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
kai.hu@ntu.edu.sg, thomas.peyrin@ntu.edu.sg, quanquan001@e.ntu.edu.sg, trevor.yap@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, 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 ℓ^{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}. 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 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 improved the distinguishing attacks on 8-round Ascon permutation, with a complexity reduced from 2^{130} to 2^{46}. Also, we pro-

vide a new zero-sum distinguisher for the full 12-round Ascon permutation with 2^{55} time/data complexity, improving over the previous best one that required 2^{130} calls. 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 [BS90,Mat93]. 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 [Vau98] is vulnerable to boomerang and Differential-Linear (DL) cryptanalysis [Wag99,BDK02] 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 [LH94] and it remains the best-known attack on many ciphers, *e.g.*, AES competition finalist Serpent [BAK98,LLL21]. For a difference-mask pair (Δ_I, λ_O), the bias q' of a DL approximation can be derived from the following equation

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

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 [LH94,BDK02], 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 indepedence assumptions.

As pointed out in [BDK02], 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.* [BLN17], 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.* [BDKW19] 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.* [LLL21] at CRYPTO 2021. If we define a Boolean function according

to λ_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 Δ_I, *i.e.*, $x\Delta_I = (\Delta_I[0] \cdot x, \ldots, \Delta_I[n-1] \cdot x)$. 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 intuitive observation linking Equation 1 and 2,

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

where $D_x f_{\Delta_I}$ is the partial derivative of f_{Δ_I} with respect to x. That is to say, considering Equations 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 the 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 [BDK05]. 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 [Lai94] and later studied by Knudsen [Knu94]. It is a natural generalization of the differential attack that takes advantage of having access to more plaintexts. Given an ℓ^{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 ℓ^{th} derivative of a (partial) cipher E with respect to $\boldsymbol{\Delta}_I$ studies the probability

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

[1] In [LLL21], there is another terminology DATF when ATF is used to construct transitional expressions for f_Δ. In this paper, we directly use ATF for all kinds of Boolean functions no matter whether we target f or f_Δ.

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

As the name higher-order differential-linear suggests, HDL cryptanalysis [BDK05] studies the bias concerning an ℓ^{th}-order input difference $\boldsymbol{\Delta}_I$ and an output mask λ_O. The bias ε 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.* [BDK05] gave an analysis based on viewing E as two sub-ciphers $E = E_1 \circ E_0$. Suppose that we know an ℓ^{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 ε 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 ℓ to attack an interesting number of rounds, but if $|q| \neq \frac{1}{2}$, ε 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 [LM90] allow weak-key linear approximations with bias $\frac{1}{2}$, which makes them vulnerable to HDL attacks [BDK05,BDK07].

1.3 Motivation and Contributions

Considering that DL attacks are efficient for many important primitives, such as Ascon [DEMS21] (recent winner of the NIST lightweight competition) and Xoodyak [DHP+20], 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, the algebraic perspective on DL attacks [LLL21] opened up a new road to study the differential/DL attacks and achieved better precision for some important ciphers such as Ascon [DEMS21]. 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 the 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, and the other one 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.

