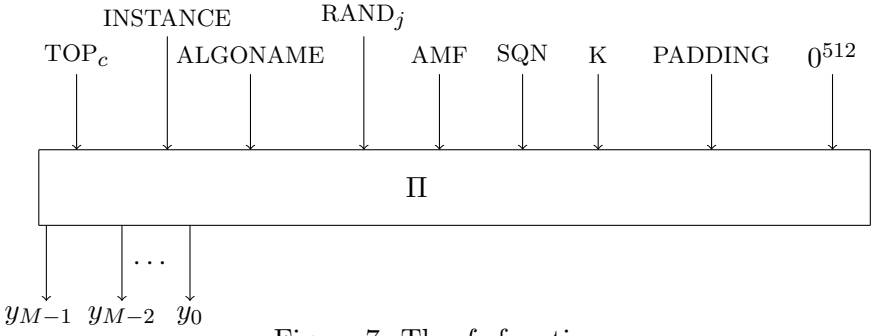

Figure 7: The f_1 function

Mathematically, we can write f_1 in the form:

$$f_1 \triangleq \Pi\,(\text{INPUT}) = (y_0, y_1 \cdots, y_{M-1}), \tag{11}$$

where M is the length of the MAC and INPUT is defined as

$$\text{INPUT} = \text{TOP}_c||\text{INSTANCE}||\text{ALGORITHM}||\text{K}||\text{RAND}||\text{AMF}||\text{SQN}||\text{PADDING}||0^{512}. \quad (12)$$

Note that, in the INPUT, except K, RAND, SQN, the other parameters are all prescribed constants. For details, see [44]. For the convenience, in the rest of the paper, we write f_1 as

$$f_1(\text{K}, \text{RAND}, \text{SQN}) = (y_0, y_1 \cdots, y_{M-1}).$$

The algorithm f_1 is flexible with the length of the parameters. The key length is 128 or 256 bits, the length of RAND is 128 bits, the length of SQN is 48 bits, and the possible output lengths are 64, 128, and 256.

A.3 Slope of the linear complexity distribution

Here we present the results of the test in Section 5.2 in Table 7. The slope in Table 7 is the average slope over 10^8 samples. The column "Slope (L)" contains the slopes computed from the LFSR input, and the column "Slope (R)" contains the slopes computed from the random input. The last column shows the absolute value of the difference between "Slope(L)" and "Slope(R)". We can see the "Difference" of KASUMI and PRESENT are much greater than f_1 and AES.

Table 7: The slope of f_1, AES, KASUMI and PRESENT on average

Primitive	Polynomial of the LFSR	Slope (L)	Slope (R)	| Difference |
f_1	$x^8 + x^6 + x^4 + x^3 + x^2 + x^1 + 1$	-0.125	-0.124	2.210×10^{-4}
AES	$x^8 + x^7 + x^6 + x^3 + x^2 + x^1 + 1$	-0.088	-0.087	5.190×10^{-4}
KASUMI	$x^8 + x^7 + x^6 + x^5 + x^4 + x^2 + 1$	0.130	0.015	0.115
PRESENT	$x^8 + x^6 + x^5 + x^3 + 1$	-0.057	0.036	0.093