# Revisiting Higher-Order Differential-Linear Attacks from an Algebraic Perspective

Kai Hu(✉), Thomas Peyrin, Quan Quan Tan, and Trevor Yap

School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, Singapore
kai.hu.sdu@gmail.com, {thomas.peyrin,trevor.yap}@ntu.edu.sg, quanquan001@e.ntu.edu.sg

**Abstract.** The Higher-order Differential-Linear (HDL) attack was introduced by Biham *et al.* at FSE 2005, where a linear approximation was appended to a Higher-order Differential (HD) transition. It is a natural generalization of the Differential-Linear (DL) attack. Due to some practical restrictions, however, HDL cryptanalysis has unfortunately attracted much less attention compared to its DL counterpart since its proposal.

In this paper, we revisit HD/HDL cryptanalysis from an algebraic perspective and provide two novel tools for detecting possible HD/HDL distinguishers, including: (a) Higher-order Algebraic Transitional Form (HATF) for probabilistic HD/HDL attacks; (b) Differential Supporting Function (DSF) for deterministic HD attacks. In general, the HATF can estimate the biases of $\ell^{th}$-order HDL approximations with complexity $\mathcal{O}(2^{\ell+d2^{\ell}})$ where $d$ is the algebraic degree of the function studied. If the function is quadratic, the complexity can be further reduced to $\mathcal{O}(2^{3.8\ell})$. HATF is therefore very useful in HDL cryptanalysis for ciphers with quadratic round functions, such as Ascon and Xoodyak. DSF provides a convenient way to find good linearizations on the input of a permutation, which facilitates the search for HD distinguishers.

Unsurprisingly, HD/HDL attacks have the potential to be more effective than their simpler differential/DL counterparts. Using HATF, we found many HDL approximations for round-reduced Ascon and Xoodyak initializations, with significantly larger biases than DL ones. For instance, there are deterministic $2^{nd}$-order/$4^{th}$-order HDL approximations for Ascon/Xoodyak initializations, respectively (which is believed to be impossible in the simple DL case). We derived highly biased HDL approximations for 5-round Ascon up to $8^{th}$ order, which improves the complexity of the distinguishing attack on 5-round Ascon from $2^{16}$ to $2^{12}$ calls. We also proposed HDL approximations for 6-round Ascon and 5-round Xoodyak (under the single-key model), which couldn't be reached with simple DL so far. For key recovery, HDL attacks are also more efficient than DL attacks, thanks to the larger biases of

The full version of this paper is [14].

J. Guo and R. Steinfeld (Eds.): ASIACRYPT 2023, LNCS 14440, pp. 405–435, 2023.
https://doi.org/10.1007/978-981-99-8727-6_14

HDL approximations. Additionally, HATF works well for DL ($1^{st}$-order HDL) attacks and some well-known DL biases of Ascon and Xoodyak that could only be obtained experimentally before can now be predicted theoretically.

With DSF, we propose a new distinguishing attack on 8-round Ascon permutation, with a complexity of $2^{48}$. Also, we provide a new zero-sum distinguisher for the full 12-round Ascon permutation with $2^{55}$ time/-data complexity. We highlight that our cryptanalyses do not threaten the security of Ascon or Xoodyak.

**Keywords:** Higher-Order Differential · Higher-Order Differential-Linear · Ascon · Xoodyak

## 1 Introduction

### 1.1 Differential-Linear Cryptanalysis

Differential and linear cryptanalysis have been the fundamental methods for evaluating the security of a cipher [5,22]. Nowadays, all new schemes are requested to claim resistance against these two attacks. However, resistance against the plain differential and linear cryptanalysis does not necessarily lead to resistance against their variants. For example, despite its security proof against differential attacks, the cipher Coconut98 [28] is vulnerable to boomerang and Differential-Linear (DL) cryptanalysis [3,29] which are two variants of the differential and linear attacks, leveraging a combined strategy.

Differential-linear cryptanalysis was proposed by Langford and Hellman in 1994 [18]. For a cipher $E$, let $C = E(P)$ and $C' = E(P')$. Given a difference-mask pair $(\Delta_I, \lambda_O)$, the bias $q'$ of a DL approximation can be derived from the following equation

$$\Pr[\lambda_O \cdot (C \oplus C') = 0 \mid P \oplus P' = \Delta_I] = \frac{1}{2} + q'.$$

Similar to the case of linear cryptanalysis, if the bias $|q'|$ is significantly larger than 0, we can distinguish the cipher from a random permutation.

There are mainly two types of methods to estimate $q'$ in the literature. In the classical DL cryptanalysis [3,18], a cipher $E$ is decomposed into two sub-ciphers as $E = E_1 \circ E_0$, where a differential $\Delta_I \xrightarrow{p} \Delta_O$ for $E_0$ and a linear approximation $\lambda_I \xrightarrow{q} \lambda_O$ for $E_1$ are considered. The DL bias $q'$ can be estimated by $q' = (-1)^{\Delta_O \cdot \lambda_I} 2pq^2$ under some independence assumptions.

As pointed out in [3], experiments are required to verify the estimated bias when possible because the underlying assumptions may fail. There are two main refined methods of classical DL attacks. One is from Blondeau *et al.* [6], where an accurate formula for $q'$ is given under the sole assumption that $E_0$ and $E_1$ are independent. The other, proposed by Bar-On *et al.* [1] at EUROCRYPT 2019, is called the Differential-Linear Connectivity Table (DLCT) which overcomes the independence problem between $E_0$ and $E_1$. The drawback of the first method is that it is computationally impossible to apply the formula for practical use-cases, while the second method only works when a large-enough DLCT can be built efficiently.

A new method to estimate $q'$ from an algebraic perspective has been proposed by Liu *et al.* [20] at CRYPTO 2021. If we define a Boolean function according to $\lambda_O$ as $f_{\lambda_O} : \mathbb{F}_2^n \to \mathbb{F}_2, f_{\lambda_O}(u) = \lambda_O \cdot u$ and let $f = f_{\lambda_O} \circ E$, the bias of $\lambda_O \cdot (C \oplus C')$ is equivalent to the bias of the following Boolean function

$$\mathcal{D}_{\Delta_I} f(P) = f(P) \oplus f(P \oplus \Delta_I). \tag{1}$$

Then, they introduced another function with an auxiliary variable $x \in \mathbb{F}_2$ as

$$f_{\Delta_I}(P, x) = f(P \oplus x\Delta_I), \tag{2}$$

where $x\Delta_I \in \mathbb{F}_2^n$ means that $x$ is multiplied with each coordinate of $\Delta_I$, *i.e.*, $x\Delta_I = (x\Delta_I[0], \ldots, x\Delta_I[n-1])$. Given a Boolean function $g(a_0, a_1, \ldots, a_{n-1})$ with $n$ variables and for a certain variable $a_i$ ($a_j$ for $j \neq i$ are viewed as parameters), we can write $g$ as $g = g'' a_i \oplus g'$ with $g'$ and $g''$ being independent of $a_i$ and where the partial derivative of $g$ with respect to $a_i$ is the polynomial $g''$, denoted by $D_{a_i} g$. Liu *et al.* gave the following observation linking Eqs. 1 and 2 (Eq. 3, as proposed in [20], was initially presented based on intuition, with the underlying rationale not explicitly discussed. A comprehensive explanation and detailed insight into the reasoning behind this formula will be provided in Sect. 3),

$$f'' = D_x f_{\Delta_I} = \mathcal{D}_{\Delta_I} f, \tag{3}$$

where $D_x f_{\Delta_I}$ is the partial derivative of $f_{\Delta_I}$ with respect to $x$. That is to say, considering Eqs. 1, 2, 3, in order to evaluate the bias of $\lambda_O \cdot (C \oplus C')$, we only need to evaluate the bias of the Boolean function $D_x f_{\Delta_I}$. This estimation from an algebraic perspective does not require any assumption in theory. However, it is extremely difficult to derive $D_x f_{\Delta_I}$ or evaluate its bias. To overcome this obstacle, Liu *et al.* introduced the so-called Algebraic Transitional Forms (ATF)[1] technique to construct a transitional expression of $D_x f_{\Delta_I}$. Then, the bias is estimated from this transitional expression.

### 1.2 Higher-Order Differential(-Linear) Cryptanalysis

Inspired by the boomerang and DL cryptanalysis, other combined attacks were studied by Biham, Dunkelman, and Keller [4]. These combined attacks include the differential-bilinear, Higher-order Differential-Linear (HDL), boomerang-linear attack, *etc.*

The Higher-order Differential (HD) was for the first time introduced by Lai in 1994 [30] and later studied by Knudsen [16]. It is a natural generalization of

[1] In [20], there is another terminology DATF when ATF is used to construct transitional expressions for $f_\Delta$. In this paper, we directly use ATF for all kinds of Boolean functions no matter whether we target $f$ or $f_\Delta$.

the differential attack that takes advantage of having access to more plaintexts. Given an $\ell^{th}$-order difference $\boldsymbol{\Delta}_I = (\Delta_0, \Delta_1, \ldots, \Delta_{\ell-1})$ where $\Delta_0, \Delta_1, \ldots, \Delta_{\ell-1}$ are linearly independent, the $\ell^{th}$ derivative of a (partial) cipher $E$ with respect to $\boldsymbol{\Delta}_I$ studies the probability

$$p = \Pr\left[\bigoplus_{x \in X \oplus \mathcal{L}(\boldsymbol{\Delta}_I)} E(x) = \Delta_O\right],$$

where $\mathcal{L}(\boldsymbol{\Delta}_I)$ is the linear span of $(\Delta_0, \Delta_1, \ldots, \Delta_{\ell-1})$, the $\ell$ dimensional affine space $X \oplus \mathcal{L}(\boldsymbol{\Delta})$ is called the *input set* with respect to $\boldsymbol{\Delta}$, and $\Delta_O$ is called the output difference.

As the name higher-order differential-linear suggests, HDL cryptanalysis [4] studies the bias concerning an $\ell^{th}$-order input difference $\boldsymbol{\Delta}_I$ and an output mask $\lambda_O$. The bias $\varepsilon$ of an HDL approximation is derived from the following formulation:

$$\Pr\left[\lambda_O \cdot \left(\bigoplus_{x \in X \oplus \mathcal{L}(\boldsymbol{\Delta}_I)} E(x)\right) = 0\right] = \frac{1}{2} + \varepsilon.$$

Akin to the first kind of method to evaluate the bias in DL cryptanalysis, Biham *et al.* [4] gave an analysis based on viewing $E$ as two sub-ciphers $E = E_1 \circ E_0$. Suppose that we know an $\ell^{th}$ derivative with probability $p$ for $E_0$ and that $E_1$ has a linear approximation with bias equal to $q$, then the overall bias $\varepsilon$ is estimated as $\varepsilon = 2^{2^\ell - 1} p q^{2^\ell}$. However, currently there is no effective method to trace the propagation of an HD or calculate its probability yet. Thus, Biham *et al.* had to restrain themselves to the integral property for $E_0$, which leads to $p = 1$. The integral property usually requires a large $\ell$ to attack an interesting number of rounds, but if $|q| \neq \frac{1}{2}$, $\varepsilon$ will become extremely close to zero. As a result, we can only get an interesting HDL distinguisher when there is a linear approximation with bias $\pm\frac{1}{2}$ for $E_1$. In practice, some ciphers such as IDEA [17] allow weak-key linear approximations with bias $\frac{1}{2}$, which makes them vulnerable to HDL attacks [2,4].

### 1.3 Motivation and Contributions

Considering that DL attacks are efficient for many important primitives, such as Ascon [12] (recent winner of the NIST lightweight competition) and Xoodyak [9], we are naturally interested in whether the HDL attack could achieve even better performance. However, as we mentioned, we did not have any tool to study the probabilistic HD distinguishers and they were far less practical than their DL counterparts. How to handle the probabilistic HD/HDL cryptanalysis remains an open problem.

Recently, an algebraic perspective on DL attacks [20] opened up a new road to study the differential/DL attacks and achieved better precision for some important ciphers such as Ascon [12]. However, we note that their method is based on some intuitive observations and is limited to the first-order case. In this paper, we generalize and refine this algebraic method to higher-order cases.

**Our Contributions.** In this paper, we revisit the HD/HDL cryptanalysis of a Boolean function from an algebraic perspective, which provides novel methods to study HD and HDL cryptanalysis. Two tools for HD/HDL cryptanalysis are proposed based on this new perspective, one is the Higher-order Algebraic Transitional Form (HATF), which is used to detect probabilistic HDL approximations. The other is the Differential Supporting Function (DSF), which is useful to find deterministic HD distinguishers.