Primitive	Round	Order	Bias		Method	Reference
			Experiment	Theory		
Ascon Init.	4	1^{st}	2^{-2}	2^{-20} 2^{-5} $2^{-2.365}$ $\mathbf{2^{-2.09}}$	Classical DLCT ATF **HATF**	[DEMS15] [BDKW19] [LLL21] **Section 5.1**
		2^{nd}	2^{-1}	$\mathbf{2^{-1}}$	**HATF**	**Section 5.1**
	5	1^{st}	2^{-9}	– $\mathbf{2^{-10}}$	Experimental **HATF**	[DEMS15] **Section 5.1**
		2^{nd}	$2^{-6.60}$	$\mathbf{2^{-7.05}}$	**HATF**	**Section 5.1**
		8^{th}	$2^{-3.35}$	$\mathbf{2^{-4.73}}$	**HATF**	**Section 5.1**
	6	3^{rd}	2^{-22}†	$\mathbf{2^{-25.97}}$†	**HATF**	**Section 5.1**
Xoodyak Init.	4	1^{st}	$2^{-9.7}$	– $\mathbf{2^{-9.67}}$	Experimental **HATF**	[DW22] **Section 6.1**
			$-2^{-5.36}$‡	– $\mathbf{-2^{-6.0}}$	Experimental **HATF**	[DW22] **Section 6.1**
		2^{nd}	$2^{-5.72}$	$\mathbf{2^{-5.72}}$	**HATF**	**Section 6.1**
		4^{th}	2^{-1}	$\mathbf{2^{-1}}$	**HATF**	**Section 6.1**
	5	2^{nd}	–	$\mathbf{2^{-45}}$	**HATF**	**Section 6.1**
Xoodoo	4	- 4^{th}	2^{-1} 2^{-1}	2^{-1} $\mathbf{2^{-1}}$	Rot. DL **HATF**	[LSL21] **Section 6.1**
	5	3^{rd}	$2^{-8.79}$	$\mathbf{2^{-8.96}}$	**HATF**	**Section 6.1**

† This bias holds when 24 conditions are satisfied.
‡ In [DW22], 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 ℓ dimensional affine space to an ℓ 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. 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 ℓ 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 $2^{-2.09}$, which is better than previous tools such as the DLCT [BDKW19] (2^{-5}) or the ATF [LLL21] ($2^{-2.36}$). 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 [DW22]. 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 [LLL21] and 4 times faster than the conditional cube attacks [LDW17]. For 4-round Xoodyak, the HDL attack is 4 times more efficient than the DL attack [DW22]. A summary of these key-recovery attacks is given in Table 2.

Finally, we make clear that HATF cannot give any lower or upper bounds for HDL approximations in theory. However, it is precise to predict biased bits, which has been well supported by our experiments. In cases where the reported bias is high, it was always true that the experimental bias was observed also as high (we have not seen any counterexample for this). We provide data and discuss the precision of HATF in Section J of Supplementary Material 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 [DEMS21]. A 46th-order HD will lead to a zero output difference (in 64 bits) for 8 rounds, *i.e.*, 2^{46} plaintexts are enough

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	[DEMS15]
		2	**2**	**2nd HDL**	**Section 5.1**
	5	18	18	DL	[DEMS15]
		12	**12**	**8th HDL**	**Section 5.1**
Key-Recovery	5	36	36	DL	[DEMS15]
		26	26	Cond. DL	[LLL21]
		24	24	Cond. Cube	[LDW17]
		22	**22**	**Cond. HDL**	**Section 5.2**
Best	7	77	103	Cond. Cube	[LDW17]
Xoodyak Initialization					
Key-Recovery	4	23	23	DL	[DW22]
		21	**21**	**Cond. HDL**	**Section 6.2**
	5	22	22	DL†	[DW22]
		70	**70**	**Cond. HDL**	**Section 6.2**
Best	6	43.8	43.8	Cond. Cube	[ZLD+20]

† 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.

to distinguish an 8-round Ascon permutation from a random permutation (the previous best known distinguisher requires 2^{130} computations [Tod15]). 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.

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 anonymous git repository `https://github.com/hukaisdu/HDL.git`.

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

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	[Tod15]
		46	**46**	**HD**	**Section 7**
Best	11	315	315	Integral	[Tod15]
Inside out	12	130	130	Zero-Sum	[Tod15]
		55	**55**	**Zero-Sum**	**Section I**

and Section I of Supplementary Material, we give the theory and results of DSF. In Section 8 and Section J, we have some discussions on the precision of HATF and conclude 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 ℓ 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 $\Delta \in \mathbb{F}_2^n$, $x\Delta = (\Delta[0]x, \Delta[1]x, \ldots, \Delta[n-1]x)$[2]. 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

[2] Example 1 in Section 3 is helpful for a better understanding to this notation of $x\Delta$.

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 ε 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 some papers such as [LLL21], the bias is taken while in other papers such as [AFK+08] the correlation is used. In this paper, we will only use the bias ε to measure the unbalancedness.