**Table 1.** Approximation Biases of the DL and HDL approximations for Ascon, Xoodyak and Xoodoo. The column of Expr. shows the experimental biases.

| Primitive | Round | Order | Bias | | Method | Reference |
|---|---|---|---|---|---|---|
| | | | Expr. | Theory | | |
| Ascon Init. | 4 | 1st | $2^{-2}$ | $2^{-20}$ | Classical | [11] |
| | | | | $2^{-5}$ | DLCT | [1] |
| | | | | $2^{-2.365}$ | ATF | [20] |
| | | | | $\mathbf{2^{-2.09}}$ | **HATF** | **Section** 5.1 |
| | | 2nd | $2^{-1}$ | $\mathbf{2^{-1}}$ | **HATF** | **Section** 5.1 |
| | 5 | 1st | $2^{-9}$ | – | Experimental | [11] |
| | | | | $\mathbf{2^{-10}}$ | **HATF** | **Section** 5.1 |
| | | 2nd | $2^{-6.60}$ | $\mathbf{2^{-7.05}}$ | **HATF** | **Section** 5.1 |
| | | 8th | $2^{-3.35}$ | $\mathbf{2^{-4.73}}$ | **HATF** | **Section** 5.1 |
| | 6 | 3rd | $2^{-22}$† | $\mathbf{2^{-25.97}}$† | **HATF** | **Section** 5.1 |
| Xoodyak Init. | 4 | 1st | $2^{-9.7}$ | – | Experimental | [13] |
| | | | | $\mathbf{2^{-9.67}}$ | **HATF** | **Section** 6.1 |
| | | | $-2^{-5.36}$‡ | – | Experimental | [13] |
| | | | | $\mathbf{-2^{-6.0}}$ | **HATF** | **Section** 6.1 |
| | | 2nd | $2^{-5.72}$ | $\mathbf{2^{-5.72}}$ | **HATF** | **Section** 6.1 |
| | | 4th | $2^{-1}$ | $\mathbf{2^{-1}}$ | **HATF** | **Section** 6.1 |
| | 5 | 2nd | – | $\mathbf{2^{-45}}$ | **HATF** | **Section** 6.1 |
| Xoodoo | 4 | - | $2^{-1}$ | $2^{-1}$ | Rot. DL | [21] |
| | | 4th | $2^{-1}$ | $\mathbf{2^{-1}}$ | **HATF** | **Section** 6.1 |
| | 5 | 3rd | $2^{-8.79}$ | $\mathbf{2^{-8.96}}$ | **HATF** | **Section** 6.1 |

†This bias holds when 24 conditions are satisfied.
‡In [13], this 4-round DL distinguisher was extended to 5 rounds in a natural way, with an additional cost of $2^{-4}$.

**Higher-Order Algebraic Transitional Form (HATF).** By transforming the input set of a Boolean function $f$ from an $\ell$ dimensional affine space to an

$\ell$ dimensional linear space, we can transform a general HD attack to a standard integral/cube attack. The HD/HDL approximations are then the biases of the coefficient of the maxterm in $f$ with the transformed inputs. Since almost all modern ciphers are built in a composite way, we can obtain the HD/HDL expressions in the form of a composite vectorial Boolean function, which is easier to study.

HATF is a way to estimate the biases of HDL approximations of ciphers (concretely, they are the expected biases among all input variabels). It is a two-step process: (a) constructing the composite formula of an HD/HDL expression for a cipher; (b) calculating the biases of state bits iteratively. The complexity of HATF is $\mathcal{O}(2^{\ell+d2^{\ell}})$ in general cases where $\ell$ is the HD/HDL order and $d$ is the algebraic degree of the round function. However, for ciphers with quadratic round functions, the complexity is $\mathcal{O}(2^{3.8\ell})$. Thus, HATF is a very useful tool to study the HDL approximations of some permutation-based ciphers such as Ascon and Xoodyak.

Using HATF, we detected many highly biased HDL approximations for round-reduced Ascon, Xoodyak and Xoodoo. For example, we propose deterministic HDL approximations for both Ascon and Xoodyak on 4 rounds. For 5-round Ascon, we give HDL approximations up to the $8^{th}$ order. Based on these, we have improved the distinguishing attacks for 4- and 5-round Ascon and Xoodyak (see Table 2).

We can improve the precision of HATF with a so-called partitioning technique as compared to all previous detection tools for DL (first-order HDL) attacks. For instance, HATF estimates the bias of the well-studied 4-round Ascon's DL approximation as $2^{-2.09}$ (the experimental results is $2^{-2}$), which is better than previous tools such as the DLCT [1] ($2^{-5}$) or the ATF [20] ($2^{-2.36}$). Also, for the first time we give the theoretical bias for the 5-round Ascon's DL approximation: the bias is estimated as $2^{-10}$ while the experimental value is $2^{-9}$, no previous tool could predict this bias. For Xoodyak, HATF also gives precise theoretical predictions for two DL approximations found by experiments [13]. These results are shown in Table 1.

In addition, by injecting some conditions into HATF, we obtained the best key-recovery attack on 5-round Ascon with time/data complexity of $2^{22}$, which is 16 times faster than the DL attacks [20] and 4 times faster than the conditional cube attacks [19]. For 4-round Xoodyak, the HDL attack is 4 times more efficient than the DL attack [13]. A summary of these key-recovery attacks is given in Table 2.

Finally, we make clear that HATF cannot give any lower or upper bound for HDL approximation biases in theory. However, empirically, it is quite precise to predict biased bits, as we show in our experiments. In cases where the reported bias is high, we note that it was always the case that the experimental bias was also observed as high (we have not seen any counterexample for this). We provide data and discuss the precision of HATF in the full version of this paper based on HDL cryptanalysis of Ascon.

**Differential Supporting Function (DSF).** Instead of using the degree evaluation of a cipher to derive deterministic HD distinguishers, we can evaluate the algebraic degree of its DSF. As we will see, the DSF is parameterized by the input value and the (higher-order) difference. Thus, a proper choice of the parameters could significantly reduce its algebraic degree, leading to a greater chance of detecting a deterministic HD distinguisher for the DSF. After that, we can conveniently transform it into an HD distinguisher for the original cipher. With this technique, we improve the best-known distinguishing attacks on round-reduced Ascon permutation [12]. A 46th-order HD will lead to a zero output difference (in 64 bits) for 8 rounds, *i.e.*, $2^{46}$ plaintexts are enough to distinguish an 8-round Ascon permutation from a random permutation (the previous best-known distinguisher requires $2^{130}$ computations [26]). With a similar method applied to the inverse Ascon permutation, we constructed a zero-sum distinguisher for a full 12-round Ascon permutation requiring only $2^{55}$ calls while the previous best zero-sum distinguisher costs $2^{130}$ calls. These distinguishers are demonstrated in Table 3.

**Table 2.** Summary of DL-like attacks on the Ascon and Xoodyak initializations. Cond. is short for conditional.

| Type | Rnd | Data(log) | Time (log) | Method | Reference |
|---|---|---|---|---|---|
| Ascon Initialization | | | | | |
| Distinguisher | 4 | 5 | 5 | DL | [11] |
| | | **2** | **2** | **2nd HDL** | **Section** 5.1 |
| | 5 | 18 | 18 | DL | [11] |
| | | **12** | **12** | **8th HDL** | **Section** 5.1 |
| Key-Recovery | 5 | 36 | 36 | DL | [11] |
| | | 31.44 | 31.44 | DL | [25] |
| | | 26 | 26 | Cond. DL | [20] |
| | | 24 | 24 | Cond. Cube | [19] |
| | | **22** | **22** | **Cond. HDL** | **Section** 5.2 |
| Best | 7 | 77 | 103 | Cond. Cube | [19] |
| | 7 | 64 | 123 | Cube | [23] |
| Xoodyak Initialization | | | | | |
| Key-Recovery | 4 | 23 | 23 | DL | [13] |
| | | **21** | **21** | **Cond. HDL** | **Section** 6.2 |
| | 5 | 22 | 22 | DL† | [13] |
| | | **70** | **70** | **Cond. HDL** | **Section** 6.2 |
| Best | 6 | 43.8 | 43.8 | Cond. Cube | [31] |

† This attack is under the related-key model because they obtained the 5-round DL approximation by extending a 4-round one. Our attack is a single-key one, which means we have to choose the input differences from the beginning of 5 rounds.

**Table 3.** Summary of zero-sum attacks on Ascon permutation. We verified them up to 7 rounds by experiments.

| Type | Rnd | Data(log) | Time (log) | Method | Reference |
|---|---|---|---|---|---|
| From Start | 8 | 130 | 130 | Integral | [26] |
| | | **48** | **48** | **HD** | **Section 7** |
| Best | 11 | 315 | 315 | Integral | [26] |
| Inside out | 12 | 130 | 130 | Zero-Sum‡ | [26] |
| | | **55** | **55** | **Zero-Sum** | **Full Version** [14] |

‡Their zero-sum distinguisher can be further extended to a zero-sum partition distinguisher, while ours cannot.

We emphasize that these results do not threaten the security of the Ascon and Xoodyak AEAD schemes.

**Source Code.** We implemented the HATF algorithms in `C++` and DSF in `Python`, the source codes are provided in the git repository https://github.com/hukaisdu/HDL.git.

*Outline.* In Sect. 2, we briefly recall the main concepts of the HD and an algebraic perspective on the differential attack, and other useful background knowledge used in this paper. In Sect. 3, we provide an algebraic perspective on the HD/HDL. The HATF technique is introduced in Sect. 4. In the following sections, we describe the HDL attacks on Ascon and Xoodyak. In Sect. 7, we give the theory and results of DSF. Section 8 concludes this paper.

## 2 Preliminaries

### 2.1 Notations

We use italic lower-case letters such as $x$ to represent elements in $\mathbb{F}_2^n, n \geq 1$. The $j^{th}$ bit of $x$ is denoted by $x[j]$, $0 \leq j < n$, where $x[0]$ is the most significant (the leftmost) bit. The vectors of $\ell$ elements in $\mathbb{F}_2^n$ are denoted by $\boldsymbol{x} = (x_0, x_1, \ldots, x_{\ell-1}) \in (\mathbb{F}_2^n)^\ell$, the $i^{th}$ element of $\boldsymbol{x}$ is denoted by $x_i$ (the $j^{th}$ bit of $x_i$ is then denoted by $x_i[j]$). Given $x \in \mathbb{F}_2$ and $\Delta \in \mathbb{F}_2^n$, $x\Delta = (\Delta[0]x, \Delta[1]x, \ldots, \Delta[n-1]x)$. For $a, b \in \mathbb{F}_2^n$, $a||b \in \mathbb{F}_2^{2n}$ represents the concatenation of $a$ and $b$, $a \cdot b$ stands for the product as $a \cdot b = \bigoplus_{0 \leq i < n} a[i]b[i]$.

In this paper, $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{F}_2^n$ is usually used as symbolic variables. Given $u \in \mathbb{F}_2^n$, $\boldsymbol{x}^u$ is a monomial of $\boldsymbol{x}$ as $\boldsymbol{x}^u = \prod_i x_i^{u[i]}$. For a vectorial Boolean function $E : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we use the $E[0], E[1], \ldots, E[n-1]$ to represent the Boolean functions of its bits.

## 2.2 Boolean Function

An $n$-variable Boolean function is a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2$, which can be uniquely written as its Algebraic Normal Form (ANF) as a multivariate polynomial over $\mathbb{F}_2$ as (note the input $x \in \mathbb{F}_2^n$ of this Boolean function is written as $\boldsymbol{x} \in (\mathbb{F}_2)^n$ to stress that the input can be seen as $n$ bit variables)

$$f(\boldsymbol{x}) = f(x_0, x_1, \ldots, x_{n-1}) = \bigoplus_{u \in \mathbb{F}_2^n} a_u \boldsymbol{x}^u = \bigoplus_{u \in \mathbb{F}_2^n} a_u \prod_{i=0}^{n-1} x_i^{u[i]}, a_u \in \mathbb{F}_2.$$

The algebraic degree of $f$, denoted by $\deg(f)$ is defined as $\max_{a_u \neq 0}\{wt(u)\}$ for all $u \in \mathbb{F}_2^n$ in the above formula. The monomial $x_0 x_1 \cdots x_{n-1}$ is called the *maxterm* of $f$, denoted by $\pi(\boldsymbol{x})$. The coefficient of a monomial $\boldsymbol{x}^u$ of $f$ is denoted by $\mathsf{Coe}(f, \boldsymbol{x}^u)$. Each output bit of a cryptographic primitive can be written as a Boolean function of its public variables (such as plaintexts, initial values (IV), or nonces) and secret variables such as the key bits.

The bias and correlation are two ways of measuring the unbalancedness of an $n$-variable Boolean function $f$. The bias $\varepsilon$ is defined as $\varepsilon = \frac{1}{2^n}|\{f(x) = 0\}| - \frac{1}{2} = \Pr[f = 0] - \frac{1}{2}$ while the correlation $c = \frac{1}{2^n}\sum_{x \in \mathbb{F}_2^n}(-1)^{f(x)}$. Actually, $c = 2\varepsilon$. In this paper, we will only use the bias $\varepsilon$ to measure the unbalancedness.

## 2.3 Algebraic Perspective on DL

In [20], Liu *et al.* introduced a new algebraic method for the differential and DL cryptanalysis as we have already mentioned in Sect. 1. Recalling Eq. 1, the bias of a DL approximation is related to the differential bias of the Boolean function $f = f_{\lambda_O} \circ E$. Thus, to study the DL attack it is enough to focus on the differential property of a sole Boolean function. As explained in Sect. 1, Liu *et al.* proposed Eq. 3 ($f'' = D_x f_{\Delta_I} = \mathcal{D}_{\Delta_I} f$) based on some intuitive observations, but no formal proof nor clear motivation was given in their article. In the next section, we will make it clearer when introducing our algebraic perspective on the $\ell^{th}$-order HD.

**Basic Idea of Algebraic Transitional Forms.** Eq. 3 tells us that if we can (a) calculate the ANF of $D_x f_\Delta$, (b) evaluate the bias of $D_x f_\Delta$, then we can directly know the bias of the output difference. Unfortunately, both tasks are computationally infeasible for modern cryptographic primitives. To overcome these two obstacles, Liu *et al.* introduced the ATF of the exact ANF of $f_\Delta$. ATF of a Boolean function $f$ is a composite representation of $f$, denoted by $\mathcal{A}$. From $\mathcal{A}(f_\Delta)$, we obtain a simpler expression of $D_x f_\Delta$, say $D_x \mathcal{A}(f_\Delta)$, whose bias will be regarded as an estimation of the real bias.

The core of ATF technique is to substitute some parts of a Boolean function with new variables to simplify its form. Finally, $D_x \mathcal{A}(f_\Delta)$ will be a simple formula of intermediate variables (some are variables introduced for substitution). The bias of $D_x \mathcal{A}(f_\Delta)$ is relatively easier to calculate.