2.3 The Algebraic Perspective on DL

In [LLL21], Liu *et al.* introduced a new algebraic method for the differential and DL cryptanalysis as we have already mentioned in Section 1. Recalling Equation 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 Section 1, Liu *et al.* proposed Equation 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 the algebraic perspective on the ℓ^{th}-order HD.

Basic idea of Algebraic Transitional Forms. Equation 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_Δ. ATF of a Boolean function f, denoted by $\mathcal{A}(f)$, is a composite representation of f constructed iteratively. From $\mathcal{A}(f_\Delta)$, we obtain a simple 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 [LLL21], 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 ([LLL21]). *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*

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

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 ε_i, so it happens

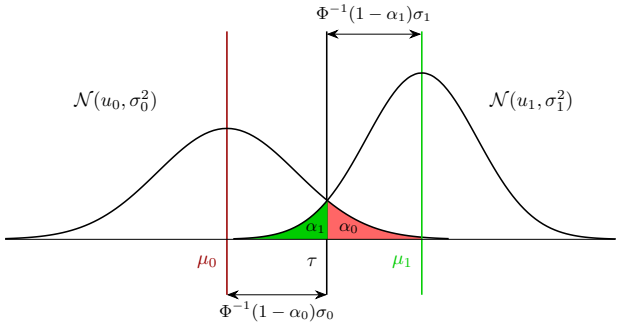


Fig. 1: The relationship among u_0, u_1, τ, α_0 and α_1

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 Equation 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. If there is one so-called isolate term in $D_x\mathcal{A}(f_\Delta)$, this term would be expanded with deeper variables and its bias would be calculated according to the expanded expression. 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.

2.4 Distinguishing Two Normal Distributions

Suppose that we have known a statistics T obeys either $\mathcal{N}(\mu_0, \sigma_0^2)$ or $\mathcal{N}(\mu_1, \sigma_1^2)$ and *w.l.o.g.* $u_0 < u_1$, we want to judge which one T really follows. The method is to find a threshold $\mu_0 < \tau < u_1$ such that when $T < \tau$ we judge $T \sim \mathcal{N}(\mu_0, \sigma_0^2)$, otherwise $T \sim \mathcal{N}(\mu_1, \sigma_1^2)$, enduring the risks of two types of errors:

1. α_0: the probability that $T \sim \mathcal{N}(\mu_0, \sigma_0^2)$ but we judge it as $T \sim \mathcal{N}(\mu_1, \sigma_1^2)$;
2. α_1: the probability that $T \sim \mathcal{N}(\mu_1, \sigma_1^2)$ but we judge it as $T \sim \mathcal{N}(\mu_0, \sigma_0^2)$;

The relationship between $\mu_0, \mu_1, \tau, \alpha_0$ and α_1 is illustrated in Figure 1. Let Φ be the cumulative distribution functions of the standard normal distribution. According to Figure 1 we have

$$\begin{cases} \tau = \mu_0 + \Phi^{-1}(1-\alpha_0)\sigma_0 = \mu_1 - \Phi^{-1}(1-\alpha_1)\sigma_1 \\ \mu_1 - \mu_0 = \Phi^{-1}(1-\alpha_0)\sigma_0 + \Phi^{-1}(1-\alpha_1)\sigma_1 \end{cases} \tag{5}$$

In practice, the normal distribution is usually derived from the binary distribution, so $\mu_0, \mu_1, \sigma_0, \sigma_1$ are related to the number of tests. According to Equation 5, it is easy to calculate the proper number of tests satisfying the desiring α_0, α_1.