In [20], Liu *et al.* proposed two methods to estimate the bias of $D_x\mathcal{A}(f_\Delta)$. Both methods are based on the following Lemma,

**Lemma 1.** **(** *[20]***).** *Given a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and $n$ input bits $x_0, x_1, \ldots, x_{n-1}$ with biases $\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{n-1}$ respectively. Under the assumption that all inputs are independent, the bias of $f$ is*

$$\texttt{Bias}(f) = \sum_{\substack{x_0,x_1,\ldots,x_{n-1} \\ s.t. f(x_0,\ldots,x_{n-1})=0}} \prod_{i=0}^{n-1} \left( \frac{1}{2} + (-1)^{x_i}\varepsilon_i \right) - \frac{1}{2}. \tag{4}$$

Equation 4 is derived from such an idea: the event of $f = 0$ happens means any of the input that makes $f = 0$ happens. The bias of $x_i$ is $\varepsilon_i$, so it happens with probability of $\frac{1}{2} + \varepsilon_i$ when $x_i = 0$ or $\frac{1}{2} - \varepsilon_i$ when $x_i = 1$. Equation 4 follows. When using Eq. 4, we need to find out all inputs that make $f = 0$. Thus the complexity to calculate the bias of $f$ is about $\mathcal{O}(2^n)$.

In the basic method, Liu *et al.* assume that all inputs of $D_x\mathcal{A}(f_\Delta)$ are uniformly random (*i.e.*, the biases of all inputs are exactly 0), the bias of $D_x\mathcal{A}(f_\Delta)$ is computed according to Lemma 1. The improved method is similar to the basic one, but the bias of the intermediate variables will be calculated in advance. Thus, the precision can be improved.

## 3 HD/HDL Cryptanalysis from an Algebraic Perspective

In this section, we give the theory about the $\ell^{th}$ derivative of a Boolean function $f$ from an algebraic perspective. This is a general case of the algebraic perspective on DL proposed in [20]. It is well known that the cube/integral attacks are special cases of HD attacks with all $\ell$ linearly-independent differences being unit vectors. The expression of the HD derivative of $f$ in this case is the coefficient of the so-called cube term [10]. The theory in this section answers such a question: given any $\ell$ linear-independent differences, what is the expression of the HD derivative of $f$?

Given a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ and an $\ell^{th}$-order input difference $\boldsymbol{\Delta} = (\Delta_0, \Delta_1, \ldots, \Delta_{\ell-1}) \in (\mathbb{F}_2^n)^\ell$, the input set is $X \oplus \mathcal{L}(\boldsymbol{\Delta})$ for a certain input $X \in \mathbb{F}_2^n$. The $\ell^{th}$ derivative of $f$ is calculated as

$$\mathcal{D}_{\boldsymbol{\Delta}} f(X) = \bigoplus_{a \in X \oplus \mathcal{L}(\boldsymbol{\Delta})} f(a).$$

Note that $\mathbb{A}^\ell = X \oplus \mathcal{L}(\boldsymbol{\Delta})$ is an $\ell$-dimensional affine space, so we can link $\mathbb{A}^\ell$ to any another $\ell$-dimensional affine space $(\mathbb{A}^\ell)'$ by a bijective mapping $\mathcal{M}$ that sends $(\mathbb{A}^\ell)'$ to $\mathbb{A}^\ell$. Not surprisingly, we tend to choose the simplest $\ell$-dimensional affine space, *i.e.*, the $\ell$-dimensional linear space $\mathbb{F}_2^\ell$. One choice of $\mathcal{M}$ can be

$$\begin{aligned} \mathcal{M} : \mathbb{F}_2^\ell &\to \mathbb{A}^\ell \\ (x_0, x_1, \ldots, x_{\ell-1}) &\mapsto X \oplus x_0\Delta_0 \oplus x_1\Delta_1 \oplus \cdots \oplus x_{\ell-1}\Delta_{\ell-1} \triangleq X \oplus \boldsymbol{x}\boldsymbol{\Delta} \end{aligned} \tag{5}$$

We define a new function $f_{\boldsymbol{\Delta}}$ from $f$ with the transformed input set as[2]:

$$f_{\boldsymbol{\Delta}} : \mathbb{F}_2^{\ell} \rightarrow \mathbb{F}_2, \quad \boldsymbol{x} \mapsto f(X \oplus \boldsymbol{x}\boldsymbol{\Delta}).$$

If we let $D_{\boldsymbol{x}} f_{\boldsymbol{\Delta}}$ represent the coefficient of the maxterm in $f_{\boldsymbol{\Delta}}$, *i.e.*, $D_{\boldsymbol{x}} f_{\boldsymbol{\Delta}} = \mathsf{Coe}\left(f\left(X \oplus \boldsymbol{x}\boldsymbol{\Delta}\right), \pi(\boldsymbol{x})\right)$ (recall that the maxterm is $\pi(\boldsymbol{x}) = \prod_{i=0}^{\ell-1} x_i$), we can give the following formal proposition,

**Proposition 1 (Algebraic-Perspective on HD/HDL).** *Given* $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ *and an* $\ell^{th}$*-order difference* $\boldsymbol{\Delta} \in (\mathbb{F}_2^n)^{\ell}$, $\mathcal{D}_{\boldsymbol{\Delta}} f = D_{\boldsymbol{x}} f_{\boldsymbol{\Delta}}$.

*Proof.* With $\mathcal{M}$ as given in Eq. 5, for any $X$ we have

$$\mathcal{D}_{\boldsymbol{\Delta}} f(X) = \bigoplus_{a \in X \oplus \mathcal{L}(\boldsymbol{\Delta})} f(a) = \bigoplus_{\boldsymbol{x} \in \mathbb{F}_2^{\ell}} f(\mathcal{M}(\boldsymbol{x})) = \bigoplus_{\boldsymbol{x} \in \mathbb{F}_2^{\ell}} f(X \oplus \boldsymbol{x}\boldsymbol{\Delta}).$$

From the perspective of the Möbius transform, the sum over all $\boldsymbol{x} \in \mathbb{F}_2^{\ell}$ is the coefficient of $\pi(\boldsymbol{x})$, *i.e.*,

$$\bigoplus_{\boldsymbol{x} \in \mathbb{F}_2^{\ell}} f(X \oplus \boldsymbol{x}\boldsymbol{\Delta}) = \mathsf{Coe}\left(f(X \oplus \boldsymbol{x}\boldsymbol{\Delta}), \pi(\boldsymbol{x})\right) = D_{\boldsymbol{x}} f_{\boldsymbol{\Delta}}.$$

□

## 4 Estimating HDL Approximation Biases Using HATF

On the basis of Sect. 3, we propose a technique to measure the bias of a probabilistic HDL approximation. The basic idea is inspired by the ATF technique introduced by Liu *et al.* for the DL cryptanalysis [20]: we construct a composite representation of the HDL approximation, then estimate the bias according to the composite representation. In Sect. 4.1, we construct the HATF for a cipher $E$, which is a composite representation of the $\ell^{th}$ derivative of $E$. In Sect. 4.2, we estimate the bias of the $\ell^{th}$-order HDL based on HATF under some reasonable assumptions. In Sect. 4.4, the partitioning technique is introduced to further improve the precision of the HATF method.

### 4.1 Construction of the HATF

According to Proposition 1, if for a Boolean function $f$, we have the ability to calculate the bias of $D_{\boldsymbol{x}} f_{\boldsymbol{\Delta}} = \mathsf{Coe}\left(f(X \oplus \boldsymbol{x}\boldsymbol{\Delta}), \pi(\boldsymbol{x})\right)$, then we will also have the bias of $\mathcal{D}_{\boldsymbol{\Delta}} f$. However, $D_{\boldsymbol{x}} f_{\boldsymbol{\Delta}}$ is too complicated to derive, let alone calculate its bias. Considering that almost all modern ciphers are constructed as a composition of small functions whose ANFs are available, we can represent $D_{\boldsymbol{x}} f_{\boldsymbol{\Delta}}$ in a composite way. Based on the composite representation, it becomes possible to estimate its bias under some assumptions.

[2] Note that $f_{\boldsymbol{\Delta}}$ is a Boolean function of $\boldsymbol{x} = (x_0, x_1, \ldots, x_{\ell-1})$, $X$ and $\boldsymbol{\Delta}$ are regarded as parameters.

Suppose that an $R$-round cipher $E : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is represented as the following composition,

$$E = E_{R-1} \circ E_{R-2} \circ \cdots \circ E_0, \quad E_r : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n, \tag{6}$$

then, according to Proposition 1, to calculate the $\ell^{th}$-order differential of $E$ with the input difference $\boldsymbol{\Delta} = (\Delta_0, \Delta_1, \ldots, \Delta_{\ell-1})$, we can calculate $D_{\boldsymbol{x}} E\left(X \oplus \boldsymbol{x}\boldsymbol{\Delta}\right)$. Here $E$ is a vectorial Boolean function as $E = (E[0], E[1], \ldots, E[n-1])$, so $D_{\boldsymbol{x}} E\left(X \oplus \boldsymbol{x}\boldsymbol{\Delta}\right) = \left(D_{\boldsymbol{x}} E[0]\left(X \oplus \boldsymbol{x}\boldsymbol{\Delta}\right), \ldots, D_{\boldsymbol{x}} E[n-1]\left(X \oplus \boldsymbol{x}\boldsymbol{\Delta}\right)\right)$.

In the following, we will write $X \oplus \boldsymbol{x}\boldsymbol{\Delta}$ in a more general form. Let $e_i$ be the unit vector with only the $i^{th}$ bit being 1 and $\mathbf{0}$ be the vector with all elements being zero, then

$$\alpha_u = \begin{cases} X, & \text{if } u = \mathbf{0} \\ \Delta_i, & \text{if } u = e_i \\ \mathbf{0}, & \text{otherwise} \end{cases},$$

then $X \oplus \boldsymbol{x}\boldsymbol{\Delta} = X \oplus x_0\Delta_0 \oplus x_1\Delta_1 \oplus \cdots \oplus x_{\ell-1}\Delta_{\ell-1}$ can be written in an equivalent form as

$$X \oplus \boldsymbol{x}\boldsymbol{\Delta} = \bigoplus_{u \in \mathbb{F}_2^\ell} \alpha_u \boldsymbol{x}^u, \quad \alpha_u \in \mathbb{F}_2^n.$$

$\boldsymbol{x} = (x_0, x_1, \ldots, x_{\ell-1})$ are $\ell$ symbolic variables in this representation. Hence, the input and output of $E_r$ are both polynomials of $\boldsymbol{x}$ as follows,

$$\bigoplus_{u \in \mathbb{F}_2^\ell} \alpha_u^{(r+1)} \boldsymbol{x}^u = E_r\left(\bigoplus_{u \in \mathbb{F}_2^\ell} \alpha_u^{(r)} \boldsymbol{x}^u\right), \quad \alpha_u^{(r+1)}, \alpha_u^{(r)} \in \mathbb{F}_2^n.$$

Since $\alpha_u^{(r+1)}$ is a vectorial Boolean function with all $\alpha_u^{(r)}$ as input, we derive a new vectorial Boolean function from $E_r$:

$$\mathcal{E}_r^\ell : \left(\mathbb{F}_2^n\right)^{2^\ell} \rightarrow \left(\mathbb{F}_2^n\right)^{2^\ell}, \quad \left(\alpha_u^{(r)}, u \in \mathbb{F}_2^\ell\right) \mapsto \left(\alpha_u^{(r+1)}, u \in \mathbb{F}_2^\ell\right),$$

where

$$\alpha_u^{(r+1)} = f_u\left(\alpha_u^{(r)}, u \in \mathbb{F}_2^n\right) = \mathsf{Coe}\left(E_r\left(\bigoplus_{u \in \mathbb{F}_2^n} \alpha_u^{(r)} \boldsymbol{x}^u\right), \boldsymbol{x}^u\right).$$

Connecting all $\mathcal{E}_r^\ell, 0 \leq r < R$, we derive from $E$ a composite function $\mathcal{E}^\ell$ (an example is illustrated in Fig. 1):

$$\mathcal{E}^\ell = \mathcal{E}_{R-1}^\ell \circ \mathcal{E}_{R-2}^\ell \circ \cdots \circ \mathcal{E}_0^\ell, \quad \mathcal{E}_r^\ell : \left(\mathbb{F}_2^n\right)^{2^\ell} \rightarrow \left(\mathbb{F}_2^n\right)^{2^\ell}. \tag{7}$$

**Definition 1. ($\ell^{\mathbf{th}}$ Higher-order Algebraic Transitional Form ($\ell^{th}$ HATF)).** *The composite function in Eq. 7 above is called the $\ell^{th}$ Higher-order Algebraic Transitional Form ($\ell^{th}$ HATF) of $E$. If the order information is clear from the context, we will omit the superscript $\ell$ for convenience.*

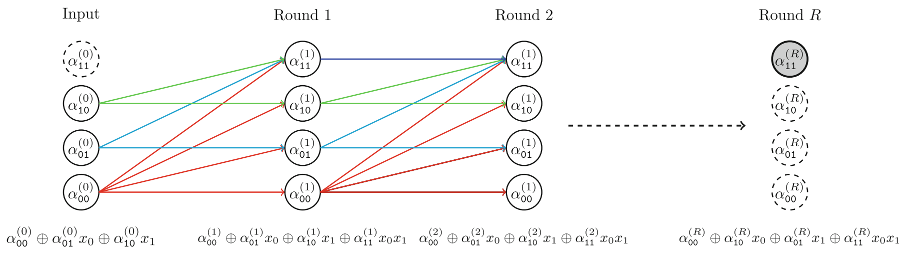

Input
Round 1
Round 2
Round $R$
$\alpha_{00}^{(0)} \oplus \alpha_{01}^{(0)}x_0 \oplus \alpha_{10}^{(0)}x_1$
$\alpha_{00}^{(1)} \oplus \alpha_{01}^{(1)}x_0 \oplus \alpha_{10}^{(1)}x_1 \oplus \alpha_{11}^{(1)}x_0x_1$
$\alpha_{00}^{(2)} \oplus \alpha_{01}^{(2)}x_0 \oplus \alpha_{10}^{(2)}x_1 \oplus \alpha_{11}^{(2)}x_0x_1$
$\alpha_{00}^{(R)} \oplus \alpha_{10}^{(R)}x_0 \oplus \alpha_{01}^{(R)}x_1 \oplus \alpha_{11}^{(R)}x_0x_1$

**Fig. 1.** An illustration of $\mathcal{E}^\ell$ when $\ell = 2$ (2nd HATF). The input is $X \oplus x_0\Delta_0 \oplus x_1\Delta_1 = \alpha_{00}^{(0)} \oplus \alpha_{01}^{(0)}x_0 \oplus \alpha_{10}^{(0)}x_1$. The outputs of the $i^{th}$ round of $\mathcal{E}^\ell$ are the constant monomial and coefficients of $x_0, x_1, x_0x_1$. Finally, the $R$-round HDL bias is just the bias of $\alpha_{11}^{(R)}$.