3 HD/HDL Cryptanalysis from an Algebraic Perspective

In this section, we give the theory about the ℓ^{th} derivative of a Boolean function f from an algebraic perspective. This is a general case of the algebraic perspective on DL [LLL21]. It is well known that the cube/integral attacks are special cases of HD attacks with all ℓ 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 [DS09]. The theory in this section answers such a question: given any ℓ 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 ℓ^{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 ℓ^{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 ℓ-dimensional affine space, so we can link $\mathbb{A}^\ell$ to any another ℓ-dimensional affine space $(\mathbb{A}^\ell)'$ by a bijective mapping $\mathcal{M}^\ell$ that sends $(\mathbb{A}^\ell)'$ to $\mathbb{A}^\ell$. Not surprisingly, we tend to choose the simplest ℓ-dimensional affine space, *i.e.*, the ℓ-dimensional linear space $\mathbb{F}_2^\ell$. One choice of $\mathcal{M}^\ell$ can be

$$\begin{aligned} \mathcal{M}^\ell &: \mathbb{F}_2^\ell \to \mathbb{A}^\ell \\ (x_0, x_1, \ldots, x_{\ell-1}) &\mapsto X \oplus x_0\Delta_0 \oplus x_1\Delta_1 \oplus \cdots \oplus x_{\ell-1}\Delta_{\ell-1} \triangleq X \oplus \boldsymbol{x}\boldsymbol{\Delta} \end{aligned} \tag{6}$$

We define a new function $f_{\boldsymbol{\Delta}}$ from f with the transformed input set as[3]:

$$\begin{aligned} f_{\boldsymbol{\Delta}} &: \mathbb{F}_2^n \to \mathbb{F}_2 \\ X &\mapsto f(X \oplus \boldsymbol{x}\boldsymbol{\Delta}) \end{aligned}$$

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 have the following proposition,

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

Proof. With $\mathcal{M}^\ell$ as given in Equation 6, 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 cube attacks,

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

□

[3] 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.

Example 1. Let $f : \mathbb{F}_2^3 \to \mathbb{F}_2$ be $f(a_0, a_1, a_2) = a_0a_1a_2 \oplus a_0a_1 \oplus a_0a_2 \oplus a_1a_2$, $\boldsymbol{\Delta} = (\Delta_0, \Delta_1)$ where $\Delta_0 = (1, 0, 1)$ and $\Delta_1 = (1, 1, 1)$, we consider the 2^{nd} derivative of f at a point $X = (X_0, X_1, X_2) \in (\mathbb{F}_2)^3$. According to Equation 6, $\mathcal{M}(x_0, x_1) = X \oplus x_0\Delta_0 \oplus x_1\Delta_1 = (X_0 \oplus x_0 \oplus x_1, X_1 \oplus x_1, X_2 \oplus x_0 \oplus x_1)$. The composition of f and $\mathcal{M}$ is then

$$\begin{aligned} f \circ \mathcal{M}(x_0, x_1) &= f(X_0 \oplus x_0 \oplus x_1, X_1 \oplus x_1, X_2 \oplus x_0 \oplus x_1) \\ &= x_0x_1(X_0 \oplus X_2 \oplus 1) \oplus x_0X_0X_1 \oplus x_0X_0 \oplus x_0X_1X_2 \oplus x_0X_1 \\ &\oplus x_0X_2 \oplus x_0 \oplus x_1X_0X_1 \oplus x_1X_0X_2 \oplus x_1X_0 \oplus x_1X_1X_2 \\ &\oplus x_1X_1 \oplus x_1X_2 \oplus X_0X_1X_2 \oplus X_0X_1 \oplus X_0X_2 \oplus X_1X_2 \end{aligned}$$

We can see that $\mathcal{D}_{\boldsymbol{\Delta}} f(X) = \mathsf{Coe}\left(f \circ M(x_0, x_1), x_0x_1\right) = X_0 \oplus X_2 \oplus 1$.

4 Estimating HDL Approximation Biases Using HATF

On the basis of Section 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 [LLL21]: we construct a composite representation of the HDL approximation, then estimate the bias according to the composite representation. In Section 4.1, we construct the HATF for a cipher E, which is a composite representation of the ℓ^{th} derivative of E. In Section 4.2, we estimate the bias of the ℓ^{th}-order HDL based on HATF under some reasonable assumptions. In Section 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.