**Algorithm 1.** Construction of the HATF $\mathcal{E}^\ell$ from a cipher $E$

```
Input: 1. the ANFs of components of $E = E_{R-1} \circ \cdots \circ E_0$,
       2. the order $\ell$,
       3. the block size $n$,
       4. an $\ell^{th}$-order difference $(\Delta_0, \ldots, \Delta_{\ell-1})$,
       5. an input value $X$
Output: the $\ell^{th}$-order HATF $\mathcal{E}^\ell = \mathcal{E}^\ell_{R-1} \circ \cdots \circ \mathcal{E}^\ell_0$
 1: Let $\alpha_{\mathbf{0}}^{(0)} = X$, $\alpha_{e_i}^{(0)} = \Delta_i$, $\Delta_u^{(0)} = 0$ for all $wt(u) \geq 2$
 2: for $0 \leq r < R$ do
 3:     for $0 \leq i < n$ do
 4:         Calculate $f = E_r[i]\left(\bigoplus_{u \in \mathbb{F}_2^n} \alpha_u^{(r)} \boldsymbol{x}^u\right)$
 5:         for $0 \leq u < 2^\ell$ do
 6:             Calculate $\alpha_u^{(r+1)}[i] = \mathsf{Coe}(f, \boldsymbol{x}^u)$
 7:         end for
 8:     end for
 9: end for
10: return $\alpha_u^{(r)}$ for all $1 \leq r \leq R$ and $u \in \mathbb{F}_2^n$, which are actually $\mathcal{E}^\ell$
```

Algorithm 1 shows the detailed process of constructing a HATF. The time complexity of Algorithm 1 is dominated by line 4, *i.e.*, calculating $E_r[i]\left(\bigoplus_{u \in \mathbb{F}_2^n} \alpha_u \boldsymbol{x}^u\right)$. If $\deg(E_r) = d$, then the complexity of calculating $E_r[i]$ is dominated by the calculation of all the $d$-degree monomials in $E_r[i]$. For each $d$-degree monomial, we need to multiply $d$ bits of $\bigoplus_{u \in \mathbb{F}_2^n} \alpha_u \boldsymbol{x}^u$. The complexity of computing a $d$-degree monomial is about $2^{d\ell}$ multiplications and $2^{d\ell}$ additions. Suppose there are $t$ $d$-degree monomials in $E_r[i]$, the time complexity of computing all $d$-degree monomials is about $C_1 = 2 \cdot t \cdot 2^{d\ell}$. Then the complexity of computing an $R$-round cipher is approximately $C_h = 2 \cdot R \cdot n \cdot t \cdot 2^{d\ell}$ multiplications or additions. For a specific cipher, the round $R$, block size $n$, algebraic degree $d$ and the number of $d$-degree monomials are all constants, thus the complexity of constructing the HATF is $\mathcal{O}(2^{d\ell})$.

The main part of memory complexity is to store $\alpha_u^{(r)}[i]$ for every round (line 6 in Algorithm 1). In each $\alpha_u^{(r)}[i]$, there are at most $2^\ell$ terms, so the memory cost is bounded by $\mathcal{O}(2^{2\ell})$.

Next, we introduce a useful property of HATF as Proposition 2.

**Proposition 2.** *Let $\mathcal{E}^\ell$ in Eq. 7 be the HATF of $E$ in Eq. 6. For each $0 \leq r < R$, the algebraic degree of $\mathcal{E}_r^\ell$ is equal to the algebraic degree of $E_r$.*

*Proof.* Let $\deg(E_r) = d$ and consider the output of $\mathcal{E}_r^\ell$. Since

$$\alpha_u^{(r+1)} = \mathsf{Coe}\left(E_r\left(\bigoplus_{u\in\mathbb{F}_2^n} \alpha_u^{(r)}\boldsymbol{x}^u\right), \boldsymbol{x}^u\right),$$

each bit of $\alpha_u^{(r+1)}\boldsymbol{x}^u$ is obtained by multiplying at most $d$ different bits of $\bigoplus_{u\in\mathbb{F}_2^\ell} \alpha_u^{(r)}\boldsymbol{x}^u$. Therefore, the algebraic degree of $\alpha_u^{(r+1)}$ is at most $d$. Finally, when $u = \mathbf{0}$, $\alpha_{\mathbf{0}}^{(r+1)}$ is just the output of $E_r(\alpha_{\mathbf{0}}^{(r)})$, so $\deg(\mathcal{E}_r^\ell) = \deg(E_r)$. □

## 4.2 Estimation of the HDL Bias Based on HATF

Suppose we have obtained the $\ell^{th}$ HATF of a cipher $E = E_{R-1}\circ\cdots\circ E_0$ according to Algorithm 1. The biases of the $\ell^{th}$-order HD/HDL approximations of all the output bits of $E$ are biases of $\alpha_{\mathbf{1}}^{(R)}$ (where $\mathbf{1}$ is the $\ell$-bit vector with all elements being 1). From the HATF of $E$, we know the composite form of $\alpha_{\mathbf{1}}^{(R)}$ is as follows,

$$\left(\alpha_u^{(0)}, u \in \mathbb{F}_2^n\right) \xrightarrow{\mathcal{E}_0} \left(\alpha_u^{(1)}, u \in \mathbb{F}_2^n\right) \xrightarrow{\mathcal{E}_1} \cdots \xrightarrow{\mathcal{E}_{R-2}} \left(\alpha_u^{(R-1)}, u \in \mathbb{F}_2^n\right) \xrightarrow{\mathcal{E}_{R-1}} \alpha_{\mathbf{1}}^{(R)}.$$

Besides, the bias of $a_u^{(0)}, u \in \mathbb{F}_2^n$ is available since they are the input values by adversaries (under a chosen-plaintext attack)[3].

Under the assumption that all the bits of $\alpha_u^{(r)}, u \in \mathbb{F}_2^n$ are independent, the bias of $\alpha_u^{(r+1)}, u \in \mathbb{F}_2^n$ can be estimated according to Lemma 1. Therefore, we can calculate the bias of $\alpha_{\mathbf{1}}^{(R)}$ from $\alpha_u^{(0)}, u \in \mathbb{F}_2^n$ iteratively.

The detailed process is shown in Algorithm 2 with blue words. According to Lemma 1, the time complexity of computing the bias of a Boolean function is exponentially related to the number of variables. For a fixed round $r$ and index $i$, $\alpha_{\mathbf{1}}^{(r+1)}[i]$ has the most number of variables as compared to $\alpha_u^{(r+1)}, u \neq \mathbf{1}$. If the algebraic degree of $\alpha_{\mathbf{1}}^{(r+1)}[i]$ is $d$, then the number of variables in it is at most $d \times 2^\ell$, and the numbers of variables in other $\alpha_u^{(r)}[i], u \neq \mathbf{1}$ are significantly smaller. Therefore the time complexity of line 5 in Algorithm 2 is about $2^{d\times 2^\ell}$. The whole complexity is then approximately $R \cdot n \cdot 2^\ell \cdot 2^{d\times 2^\ell}$, which can be bounded by $\mathcal{O}(2^{\ell+d\times 2^\ell})$. The memory complexity is negligible.

[3] In all attacks of this paper, we simply use uniform $\alpha_u^{(0)}$, *i.e.*, the input values do not have biases.

**Algorithm 2.** Estimate the bias of $\alpha_{\mathbf{1}}^{(R)}$

```
Input: 1. the HATF $\mathcal{E}^\ell = \mathcal{E}^\ell_{R-1} \circ \cdots \circ \mathcal{E}^\ell_0$,
       2. the bias of $\alpha_u^{(0)}[i]$ for all $0 \le i < n$ and $u \in \mathbb{F}_2^n$
Output: the $\ell^{th}$-order HATF $\mathcal{E}^\ell = \mathcal{E}^\ell_{R-1} \circ \cdots \circ \mathcal{E}^\ell_0$
1: for $1 \le r < R$ do
2:     for $0 \le i < n$ do
3:         for $0 \le u < 2^\ell$ do
4:             /* For general cases */
5:             Compute the bias of $\alpha_u^{(r)}[i]$ using Lemma 1
6:             /* For quadratic cases */
7:             Find $M$ so that $\alpha_u^{(r)}[i] = g \circ M^{-1}$ (Lemma 2)
8:             Compute the bias of $M^{-1}\left(\alpha_u^{(r)}, u \in \mathbb{F}_2^n\right)$ (Piling-up lemma [22])
9:             Compute the bias of $\alpha_u^{(r)}[i]$ with $g \circ M^{-1}$ (Lemma 3)
10:        end for
11:    end for
12: end for
13: return the bias of $\alpha_{\mathbf{1}}^{(R)}[i]$ for $0 \le i < n$
```

**Reducing the Complexity for Quadratic Boolean Functions.** Since the complexity of estimating the bias from HATF is $\mathcal{O}(2^{\ell+d\times 2^\ell})$, even a small order will result in high complexity. In the following, we show that for ciphers whose round functions are quadratic, the complexity can be reduced from $\mathcal{O}(2^{\ell+d\times 2^\ell})$ to $\mathcal{O}(2^{3.8\ell})$.

Note that a Boolean function is quadratic if its algebraic degree is 2. A disjoint quadratic Boolean function is defined as follows,

**Definition 2 (Disjoint quadratic Boolean function).** *A quadratic Boolean function is disjoint if all its quadratic monomials do not share any common variables.*

Any quadratic Boolean functions can be decomposed into a disjoint form [15, P438]. In [24], Shi *et al.* applied this method to cryptanalysis of Morus. We omit the detailed algorithm here and only give a small example to show its core idea.

*Example 1.* Let $f = x_0x_1 \oplus x_0x_2 \oplus x_1x_2$. It is not disjoint, but we can convert it to a disjoint Boolean function with the following steps:

1. $f = x_0x_1 \oplus x_0x_2 \oplus x_1x_2 = x_0(x_1 \oplus x_2) \oplus x_1x_2$, we first let $x_1' = x_1 \oplus x_2$ to obtain $g = x_0x_1' + (x_1' \oplus x_2)x_2 = x_0x_1' \oplus x_1'x_2 \oplus x_2$.
2. $g = x_1'(x_0 \oplus x_2) + x_2$, we then let $x_0' = x_0 \oplus x_2$ and obtain $g = x_1'x_0' \oplus x_2$, then $g$ is a disjoint quadratic Boolean function.

During the process, we do linear variable substitutions with $x_1' = x_1 \oplus x_2$ and $x_0' = x_0 \oplus x_2$. For sake of convenience, we let $x_2' = x_2$, so $g = f(M[x_0, x_1, x_2]^t)$ where $M = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$. Equivalently, $f = g \circ M^{-1}$.

We write this method as a lemma for a convenient citation.

**Lemma 2.** ( [15]). *A quadratic Boolean function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ can be converted into a disjoint Boolean function $g : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ with $g = f \circ M$ where $M \in \mathbb{F}_2^{n \times n}$ is an invertible matrix. The time complexity is $\mathcal{O}(n^{3.8})$ and the memory complexity is $\Omega(n^2)$.*

The bias of a disjoint quadratic Boolean function can be computed with ease, as shown in Lemma 3.

**Lemma 3.** *Let $g : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ be a disjoint quadratic Boolean function as*

$$g = g_0 \oplus g_1 \oplus \cdots \oplus g_{T-1}$$

*where all $g_i, 0 \leq i < T$ do not share common variables, and the biases of all input variables of $g$ are available. Then we can compute the bias of each $g_i$ using Lemma 1 with small complexities. Finally, the bias of $g$ can be computed with the piling-up lemma with biases of all $g_i$.*

According to Proposition 2, when a cipher uses quadratic round functions, the round functions of its HATF is also quadratic. For calculating the bias of $\alpha_u^{(r+1)}[i]$ (line 5 of Algorithm 2) from $\alpha_u^{(r)}$, we first find an invertible matrix $M$ such that $g = \alpha_u^{(r+1)}[i] \circ M$ is disjoint according to Lemma 2. Equivalently, $\alpha_u^{(r+1)}[i] = g \circ M^{-1}(\alpha_u^{(r)}, u \in \mathbb{F}_2^n)$.[4] Based on the biases of $\alpha_u^{(r)}$, the bias of $M^{-1}\left(\alpha_u^{(r)}, u \in \mathbb{F}_2^n\right)$ can be calculated using the piling-up lemma. Applying the disjoint Boolean function $g$ to the output of $M^{-1}\left(\alpha_u^{(r)}, u \in \mathbb{F}_2^n\right)$, the bias of $\alpha_u^{(r+1)}[i]$ can be obtained according to Lemma 3.

The process is also given in Algorithm 2, but with red words. The complexity is dominated by line 7, *i.e.*, converting $\alpha_u^{(r+1)}[i]$ to a disjoint quadratic form. Since the number of variables in $\alpha_u^{(r+1)}[i]$ is at most $2 \times 2^\ell$ (we are working on quadratic functions), the time complexity of line 7 is $\mathcal{O}(2^{3.8\ell})$, and the memory complexity is $\Omega(2^{2\ell})$.

Considering both Algorithms 1 and 2, the time complexity of computing the biases of the $\ell^{th}$ HDL approximations is $\mathcal{O}(2^{\ell+d\times 2^\ell})$ in general case, and $\mathcal{O}(2^{3.8\ell})$ for ciphers with quadratic round functions. The memory complexity is $\Omega(2^{2\ell})$.

## 4.3 Discussion on the Assumption of Independence and Precision

HATF works on the assumption that all bits of $\alpha_u^{(r)}, 0 \leq r < R, u \in \mathbb{F}_2^n$ are independent. If we directly use Algorithm 1 to construct the HATF, many related $\alpha_u^{(r+1)}[i]$ will be regarded as independent variables (line 6 of Algorithm 1), which makes our assumption less valid. Thus, we need to avoid such cases as much as possible. Methods that we use to avoid these related variables are introduced as follows, both of which are concerned about line 6 only

[4] Note that not all bits in $\alpha_u^{(r)}$, $u \in \mathbb{F}_2^n$ are input of $g \circ M^{-1}$. We write it in this way for convenience.

1. When $\deg(\alpha_u^{(r+1)}[i]) \leq 1$, we do not introduce a new variable $\alpha_u^{(r+1)}[i]$ to substitute $\mathsf{Coe}\,(f, \boldsymbol{x}^u)$, because variables in linear expressions are easier to be related with other variables. In this case, the following computation will depend on $\mathsf{Coe}\,(f, \boldsymbol{x}^u)$ directly rather than $\alpha_u^{(r+1)}[i]$.
2. We use a dictionary $Q$ to store each variable substitution as
$$Q[\mathsf{Coe}\,(f, \boldsymbol{x}^u)] = \alpha_u^{(r+1)}[i],$$
then if $\mathsf{Coe}\,(f, \boldsymbol{x}^u)$ or $\mathsf{Coe}\,(f, \boldsymbol{x}^u) \oplus 1$ has been in $Q$, we do not need to introduce new variables, $\alpha_u^{(r+1)}[i]$ or $\alpha_u^{(r+1)}[i] \oplus 1$ can be reused.

By these two methods, we can avoid most simple related-bit cases. Other kinds of relations are relatively more complicated and are not considered in this paper. We hope that those bits with complicated relationships can be approximately treated as independent bits.

In terms of the time/memory complexities, the first method increases the number of variables linearly but does not affect its order of magnitude; the second method saves the number of new variables, so it actually reduces the complexity of Algorithm 2. Hence, the time/memory complexities of HATF remain unchanged up to the $\mathcal{O}/\Omega$ notations.

### 4.4 Improving the Precision with Partitioning Technique

As the order and rounds increase, the HATF systems become more and more complicated. The precision of HATF for complicated Boolean functions according to Lemma 1 or Lemma 3 drops accordingly. To mitigate the imprecision, we partition the whole input space (in the HDL distinguishers, the input includes both the public and secret variables) into several smaller *equal-size* subsets. Given a cipher $E$, suppose we have partitioned the whole input $S$ into $\kappa$ disjoint subsets as $S = S_0 + S_1 + \cdots + S_{\kappa-1}$ (we use "+" to stress that $S_i$ are disjoint) where $S_i, 0 \leq i < \kappa$ have the same size. The bias of $\mathsf{Coe}\,(\mathcal{E}, \boldsymbol{\pi}(\boldsymbol{x}))$ is the average value among the $S$. Note that $\mathsf{Coe}\,(\mathcal{E}, \boldsymbol{\pi}(\boldsymbol{x}))$ is a Boolean function of values in $S$ (recall Sect. 4.1). Let $|S_i| = w$ for $0 \leq i < \kappa$ and $|S| = \kappa w$, we have

$$\begin{aligned}
\mathtt{Bias}\,(\mathsf{Coe}\,(\mathcal{E}, \boldsymbol{\pi}(\boldsymbol{x}))) &= \frac{\#\{\mathsf{Coe}\,(\mathcal{E}, \boldsymbol{\pi}(\boldsymbol{x}))\,(s) = 0 : s \in S\}}{\kappa w} - \frac{1}{2} \\
&= \frac{\sum_{i=0}^{\kappa-1} \#\{\mathsf{Coe}\,(\mathcal{E}_{S_i}, \boldsymbol{\pi}(\boldsymbol{x}))\,(s') = 0 : s' \in S_i\}}{\kappa w} - \frac{1}{2} \\
&= \frac{1}{\kappa} \sum_{i=0}^{\kappa-1} \frac{\#\{\mathsf{Coe}\,(\mathcal{E}_{S_i}, \boldsymbol{\pi}(\boldsymbol{x}))\,(s') = 0 : s' \in S_i\}}{w} - \frac{1}{2} \\
&= \frac{1}{\kappa} \sum_{i=0}^{\kappa-1} \left( \frac{\#\{\mathsf{Coe}\,(\mathcal{E}_{S_i}, \boldsymbol{\pi}(\boldsymbol{x}))\,(s') = 0 : s' \in S_i\}}{w} - \frac{1}{2} \right) \\
&= \frac{1}{\kappa} \sum_{i=0}^{\kappa-1} \left( \mathtt{Bias}\,(\mathsf{Coe}\,(\mathcal{E}_{S_i}, \boldsymbol{\pi}(\boldsymbol{x}))) \right),
\end{aligned} \tag{8}$$

where $\mathcal{E}_{S_i}$ is the HATF of $E$ with the subset $S_i$ as the input. As a result, for each of these subsets, we first apply our HATF technique to evaluate the bias for each $S_i$, then calculate the average bias among all these $S_i$. The partition simplifies the ANFs of $\mathcal{E}^\ell$, so for each subset, the precision can be improved. The methods of partitioning the input are chosen in different ways for different ciphers, which will be described in our applications.

### 4.5 Conditional HDL Cryptanalysis by Injecting Conditions

In [20], to improve the biases of DL approximations, Liu *et al.* imposed some conditions to the first $R_0$ rounds in the ATF. The basic principle is to zero the differences in the first $R_0$ rounds as much as possible. In our HDL attacks based on HATF, we can also use this method to obtain a set of conditions to improve the HDL biases.

In the construction of the first $R_0$-round HATF, we put the first non-constant $\alpha_u^{(r)}, r \leq r_0, u \neq \mathbf{0}$ into a set $I$ as ideal generators. Next, we reduce all the $\alpha_u^{(r)}$ over the ideal generated from $I$, denoted by "mod $I$". If a certain $\alpha_u^{(r)}$ cannot be reduced to a constant, we will add this $\alpha_u^{(r)}$ into $I$ and use the updated $I$ to reduce the remaining non-constant $\alpha_u^{(r)}$. Finally, all $\alpha_u^{(r)}, u \neq \mathbf{0}, r \leq r_0$ are usually reduced to constants, and a system of equations $S = \{f = 0 \mid f \in I\}$ is obtained. When the conditions in $I$ are satisfied, the HDL distinguisher will have a significantly higher bias. By checking these conditions, we can recover the secret keys. These conditions are also used for partitioning the input space.

## 5 Applications to Ascon Initialization

Ascon, designed by Dobraunig, Eichlseder, Mendel, and Schläffer, is a family of AEAD and hash algorithms [12]. It has been selected as the winner in the NIST Lightweight Cryptography competition. Due to page limits, the description of the Ascon AEAD and its permutation is provided in [14], we also recommend that readers refer to [12] for the whole specification.

**Notations used for describing the Ascon initialization.** For the Ascon initialization, the 320-bit output state after $r$ rounds is denoted by

$$S^{(r)} = S^{(r)}[0] \| S^{(r)}[1] \| S^{(r)}[2] \| S^{(r)}[3] \| S^{(r)}[4],$$

where $S^{(r)}[i]$ is the $i^{th}$ word (the $i^{th}$ row) of $S^{(r)}$; $S^{(0)}$ is the input of the whole permutation. The $j^{th}$ bit of $S^{(r)}[i]$ is denoted by $S^{(r)}[i][j]$ where $0 \leq i < 5, 0 \leq j < 64$. $S^{(r)}[0][0]$ is the leftmost bit of the first row of the state matrix $S^{(r)}$. Let $p_C, p_S, p_L$ represent the operations of *addition of constants*, *substitution layer*, *linear diffusion layer*, respectively. Then $S^{(r)} = (p_L \circ p_S \circ p_C)^r(S^{(0)})$. The adversary can only access the first word of the output state for Ascon-128, and the first two words for Ascon-128a (our cryptanalysis focuses on Ascon-128, so it is also applicable to Ascon-128a). Since the linear layer is applied to each row, we do not consider the linear layer of the last round.

Since $p_S$ is quadratic, the time complexity for an $\ell^{th}$-order HDL cryptanalysis of Ascon is $\mathcal{O}(2^{3.8\ell})$. To apply the HATF technique, we need to decompose the

$R$-round Ascon initialization into several small parts. In this paper, we take the same method to cut the Ascon functions as [20][5]. Firstly, we divide the Sbox of Ascon into two parts, $p_{S_L}$ and $p_{S_N}$. The first part of the Sbox, $p_{S_L}$, is a linear operation

$$x_0 = x_0 \oplus x_4; \qquad x_4 = x_4 \oplus x_3; \qquad x_2 = x_2 \oplus x_1;$$

where $(x_0, x_1, x_2, x_3, x_4)$ is the input of $p_{S_L}$. The round function of the Ascon permutation is then divided into two parts, $p_A = p_{S_L} \circ p_{P_C}$ and $p_B = p_L \circ p_{S_N}$.

In Algorithm 1, we let $E^{(0)} = p_A$, and $E^{(r)} = p_A \circ p_B$ for $1 \leq r < R$, and $E^{(R)} = p_{S_N}$. Thus $R$-round Ascon is represented as

$$E = p_{S_N} \circ (p_A \circ p_B)^{R-1} \circ p_A$$

The 128-bit key and 128-bit nonce are set to 256 binary variables, the IV is set to the constant specified in [12].

When applying the $\ell^{th}$-order HDL distinguishing attack on the Ascon initialization, we choose $\Delta_j, 0 \leq j < \ell$ as the $\ell$ linearly-independent differences, where $\Delta_j$ is active in the two nonce bits of the same Sbox, *i.e.*, $S^{(0)}[3][i_j]$ and $S^{(0)}[4][i_j]$ $(0 \leq i_j < 64)$. (We also tested other kinds of differences, but this setting brings the best results.) Then the input difference can be denoted by an $\ell$-tuple, denoted by $\Delta(i_0, i_1, \ldots, i_{\ell-1})$. To simplify the ANFs, we by default always set $S^{(0)}[3][i_j] = S^{(0)}[4][i_j] = 0$. For $R$-round outputs, we consider the single-bit bias of the first word, *i.e.*, $S^{(R)}[0][i], 0 \leq i < 64$. We choose such input differences because the input of Ascon comes into Sboxes directly and our choices of input can simplify the ANFs.

## 5.1 HDL Distinguishers for Ascon

**Application 1: Revisiting the first-order DL distinguishers for 4- and 5-round** Ascon. Our first application is to revisit two DL distinguishers on the 4- and 5-round initialization of Ascon. These two DL distinguishers were first found by the designers in [11] with experiments. The input difference was set as $\Delta(0)$. Although the classical DL attack theory predicted that the 4-round distinguisher has a bias of $2^{-20}$, experiments showed that its real bias is about $2^{-2}$ which is significantly higher. Later, at EUROCRYPT 2019, Bar-On *et al.* [1] revisited this distinguisher and used the Differential-Linear Connectivity Table (DLCT) technique to give a higher theoretical estimation of $2^{-5}$. Recently, at CRYPTO 2021, Liu *et al.* used the ATF to improve the theoretical bias to $2^{-2.36}$, which is the most precise value before this paper. However, none of the three methods can find any 5-round DL distinguisher.

Our HATF technique is a higher-order extension of the ATF technique, so it is also applicable to the first-order DL attack. With the partitioning technique, we achieved better estimation. Setting the input difference as $\Delta(0)$, the two key bits in the same Sbox, *i.e.*, $S^{(0)}[1][0]$ and $S^{(0)}[2][0]$, are chosen to partition the

[5] Our experiments show such cutting can lead to slightly better results compared to the cutting method according to the rounds, in the case of HATF.

input space. Let $S^{(0)}[1][0]||S^{(0)}[2][0]$ be `00`, `01`, `10` and `10`, we partition the input subspace to 4 equal-size subspaces. The bias of $S^{(4)}[0][54]$ is then

$$\texttt{Bias}(S^{(4)}[0][54]) = \begin{cases} 2^{-2.678}, & \text{when } (S^{(0)}[1][0], S^{(0)}[2][0]) = \texttt{00} \\ 2^{-2.678}, & \text{when } (S^{(0)}[1][0], S^{(0)}[2][0]) = \texttt{01} \\ 2^{-1.678}, & \text{when } (S^{(0)}[1][0], S^{(0)}[2][0]) = \texttt{10} \\ 2^{-1.678}, & \text{when } (S^{(0)}[1][0], S^{(0)}[2][0]) = \texttt{11} \end{cases}$$

According to the partitioning technique and Eq. 8,

$$\texttt{Bias}(S^{(4)}[0][54]) = 2^{-2}(2^{-2.678} + 2^{-2.678} + 2^{-1.678} + 2^{-1.678}) \approx 2^{-2.09}.$$

This theoretical bias is again closer to the experimental bias $2^{-2}$.