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

$$E = E_{R-1} \circ E_{R-2} \circ \cdots \circ E_0, \quad E_r : \mathbb{F}_2^n \to \mathbb{F}_2^n \tag{7}$$

then, according to Proposition 1, to calculate the ℓ^{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, then

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

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 ℓ 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:

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

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 $\mathcal{E}_r^\ell, 0 \le r < R$, we derive from E a composite function $\mathcal{E}^\ell$:

$$\mathcal{E}^\ell = \mathcal{E}_{R-1}^\ell \circ \mathcal{E}_{R-2}^\ell \circ \cdots \circ \mathcal{E}_0^\ell, \quad \mathcal{E}_i^\ell : (\mathbb{F}_2^n)^{2^\ell} \to (\mathbb{F}_2^n)^{2^\ell} \tag{8}$$

Definition 1 (ℓ^{th} Higher-order Algebraic Transitional Form (ℓ^{th} HATF)). *The composite function in Equation 8 above is called the ℓ^{th} Higher-order Algebraic Transitional Form (ℓ^{th} HATF) of E. If the order information is clear from the context, we will omit the superscript ℓ for convenience.*

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

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

Input: 1. the ANFs of components of $E = E_{R-1} \circ \cdots E_0$,
2. the order ℓ,
3. the block size n,
4. an ℓ^{th}-order difference $(\Delta_0, \ldots, \Delta_{\ell-1})$,
5. an input value X

Output: the ℓ^{th}-order HATF $\mathcal{E}^\ell = \mathcal{E}^\ell_{r-1} \circ \cdots \circ \mathcal{E}^\ell_0$

1: Let $\alpha_0^{(0)} = X$, $\alpha_{e_i}^{(0)} = \Delta_i$, $\Delta_u^{(0)} = 0$ for all $wt(u) \geq 2$
2: **for** $1 \leq r < R$ **do**
3: **for** $0 \leq i < n$ **do**
4: Calculate $f = E_r[i]\left(\bigoplus_{u \in \mathbb{F}_2^n} \alpha_u \boldsymbol{x}^u\right)$
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 $\mathcal{E}^\ell$

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^ℓ 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 Equation 8 be the HATF of E in Equation 7. For each $0 \leq r < R$, the algebraic degree of $\mathcal{E}^\ell_r$ is equal to the algebraic degree of E_r.*

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

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

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* $u = 0$, $\alpha_0^{(r+1)}$ *is just the output of* $E_r(\alpha_0^{(r)})$, *so* $\deg(\mathcal{E}^\ell_r) = \deg(E_r)$. □

4.2 Estimation of the HDL Bias based on HATF

Suppose we have obtained the ℓ^{th} HATF of a cipher $E = E_{R-1} \circ \cdots \circ E_0$ according to Algorithm 1. The biases of the ℓ^{th}-order HD/HDL approximations of all the

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

Input: 1. the HATF $\mathcal{E}^\ell = \mathcal{E}_{r-1}^\ell \circ \cdots \circ \mathcal{E}_0^\ell$,
2. the bias of $\alpha_u^{(0)}[i]$ for all $0 \le i < n$ and $u \in \mathbb{F}_2^n$

Output: the ℓ^{th}-order HATF $\mathcal{E}^\ell = \mathcal{E}_{r-1}^\ell \circ \cdots \circ \mathcal{E}_0^\ell$

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)
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 $0 \le i < n$

output bits of E are biases of $\alpha_{\mathbf{1}}^{(R)}$ (where $\mathbf{1}$ is the ℓ-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 and differences chosen by adversaries (under a chosen-plaintext attack).

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.

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 [SSS+19]). *A quadratic Boolean function is disjoint if all its quadratic monomials do not share any common variables.*

In [SSS+19], Shi *et al.* introduced a method of converting any quadratic Boolean function to a disjoint form with polynomial time complexity. The detailed algorithm can be found in [SSS+19] and will be omitted here. We only give a small example to show its core idea.