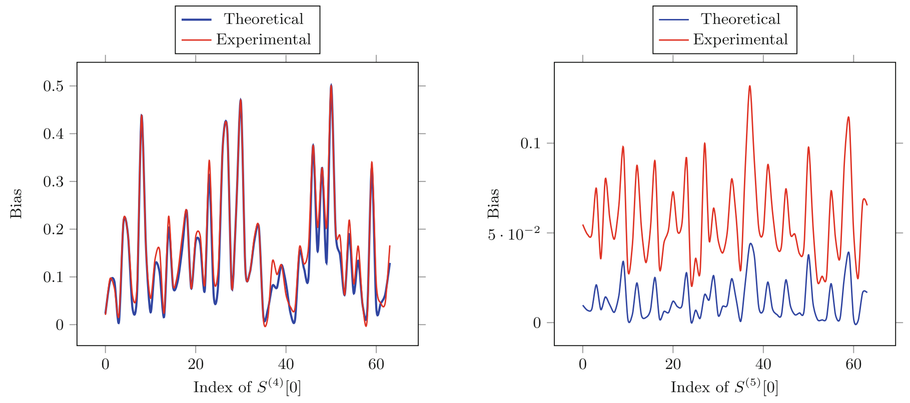
For 5-round Ascon, the known DL distinguisher is also with the input difference $\Delta(0)$, the bias of $S^{(5)}[0][47]$ is about $2^{-9}$.[6] With the above partition, the bias from the HATF is always 0, hence we need to partition the space into smaller ones to detect the bias. According to Sect. 4.5, we can derive a set of 7 conditions that affect the bias significantly. Since the 7 conditions are all balanced Boolean functions, by assigning all possible values to them (every Boolean function then has two statuses: true or false), we can partition the whole space into 128 subspaces[7]. Computing HATF for every individual subspace, we obtain the average bias of approximately $2^{-10}$. This is the first theoretical method that can predict this 5-round DL bias.

**Application 2: $2^{nd}$-order HDL distinguisher for 4-round** Ascon. Our second application is the $2^{nd}$-order DL distinguisher for 4-round Ascon initialization. We exhaustively search through all possible $\Delta(0, i), 1 \leq i < 64$ as our $2^{nd}$-order differences, all such differences lead to highly biased 4-round output bits. Especially, when $(i, j) = (0, 60)$, the bias of $S^{(4)}[0][50]$ is $\frac{1}{2}$, *i.e.*, this is a deterministic $2^{nd}$-order DL bias. With $2^{26}$ randomly chosen samples, this deterministic distinguisher is fully verified. We plot the theoretical and experimental biases of the 64 bits of $S^{(4)}[0]$ as shown in Fig. 2a, and the concrete data is provided in the full version. The theoretical biases are very close to the experimental ones. According to these $2^{nd}$-order HDL biases, one sample, *i.e.*, $2^2$ chosen nonces is enough to distinguish the 4-round initialization.

**Application 3: $2^{nd}$-order HDL distinguisher for 5-round** Ascon. In our $2^{nd}$-order HDL distinguishing attack on the 5-round Ascon initialization, we also exhausted all possible $\Delta(0, i), 1 \leq i < 64$ differences and checked every single bit output of $S^{(5)}[0]$, the most significant bias is $S^{(5)}[0][50]$ when $(i, j) = (0, 3)$ which is predicted to be $2^{-7.05}$ by HATF. We use $2^{26}$ samples to check this bias and find that it should be $2^{-6.60}$ approximately, which is slightly larger but still considerably close to our prediction.

[6] Under the default setting that $S^{(0)}[3][0] = S^{(0)}[4][0]$, see [11] for more information about this DL distinguisher.

[7] We also encourage readers to read our code to further understand how we use these conditions: https://github.com/hukaisdu/HDL/blob/main/HATF/ascon.cpp.



(a) 2nd-order HDL for 4-R Ascon (b) 8th-order HDL for 5-R Ascon

**Fig. 2.** Theoretical and experimental biases for 4-round and 5-round Ascon

**Application 4: 8th-order HDL distinguisher for 5-round** Ascon. Generally speaking, as the order increases, the biases become more and more significant according to HATF. Here we give the results of the 8th-order HDL distinguishing attack on the 5-round Ascon initialization. We randomly select 8 indexes $(i_0, i_1, \ldots, i_7) = (0, 8, 9, 13, 14, 26, 43, 60)$ as the 8th-order input differences $\Delta(0, 8, 9, 13, 14, 26, 43, 60)$. The 16 key bits in the same Sboxes with the input differences are used to partition the input space into $2^{16}$ subspaces. Applying HATF to each of the subspaces, and calculating the average bias, we find all single bits are highly biased. For example, HATF predicts that $\texttt{Bias}(S^{(5)}[0][50]) = 2^{-4.73}$. With $2^{22}$ samples, experiments show that this bias is about $2^{-3.35}$. The average bias over the 64 output bits is predicted as $2^{-6.34}$, and the experimental result is $2^{-4.11}$. The theoretical and experimental biases of all 64 bits of $S^{(5)}[0]$ are shown in Fig. 2b, the concrete biases are provided in the full version.

We use this 8th-order HDL approximation to mount the best distinguishing attack on 5-round Ascon. Suppose that we encrypt $N$ samples, we can observe $64N$ output bits in total. Regarding each bit of $S^{(5)}[0]$ as a Bernoulli experiment with expectation of $\frac{1}{2} + 2^{-4.11}$, The number of occurrences of 0 conforms to the binomial distribution $\mathcal{B}(64N, \frac{1}{2}+2^{-4.11})$, which can be approximated by a normal distribution $\mathcal{N}(35.71N, 15.79N)$ for a convenient analysis. In a random case, the number of occurrences of 0 conforms to another binomial distribution $\mathcal{B}(64N, \frac{1}{2})$, which can be approximated by $\mathcal{N}(32N, 16N)$. The method to distinguish two normal distributions has been well-known in cryptanalysis, which is summarized in our full vesion. Setting that the probabilities for Type-I and Type-II errors to be $\alpha_0 = \alpha_1 = 0.05$, *i.e.*, we require 95% success rate, we have $N \approx 3^{3.65}$, *i.e.*, we need to check about 801 output bits. The threshold is $\tau \approx 424$. The time complexity is about $2^{11.65}$.

We can mount a distinguishing attack as follows,

1. Encrypt a total of 13 samples, for the previous 12 samples, we count the number of occurrences of 0 in all 64 bits of $S^{(5)}[0]$; for the 13th sample, we only count the number of occurrences of 0 in $S^{(5)}[0][j], 0 \leq j < 33$. As a whole, we count 801 bits. Denote the number of occurrences of 0 by $T$,
2. If $T \geq 423$, the target is the 5-round Ascon initialization; otherwise, the target is a random function.

We did 1000 experiments, and about 900 experiments were successful. The reason for the gap between the theoretical and experimental success rates might be that the independent assumptions are not always true.

## 5.2 Conditional HDL Attack for Ascon

**Application 5: 2nd-order HDL key-recovery attack on 5-round** Ascon. Thanks to the higher bias of the HDL approximations, generally speaking, we can mount key-recovery attacks more efficiently than DL attacks. In this paper, we use several 2nd-order HDL approximations to recover the secret keys from 5-round Ascon, which is the most efficient attack for 5-round Ascon thus far. The idea of this key-recovery attack is similar to the conditional DL attacks introduced in [20]. When applying the HATF, we inject the conditions for the first two rounds according to Sect. 4.5. By exhausting all possible 2nd-order differences $\Delta(i,j)$, all of them lead to at least one highly biased bit. The two most significant ones are listed as follows (readers can use our code to generate all of them),

1. When $\Delta(i,j) = \Delta(i', i'+9), 0 \leq i' < 64$, under 14 conditions, $\texttt{Bias}(S^{(5)}[0][27+i']) = 0.375$,
2. When $\Delta(i,j) = \Delta(i', i'+24), 0 \leq i' < 64$, under 16 conditions, $\texttt{Bias}(S^{(5)}[0][51+i']) = 0.313$.

Experiments with $2^{26}$ samples have fully verified these biased bits. With these two 2nd-order approximations, we can do the key-recovery attack with about $2^{22}$ computations. Since this attack is similar to the key-recovery attack on Xoodyak, we provide all the details in the full version of this paper.

**Application 6: 3rd-order HDL approximation for 6-round** Ascon. At the end of this section, we present a conditional 3rd-order HDL approximation for 6-round Ascon initialization. In [20], Liu *et al.* showed that there are no conditional DL approximations for 6-round Ascon initialization. As the order increases, it is not surprising that there are truly some HDL approximations for 6 (or even more) rounds. However, from the 5-round to the 6-round, the complexity required to find a highly biased approximation become significantly larger. In our conditional 3rd-order HDL approximation, we inject 24 conditions into the first two rounds of the HATF. The input difference is $\Delta(0,30,61)$, the bias occurs in $S^{(6)}[0][34]$ and is predicted as $2^{-25.97}$. It is difficult to verify this bias with experiments directly. However, since $S^{(6)}[0][34]$ is the output bit of the Sbox in the 6th round, we can verify the bias of bits in $S^{(5)}$. According to the linear approximation table (LAT) of Ascon's Sbox, there is a mask propagation 0x3 → 0x10 with a bias of $-2^{-2}$. Thus, $S^{(5)}[3][34] \oplus S^{(5)}[4][34]$ (the two bits are inputs of the Sbox related to $S^{(6)}[0][34]$) may have a high bias. We use $2^{30}$ samples to test it, the bias of $S^{(5)}[3][34] \oplus S^{(5)}[4][34]$ is about $2^{-14}$. Considering the piling-up lemma [22] and we have 8 approximations, the bias of $S^{(6)}[0][34]$ should be around $2^{-22}$. It means that the 6-round conditional $3^{rd}$ HDL approximation is true.

## 6 Applications to Xoodyak Initialization and Xoodoo

Xoodyak is a cryptographic primitive for hashing, authenticated encryption, and MAC computation, and is one of the ten finalists of the NIST LWC competition [9]. Xoodyak uses Xoodoo as its underlying cryptographic permutation, which is a family of 384-bit to 384-bit permutations [8]. The 384-bit state of Xoodoo is arranged into a $4 \times 3 \times 32$ cube and a state bit is denoted by $S[x][y][z]$. When $x$ and $z$ are fixed, the three bits of $S[x][\cdot][z]$ are called a column; when $y$ is fixed, the 128 bits of $S[\cdot][y][\cdot]$ are called a plane. The input and output states of the $r^{th}$ round are denoted by $S^{(r-1)}$ and $S^{(r)}$, respectively. The initial state is then denoted by $S^{(0)}$. One round of Xoodoo consists of five operations as $\rho_{east} \circ \chi \circ \iota \circ \rho_{west} \circ \theta$.

$$\begin{aligned}
\theta:&\quad S[x][y][z] = S[x][y][z] \oplus \bigoplus_y S[x-1][y][z-5] \oplus \bigoplus_y S[x-1][y][z-14]\\
\rho_{west}:&\quad S[x][1][z] = S[x-1][1][z], S[x][2][z] = S[x][2][z-11]\\
\iota:&\quad S[0][0] = S[0][0] \oplus RC_i\\
\chi:&\quad S[x][y][z] = S[x][y][z] \oplus ((S[x][y+1][z] \oplus 1) \cdot S[x][y+2][z])\\
\rho_{east}:&\quad S[x][1][z] = S[x][1][z-1], S[x][2][z] = S[x-2][2][z-8]
\end{aligned}$$

$RC_i$ in the $\iota$ operation is the $i$-round constant, which can be found in [8]. Note that $\chi$ is a quadratic function. Xoodyak AEAD supports three methods to handle the nonces. This paper focuses on the third method's initialization. In this mode, the 128-bit state of $S^{(0)}[x][0][z], 0 \le x < 4, 0 \le z < 32$ are initialized by an 128-bit key, denoted by $k_i$ where $i = z + 32x$, and the remaining 256 bits of $S^{(0)}[x][y][z], 0 \le x < 4, 1 \le y < 3, 0 \le z < 32$ by a 256-bit nonce, denoted by $u_i$ where $i = z + 32(x + 4(y-1))$. Then, Xoodoo is applied to the initialized state, and the first 192 bits are visible and XORed to the first block of the plaintext. The following HDL approximations for Xoodyak and Xoodoo are found mainly by trying all low-weight difference-mask pairs.

### 6.1 HDL Distinguishers for Xoodyak and Xoodoo

**Application 1: Revisiting the DL Distinguishers for 4-round** Xoodyak. In [13], Dunkelman and Weizman gave the first DL attacks on 4-round Xoodyak under the single-key model and on 5-round Xoodyak under the related-key model. The two distinguishers used in the attacks are detected by experiments. HATF with the partitioning technique can easily give the theoretical biases for the two distinguishers.

For the 4 rounds, the input difference is in $(u_0, u_{128})$ (*i.e.*, $S^{(0)}[0][1][0]$ and $S^{(0)}[0][2][0]$), and the output bit of $S^{(4)}[0][1][15]$ has a bias of about $2^{-9.7}$. Applying our HATF technique to the 4-round Xoodyak, we first obtain a set of 4 conditions that are injected into the first round to zero all the differences after the first round according to Sect. 4.5. These 4 conditions are listed as follows,

$$\begin{aligned} u_{102} &= k_{11} \oplus k_{102} \oplus k_{125} \oplus u_{125} \oplus u_{230} \oplus u_{253} \\ u_{70} &= k_{70} \oplus k_{93} \oplus u_{93} \oplus u_{107} \oplus u_{198} \oplus u_{221} \oplus 1 \\ u_7 &= k_7 \oplus k_{16} \oplus u_{16} \oplus u_{135} \oplus u_{144} \oplus u_{181} \\ u_{18} &= k_{18} \oplus k_{27} \oplus k_{32} \oplus u_{27} \oplus u_{146} \oplus u_{155} \oplus 1 \end{aligned}$$

Since these 4 conditions are all linear and independent, they can partition the whole input space into 16 subspaces by assigning all possible values to them. After applying the first-order HATF technique, the bias of $S^{(4)}[0][1][15]$ is

$$\texttt{Bias}(S^{(4)}[0][1][15]) = 2^{-9.67},$$

which is very close to the experimental results.

In the related-key DL attack on the 5-round Xoodyak, Dunkelman and Weizman used another 4-round DL distinguisher where the input difference is in $(k_0, u_{128})$ (*i.e.*, $S^{(0)}[0][0][0]$ and $S^{(0)}[0][2][0]$) and the output bias of $S^{(4)}[0][0][0]$ is about $-2^{-5.36}$ [13]. Again, this distinguisher was obtained by experiments. To apply HATF to it, we also obtain 4 equations by injecting conditions into the first round,

$$\begin{aligned} u_{103} &= k_{103} \oplus k_{112} \oplus u_{112} \oplus u_{149} \oplus u_{231} \oplus u_{240} \oplus 1 \\ u_{70} &= k_{70} \oplus k_{93} \oplus u_{93} \oplus u_{107} \oplus u_{198} \oplus u_{221} \oplus 1 \\ u_{102} &= k_{11} \oplus k_{102} \oplus k_{125} \oplus u_{125} \oplus u_{230} \oplus u_{253} \\ u_{82} &= k_{82} \oplus k_{91} \oplus u_{91} \oplus u_{96} \oplus u_{210} \oplus u_{219} \end{aligned}$$

Another time, we obtain 16 subspaces. After applying the first-order HATF technique to each subspace, the bias of $S^{(4)}[0][0][0]$ is

$$\texttt{Bias}(S^{(4)}[0][0][0]) = -2^{-6},$$

which is very close to the experimental results.

**Application 2: $2^{nd}$-order HDL distinguisher for 4-round** Xoodyak. In our $2^{nd}$-order distinguisher, we choose the two differences as $\Delta_0$ that is active in $(u_0, u_{128})$, $\Delta_1$ that is active in $(u_{47}, u_{175})$. After 4-round Xoodyak initialization, our HATF technique shows that the bias of $S^{(4)}[0][0][12]$ is about $0.019 \approx 2^{-5.72}$. Experiments with $2^{26}$ randomly-selected samples show the real bias is also approximately $2^{-5.72}$.

**Application 3: 4th-order HDL distinguisher for 4-round** XOODYAK. Unlike ASCON, the nonlinear function of XOODOO ($\chi$) is after the linear operation. Hence if we select low-weight input differences before $\theta$, the input differences into $\chi$ are complicated. Thus, we also tried selecting low-weight differences before $\chi$, then we can compute the actual differences back through $(\theta \circ \rho_{west} \circ \iota)^{-1}$. In our 4th-order distinguisher, we choose four differences such as (Let $S$ be the input state): $\varDelta_0$ is active in $(S[0][0][0], S[0][2][0])$, $\varDelta_1$ is active in $(S[2][0][7], S[2][2][7])$, $\varDelta_2$ is active in $(S[2][0][15], S[2][2][15])$, and $\varDelta_3$ is active in $(S[3][0][27], S[3][2][27])$. After 3.5 rounds of XOODYAK initialization ($\rho_{east} \circ \chi$ of the first round and the remaining three full rounds), our HATF technique shows that the bias of $S^{(4)}[0][1][1]$ is $2^{-1}$. Thus, when the input difference of the 4-round XOODYAK is then $(\theta \circ \rho_{west} \circ \iota)^{-1}(\varDelta_0, \varDelta_1, \varDelta_2, \varDelta_3)$, the bias of $S^{(4)}[0][1][1]$ is $2^{-1}$. Note that $(\theta \circ \rho_{west} \circ \iota)^{-1}(\varDelta_0, \varDelta_1, \varDelta_2, \varDelta_3)$ will be active in all three planes, so this HDL distinguisher is under the related-key model.

**Application 4: 2nd-order HDL distinguisher for 5-round** XOODYAK. It becomes very difficult to detect useful HDL approximations for 5-round XOODYAK under the single-key model, we exhaust all possible 2nd-order differences that are active in $(u_0, u_{128})$ and $(u_j, u_{j+128})$. If we do not inject any conditions, the biases of all output bits from HATF are all 0. Hence, we first inject 8 conditions according to Sect. 4.5. When the input difference is active in $(u_0, u_{128})$ and $(u_{34}, u_{162})$, the highest bias occurs in $S^{(5)}[0][0][20]$ which is $2^{-37}$. We then tried all 256 possibilities of the 8 conditions and found the average bias to be $2^{-45}$.

**Application 5: 2nd- and 3rd-order HDL distinguishers for 4- and 5-round** XOODOO. Besides XOODYAK, XOODOO also plays an important role in other schemes such as XOOFFF [8]. Thus, it is also interesting to see if there are some HDL distinguishers for XOODOO. Since XOODOO is a public permutation, we do not need to consider the linear layers before the first nonlinear operations in the first round. Let $S$ be the input state of the first $\chi$. First, we let the 288 bits in $S[x], 1 \leq x < 4$ be zero. Next, for the remaining 96 bits in $S[0]$, we set $S[0][0][z] = S[0][2][z]$ and $S[0][1][z] = 0$ for $0 \leq z < 32$. For the $\ell^{th}$-order HDL attack, we choose the input differences $\varDelta(i_0, i_1, \ldots, i_{\ell-1})$ that have $2\ell$ active bits of $S[0][0][i_j]$ and $S[0][2][i_j]$.

For 4-round XOODOO, we choose the input difference as $\varDelta(0, 20)$, the bias of $S[0][0][0]$ after 4 rounds would be $\frac{1}{2}$. For 5-round XOODOO, we choose the input difference as $\varDelta(0, 13, 14)$, and the bias of $S[1][1][28]$ is $2^{-8.96}$. We experimentally verified these two approximations, and found the 4-round distinguisher is truly deterministic and the 5-round 3rd-order HDL approximation has a bias of about $2^{-8.79}$ which is very close to our prediction. The big gap of biases between XOODYAK and XOODOO implies the XOODYAK gains some strength against HDL attacks by arranging $\chi$ after $\theta$ and $\rho_{west}$.

## 6.2 HDL Key-Recovery Attacks for Xoodyak

**Application 6: $2^{nd}$-order HDL key-recovery attack on 4-round** Xoodyak. In our $2^{nd}$-order key-recovery attack, we choose the two differences as $\Delta_0$ is active in $(u_0, u_{128})$, $\Delta_1$ is active in $(u_{72}, u_{200})$. We inject 8 conditions in the first round to cancel the differences. After 4-round Xoodyak initialization, our HATF technique shows that the bias of $S^{(4)}[0][0][14]$ is about 0.141 when all the conditions are satisfied. Experiments with $2^{26}$ randomly-selected samples show that the real bias is also 0.141 (up to 3 digits precision). The 8 conditions are listed as follows,

$$x_7 = k_7 \oplus k_{16} \oplus x_{16} \oplus x_{135} \oplus x_{144} \oplus x_{181}, x_{70} = k_{70} \oplus k_{93} \oplus x_{93} \oplus x_{107} \oplus x_{198} \oplus x_{221} \oplus 1$$
$$x_5 = k_5 \oplus k_{14} \oplus x_{14} \oplus x_{51} \oplus x_{133} \oplus x_{142} \oplus 1, x_{67} = k_{67} \oplus k_{90} \oplus k_{104} \oplus x_{90} \oplus x_{195} \oplus x_{218} \oplus 1$$
$$x_{18} = k_{18} \oplus k_{27} \oplus k_{32} \oplus x_{27} \oplus x_{146} \oplus x_{155} \oplus 1, x_{102} = k_{11} \oplus k_{102} \oplus k_{125} \oplus x_{125} \oplus x_{230} \oplus x_{253}$$
$$x_{37} = k_{37} \oplus k_{46} \oplus k_{83} \oplus x_{46} \oplus x_{165} \oplus x_{174}, x_{88} = k_{79} \oplus k_{88} \oplus x_{79} \oplus x_{207} \oplus x_{216} \oplus x_{253}$$

If not all the 8 conditions hold, the bias of $S^{(4)}[0][0][14]$ is at most 0.07. Thus, doing statistical tests can find the correct assignment of the 8 variables on the left side, and then 8 bits of key information. Firstly, we fixed all the nonce variables on the right side as 0, then we try all possible $2^8$ values of the nonce bits on the left. The values making $S^{(4)}[0][0][14]$ most biased is the values of the key expressions in each condition. According to the method distinguishing two normal distributions, the complexity is about $2^9$. Thus, the time/data complexity for recovering 8 bits of key information in the above conditions is about $2^{8+9} = 2^{17}$. We experimentally tested this attack, and among 100 experiments, we can recover the correct key 96 times. Due to the rotational-variance property of Xoodoo, recovering all 128-bit keys needs about $2^{21}$ computations. This is about 4 times faster than the DL attacks in [13].

**Application 7: Theoretical $3^{rd}$-order HDL key-recovery attack on 5-round** Xoodyak **under the single-key model.** In [13], a related-key DL attack on 5-round Ascon was given by Dunkelman and Weizman. The authors built this related-key DL approximation from a 4-round one (the second DL approximation in Application 1 of this section). Until now, the conditional cube attack is still the only attack that can reach 5 rounds under the single-key model [31]. In this section, we give a $3^{rd}$-order HDL attack on 5-round Xoodyak under the single-key model.

We choose the $3^{rd}$-order difference as $(\Delta_0, \Delta_1, \Delta_2)$ where $\Delta_0$ is active in $(u_0, u_{128})$, $\Delta_1$ is active in $(u_9, u_{137})$, and $\Delta_2$ is active in $(u_{36}, u_{164})$. We inject 12 conditions into the first round, then after 5 full rounds, the bias of $S^{(5)}[0][0][29]$ is predicted as $2^{-30.72}$. The 12 conditions are all linear and provided in the full version. We assume that if not all 12 conditions are true, the bias of $S^{(5)}[0][29]$ is close to 0. Thus, a statistical test with approximately $2^{64}$ samples is enough to distinguish them. Then we can use a similar strategy as the 4-round attack to recover 12 bits of key information. The complexity is about $12 \times 2^{64} \approx 2^{68}$. We can repeat this process for 5 other positions by rotation to recover 60 more bits of key information, the remaining keys can be searched by force. The whole time/data complex of recovering all key bits is about $2^{70.2}$.

## 7 HD Cryptanalysis Based on DSF Degree Estimation

According to Sect. 3, $f \circ \mathcal{M}$ plays an important role in (higher-order) differential cryptanalysis, thus we call it *differential supporting function* and give it a formal definition.

**Definition 3. (Differential Supporting Function (**DSF**)).** *Given a Boolean function* $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ *and an* $\ell^{th}$*-order difference* $\boldsymbol{\Delta} = (\Delta_0, \Delta_1, \ldots, \Delta_{\ell-1}) \in (\mathbb{F}_2^n)^\ell$*, the composite Boolean function*

$$\mathrm{DSF}^{\ell}_{f,X,\boldsymbol{\Delta}}(\boldsymbol{x}) = f \circ \mathcal{M}(\boldsymbol{x}) = f(X \oplus \boldsymbol{x}\boldsymbol{\Delta}), \boldsymbol{x} = (x_0, x_1, \ldots, x_{\ell-1})$$

*is called the* $\ell^{th}$*-order differential supporting function* (DSF) *of* $f$ *with respect to* $(X, \boldsymbol{\Delta})$*. When the order* $\ell$ *is clear from context, we will ignore it in the notation, i.e.,* $\mathrm{DSF}_{f,X,\boldsymbol{\Delta}}(\boldsymbol{x})$*.*

The DSF provides a convenient way to find good *affine space* for the input that reduces the algebraic degree of the target cipher. In this paper, we take the Ascon permutation as an example to show the usage of the DSF. Until now, all attacks on the Ascon permutation with complexity less than or equal $2^{64}$ can only reach 7 rounds. The only integral distinguishers given by Todo [26] require more than $2^{130}$ calls to attack 8 and more rounds, already higher than Ascon's claimed security level ($2^{128}$ calls). By analyzing the degree evolution of the DSF, we present a new HD distinguisher for 8 rounds requiring only $2^{46}$ complexity.

**Basic Idea.** Note that in the Definition 3, $\boldsymbol{x}$ are variables while $X$ and $\boldsymbol{\Delta}$ are parameters (here the term "parameters" means $X$ and $\boldsymbol{\Delta}$ should be fixed as a constant). Hence, different $X$ and $\boldsymbol{\Delta}$ will lead to different DSF. So it is possible to find some proper $(X, \boldsymbol{\Delta})$ that reduce the algebraic degree of the DSF. More specifically, $\deg(\mathrm{DSF}_{f,X,\boldsymbol{\Delta}})$ may be reduced to some values smaller than the order $\ell$. In this case, we derive an integral property for $\mathrm{DSF}_{f,X,\boldsymbol{\Delta}}$. Applying the inverse of $\mathcal{M}$, we immediately derive an $\ell^{th}$-order difference yielding the following property, *i.e.*,

$$\mathcal{D}_{\boldsymbol{\Delta}} f(X) = \bigoplus_{x \in \mathbb{F}_2^\ell} \mathrm{DSF}_{f,X,\boldsymbol{\Delta}}(x) = 0.$$

To estimate the degree upper bound of a DSF, we cut a Boolean function into two phases as follows,

$$\mathrm{DSF}_{f,X,\boldsymbol{\Delta}}(\boldsymbol{x}) = f(X \oplus \boldsymbol{x}\boldsymbol{\Delta}) = f^1 \circ f^0(X \oplus \boldsymbol{x}\boldsymbol{\Delta}).$$

We let $f^0$ be simple so that we can compute out its exact ANFs as well as the exact degrees of the output of $f^0(X \oplus \boldsymbol{x}\boldsymbol{\Delta})$. Next, we update the obtained degrees by $f^1$ to obtain the degree upper bounds of the whole $\mathrm{DSF}_{f,X,\boldsymbol{\Delta}}$.