Example 2. 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 $g = x_0x'_1 + (x'_1 \oplus x_2)x_2 = x_0x'_1 \oplus x'_1x_2 \oplus x_2$.
2. $g = x'_1(x_0 \oplus x_2) + x_2$, we then let $x'_0 = x_0 \oplus x_2$ and obtain $g = x'_1x'_0 \oplus x_2$, 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 $g = f(M^t[x_0, x_1, x_2]^t)$ where $M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$. Equivalently, $f = g \circ (M^t)^{-1}$.

We write this method as a Lemma for convenient citation.

Lemma 2 ([SSS+19]). *A quadratic Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ can be converted into a disjoint Boolean function $g : \mathbb{F}_2^n \to \mathbb{F}_2^n$ 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 $g : \mathbb{F}_2^n \to \mathbb{F}_2$ is 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 ℓ^{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 $\alpha_u^{(r)}, 0 \le r < R, u \in \mathbb{F}_2^n$ 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 only concerned about line 6 only

1. When $\deg(\alpha_u^{(r+1)}[i]) \le 1$, we do not introduce 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] 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.

According to our experiments, HATF provides good precision in predicting biased HDL approximations for some ciphers. Taking HDL cryptanalysis of the Ascon initialization as an example, we give the curves of theoretical and experimental results of the HDL approximations for 4- and 5-round Ascon in Section J of Supplementary Material. It can be seen that HATF is truly useful in predicting highly biased bits.

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 can partition the whole input space into several subspaces. For each of these small spaces, we apply our HATF technique to evaluate the biases. The partition of space can significantly simplify the ANFs of $\mathcal{E}^\ell$, so for each subspace, the precision can be improved. The methods of partitioning the space are chosen in different ways for different ciphers, which will be described in our applications. Here we only give a general idea of the usage of this technique.

Given an R-round cipher E, suppose we have partitioned the whole input space S into κ subspaces as $S = (S_0, S_1, \ldots, S_{\kappa-1})$ where all subspaces have the same size. then for each subspace S_i, we will derive a variant of E with S_i as input, denoted by E_{S_i}. The ℓ^{th}-order HDL bias can be computed as

$$\mathtt{Bias}\left(\mathsf{Coe}\left(\mathcal{E}, \boldsymbol{\pi}(\boldsymbol{x})\right)\right) = 2^{-\kappa}\mathtt{Bias}\left(\mathsf{Coe}\left(\mathcal{E}_{S_i}, \boldsymbol{\pi}(\boldsymbol{x})\right)\right) \tag{9}$$

In other words, the bias of an HDL is computed as the average value of E over all subspaces.

The partitioning technique is very important to improve the precision, especially for a large order. For example, in our 8^{th}-order HDL distinguishing attacks on 5-round Ascon, if we do not partition the input space, the ANFs in our HATF system will be so complicated that we cannot get the desired biases.

4.5 Conditional HDL Cryptanalysis by Injecting Conditions

In [LLL21], 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 $\alpha_u^{(r)}, r \leq r_0, u > 0$ 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 $\alpha_u^{(r)}, u > \mathbf{0}, r \leq r_0$ are usually reduced to constants, and a system of equations $S = \{f = 0 | f \in I\}$

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. The algorithm for injecting these conditions is provided in Algorithm 3 in Section B of Supplementary Material.

5 Applications to Ascon Initialization

Ascon, designed by Dobraunig, Eichlseder, Mendel, and Schläffer, is a family of AEAD and hash algorithms [DEMS21]. 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 Section A of Supplementary Material, we also recommend that readers refer to [DEMS21] 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 ℓ^{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 [LLL21][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

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

[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.

The 128-bit key and 128-bit nonce are set to 256 binary variables, the IV is set to the constant specified in [DEMS21].