In terms of the $r$-round Ascon permutation, we choose its first $r_0 = 2.5$ rounds as $f^0$ for it achieves a balance between efficiency and precision[8]. The

[8] A larger $r_0$ will make the estimation of $\deg(\mathrm{DSF}_{f,X,\boldsymbol{\Delta}})$ more precise but more time-consuming to compute the ANFs, while a smaller $r_0$ may undermine the precision.

remaining $(r-2.5)$-round permutation is seen as $f^1$, the degree update of $f^1$ can be done by methods such as the division properties [26,27] or any other suitable methods. In this paper, we use the method of the degree matrix to update the algebraic degree of $f^1$:

**Definition 4 (Degree Matrix of $S^{(r)}$).** *The algebraic degrees or their upper bounds of the bits in the state $S^{(r)}$ are called a degree matrix of $S^{(r)}$, denoted by*

$$\mathrm{DM}(S^{(r)}) = \Big(\deg(S^{(r)}[i][j]), 0 \le i < 5, 0 \le j < 64\Big).$$

Given the degree matrix of $S^{(r)}$, we can quickly calculate the degree matrix of $S^{(r+1)}$ considering the ANFs of the $p_S$ and $p_L$ (see our full version). Our experiments show that the degree matrix method is not worse than the division property to calculate the upper bound on the algebraic degree of $\mathrm{DSF}_{f,X,\boldsymbol{\Delta}}$ for the case of Ascon permutation[9]. The only challenge now is to find a desirable combination of $(X, \boldsymbol{\Delta})$.

**Heuristic Method of Choosing** $(X, \boldsymbol{\Delta})$. To find a proper $(X, \boldsymbol{\Delta})$, a naive idea is to exhaust all possible values of $(X, \boldsymbol{\Delta})$, but the search space is clearly too large. For Ascon, we use the same notations as we do in Sect. 5. Considering the first operation of the Ascon permutation without $p_C$ (we can safely ignore the first $p_C$ operation since we target the permutation) is $p_S$ which consists of 64 parallel small Sboxes. If we consider independent $\ell'^{th}$-order differences for each Sbox $\mathcal{S}$, in total we are considering an $(\ell = 64\ell')^{th}$-order differences for the whole permutation. Our experiments show $\ell' = 1$ will achieve the best performance. This is not surprising, since $\ell' = 1$ means that we put one variable in each Sbox to linearize all Sboxes, similar ideas were already mentioned in some previous works such as [7]. With $\ell' = 1$, our $64^{th}$-order input difference is then denoted by $\boldsymbol{\Delta} = (\Delta_0, \Delta_1, \ldots, \Delta_{63})$. Thus, we write $p_S(X \oplus \boldsymbol{x}\boldsymbol{\Delta})$ as follows:

$$p_S(X \oplus \boldsymbol{x}\boldsymbol{\Delta}) = \mathcal{S}(X_0 \oplus x_0\Delta'_0)||\mathcal{S}(X_1 \oplus x_1\Delta'_1)|| \cdots ||\mathcal{S}(X_{63} \oplus x_{63}\Delta'_{63}),$$

where $X = X_0||X_1||\cdots||X_{63}$ and $\Delta_i = 0||\cdots||\Delta'_i||\cdots||0$ for $0 \le i < 64$.

To further reduce the search space, we restrict the 64 $X_i$'s and 64 $\Delta'_i$'s to be equal respectively, *i.e.*, $(X_i, \Delta'_i) = (\bar{X}, \bar{\Delta})$ for $0 \le i < 64$. Therefore, we only need to consider $2^5$ possibilities for $\bar{X}$ and 31 possibilities for $\bar{\Delta}$ (excluding the trivial case $\bar{\Delta} = 0$). The total search space is reduced to $32 \times 31 = 992$ different cases.

For each $(\bar{X}, \bar{\Delta}) \in \mathbb{F}_2^5 \times \mathbb{F}_2^5 \backslash \{0\}$, we calculate the ANFs of $f^0(X \oplus \boldsymbol{x}\boldsymbol{\Delta})$, then derive the degree matrix of its output. After that we update the degree matrix according to [14] to calculate the degree matrix of $S^{(r)}$ (for $r \ge 4$) which is the degree upper bound of the corresponding DSF. If the degree of a certain DSF

[9] Note that the degree matrix method only happens to be as good as the division property in this specific case. We choose the degree matrix method simply because it can be more easily integrated into our algorithm. In general case, the division property has overwhelming advantages in accuracy and versatility.

**Table 4.** Upper bounds on the algebraic degree of the DSF of the Ascon permutation with $(X, \boldsymbol{\Delta})$ in Eq. 9. We experimentally verified all algebraic degrees up to 7 rounds.

| Round $r$ | Upper bounds on the algebraic degree | | | | |
|---|---|---|---|---|---|
| | $S^{(r)}[0]$ | $S^{(r)}[1]$ | $S^{(r)}[2]$ | $S^{(r)}[3]$ | $S^{(r)}[4]$ |
| 4 | 3 | 3 | 2 | 2 | 3 |
| 5 | 6 | 5 | 5 | 6 | 6 |
| 6 | 11 | 11 | 12 | 12 | 11 |
| 7 | 23 | 24 | 23 | 23 | 22 |
| 8 | 47 | 47 | 45 | 46 | 47 |

is smaller than 64, we find useful $64^{th}$ HD distinguishers for $r$-round Ascon permutation.

We found dozens of useful HD distinguishers with orders lower than 64 for up to 8 rounds. Among them, there are 8 optimal combinations of $(\bar{X}, \bar{\Delta})$ that make the algebraic degree of the third word of $S^{(8)}$ be only 45. They are

$$(\bar{X}, \bar{\Delta}) \in \left\{ \begin{array}{l} (\texttt{0x6}, \texttt{0x13}), (\texttt{0xa}, \texttt{0x13}), (\texttt{0xc}, \texttt{0x17}), (\texttt{0xf}, \texttt{0x18}), \\ (\texttt{0x15}, \texttt{0x13}), (\texttt{0x17}, \texttt{0x18}), (\texttt{0x19}, \texttt{0x13}), (\texttt{0x1b}, \texttt{0x17}) \end{array} \right\}. \tag{9}$$

In Table 4, we list all the upper bounds on degrees of the DSF up to 8-round Ascon permutation with respect to $(X, \boldsymbol{\Delta})$ in Eq. 9. As is seen, for 7 rounds, the degree upper bound of $S^{(7)}$ is only 24, so $2^{25}$ chosen texts are enough to enforce the zero output difference. For the 8-round output, the algebraic degree is at most 47. Therefore if we choose $2^{48}$ plaintexts in any 48-dimensional affine space defined by values in Eq. 9, the summation of all ciphertexts will be zero. Given a random permutation, the probability that the summation of such $2^{48}$ ciphertexts will be zero is $2^{-320}$. Thus, $2^{48}$ chosen plaintexts are enough to distinguish the 8-round Ascon permutation from a random permutation.

## 8 Conclusion and Discussion

In this paper, we revisited the HD/HDL cryptanalysis from an algebraic perspective. HATF and DSF are two tools for probabilistic and deterministic HDL/HD cryptanalysis, respectively. Improved results for Ascon and Xoodyak, as well as Xoodoo are obtained from the two tools. We believe that the HDL cryptanalysis has more potential than expected, and deserves more attention.

In terms of HATF, it is the first theoretical tool for nondeterministic HDL cryptanalysis. It can predict the biases of an $\ell^{th}$-order HDL approximations with a time complexity of $\mathcal{O}(2^{\ell+d2\ell})$ for ciphers with $d$-degree round functions. For ciphers with quadratic round functions, the time complexity can be reduced to $\mathcal{O}(2^{3.8\ell})$. Thus, HATF is very useful for HDL cryptanalysis of permutation-based ciphers such as Ascon and Xoodyak. The precision of HATF is supported by experiments (see [14]). When HATF predicts a biased bit, it is of a great probability that it is biased as far as our experiments show. Finally, we make it clear again that HATF does not guarantee any lower or upper bounds on the bias of a HDL approximation. Whenever possible, the theoretical results should be verified with experiments.

For DSF, it provides an intuitive method for detecting HD distinguishers for permutations. With proper choices of $(X, \boldsymbol{\Delta})$, the algebraic degree of a DSF might drop drastically. Therefore, we have a greater opportunity to find better HD distinguishers rather than to analyze the original Boolean function.

We have shown that a proper partitioning of the input space can improve the precision of HATF, how to find better or even optimal partitioning methods? Can we use the HDL cryptanalysis to propose best key-recovery attacks on some ciphers in terms of rounds? For DSF, our method to choose $(X, \boldsymbol{\Delta})$ is intuitive and actually considers only a small percentage of candidates, can we find better $(X, \boldsymbol{\Delta})$ leading to better HD distinguishers for Ascon permutation? These are interesting questions worth exploring that we leave as future work.

**Acknowledgments.** We are grateful to the anonymous referees for their comments that improved the quality of this article. Kai Hu thanks Yang Wang for the fruitful discussion. The authors are supported by the France-Singapore NRF-ANR research grant NRF2020-NRF-ANR072 and the Singapore NRF Investigatorship research grant NRF-NRFI08-2022-0013.

## References

1. Bar-On, A., Dunkelman, O., Keller, N., Weizman, A.: DLCT: a new tool for differential-linear cryptanalysis. In: EUROCRYPT (2019)
2. Biham, E., Dunkelman, O., Keller, N.: A new attack on 6-round IDEA. In: FSE (2007)
3. Biham, E., Dunkelman, O., Keller, N.: Enhancing differential-linear cryptanalysis. In: ASIACRYPT (2002)
4. E. Biham, O. Dunkelman, Keller, N.: New combined attacks on block ciphers. In: FSE (2005)
5. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: CRYPTO (1990)
6. Blondeau, C., Leander, G., Nyberg, K.: Differential-linear cryptanalysis revisited. J. Cryptol. **30**(3), 859–888 (2017)
7. Bonnetain, X., Leurent, G., Naya-Plasencia, M., Schrottenloher, A.: Quantum linearization attacks. In: ASIACRYPT (2021)
8. Daemen, J., Hoffert, S., Assche, G., Keer, R.: The design of Xoodoo and Xoofff. IACR ToSC (4) (2018)
9. Daemen, J., Hoffert, S., Peeters, M., Assche, G., Keer, R.: Xoodyak, a lightweight cryptographic scheme. In: IACR ToSC, 2020(S1) (2020)
10. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: EUROCRYPT (2009)
11. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Cryptanalysis of Ascon. In: CT-RSA (2015)
12. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2: lightweight authenticated encryption and hashing. J. Cryptol. **34**(3), 33 (2021)

13. Dunkelman, O., Weizman, A.: Differential-linear cryptanalysis on Xoodyak. In: NIST Lightweight Cryptography Workshop (2022)
14. Hu, K., Peyrin, T., Tan, Q., Yap, T.: Revisiting Higher-Order Differential-Linear Attacks from an Algebraic Perspective. Cryptology ePrint Archive, 2022/1335
15. Florence Jessie, M., Neil James Alexander, S.: The Theory of Error-Correcting Codes, vol. 16. Elsevier (1977)
16. Knudsen, L.: Truncated and higher order differentials. In: FSE (1994)
17. Lai, X., Massey, J.: A proposal for a new block encryption standard. In: EUROCRYPT (1990)
18. Langford, S., Hellman, M.: Differential-Linear cryptanalysis. In: CRYPTO (1994)
19. Li, Z., Dong, X., Wang, X.: Conditional cube attack on round-reduced ASCON. IACR ToSC, 2017(1) (2017)
20. Liu, M., Lu, X., Lin, D.: Differential-linear cryptanalysis from an algebraic perspective. In: CRYPTO (2021)
21. Liu, Y., Sun, S., Li, C.: Rotational cryptanalysis from a differential-linear perspective - practical distinguishers for round-reduced FRIET, Xoodoo, and Alzette. In: EUROCRYPT (2021)
22. Matsui, M.: Linear cryptanalysis method for DES cipher. In: EUROCRYPT (1993)
23. Rohit, R., Hu, K., Sarkar, S., Sun, S.: Misuse-free key-recovery and distinguishing attacks on 7-Round Ascon. IACR ToSC, 2021(1) (2021)
24. Shi, D., Sun, S., Sasaki, Y., Li, C., Hu, L.: Correlation of quadratic Boolean functions: cryptanalysis of all versions of full MORUS. In: CRYPTO (2019)
25. Tezcan, C.: Analysis of Ascon, DryGASCON, and Shamash Permutations. IACR Cryptol. ePrint Arch., 2020/1458
26. Todo, Y.: Structural evaluation by generalized integral property. In: EUROCRYPT (2015)
27. Todo, Y., Morii, M.: Bit-based division property and application to Simon family. In: FSE (2016)
28. Vaudenay, S.: Provable security for block ciphers by decorrelation. In: STACS (1998)
29. Wagner, D.: The Boomerang Attack. In: FSE (1999)
30. Xuejia, L.: Higher order derivatives and differential cryptanalysis. In: Communications and Cryptography, pp. 227–233 (1994)
31. Zhou, H., Li, Z., Dong, X., Jia, K., Meier, W.: Practical key-recovery attacks on round-reduced Ketje Jr, Xoodoo-AE and Xoodyak. Comput. J. **63**(8), 1231–1246 (2020)