When applying the ℓ^{th}-order HDL distinguishing attack on the Ascon initialization, we choose $\Delta_j, 0 \leq j < \ell$ as the ℓ linearly-independent differences, where Δ_j is active in the two nonce bits of the same Sbox, *i.e.*, $S^{(0)}[3][i_j]$ and $S^{(0)}[4][i_j]$ $(0 \leq i_j < 64)$. Then the input difference can be denoted by an ℓ-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 [DEMS15] 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.* [BDKW19] 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 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

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

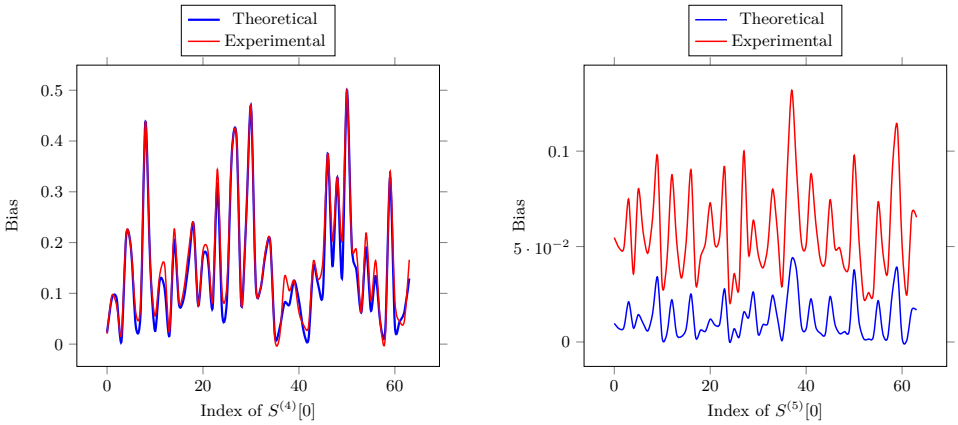
According to the partitioning technique and Equation 9,

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

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,

[6] Under the default setting that $S^{(0)}[3][0] = S^{(0)}[4][0]$, see [DEMS15] for more information about this DL distinguisher.



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

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

the bias from the HATF is always 0, hence we need to partition the space into smaller ones to detect the bias. According to Section 4.5, we can derive a set of 7 conditions that affect the bias significantly. The 7 conditions are provided in Section C of Supplementary Material. 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. 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 bias, this determinant distinguisher is fully verified. We plot the theoretical and experimental biases of the 64 bits of $S^{(4)}[0]$ as shown in Figure 2a, and the concrete data is provided in Table 6 in Section D of Supplementary Material. 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.

Application 4: 8^{th}-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 8^{th}-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 8^{th}-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 Figure 2b, the concrete biases are provided in Table 7 in Section D of Supplementary Material.

We use this 8^{th}-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 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 introduced in Section 2.4. Setting that $\alpha_0 = \alpha_1 = 0.05$, *i.e.*, we require 95% success rate, according to the second equation of Equation 5,

$$35.71N - 32N = \Phi^{-1}(0.95) \cdot \sqrt{16N} + \Phi^{-1}(0.95) \cdot \sqrt{15.79N},$$

thus $N \approx 3^{3.65}$, *i.e.*, we need to check about 801 output bits. According to the first equation of Equation 5, the threshold is $\tau = 35.71N - \Phi^{-1}(0.95) \cdot \sqrt{15.79N} \approx 424$. 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 13^{th} 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 times of 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.

In Section J of Supplementary Material, we provide the figures of the theoretical and experimental results of all 2^{nd}-order HDL approximations for 4- and

5-round Ascon with $\Delta(0,i), 1 \leq i < 64$, as well as the figures from 3^{rd} to 8^{th}-order HDL approximations for 5-round Ascon. Based on these data, we have a detailed discussion on the precision of HATF and the impact of the partitioning technique. We hope that readers can have a better knowledge of the precision of HATF from these figures.

5.2 Conditional HDL Attack for Ascon

Application 5: 2^{nd}-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 2^{nd}-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 [LLL21]. When applying the HATF, we inject the conditions for the first two rounds according to Section 4.5. By exhausting all possible 2^{nd}-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 2^{nd}-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 Section E of Supplementary Material.

Application 6: 3^{rd}-order HDL approximation for 6-round Ascon. At the end of this section, we present a conditional 3^{rd}-order HDL approximation for 6-round Ascon initialization. In [LLL21], 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 3^{rd}-order HDL approximation, we inject 24 conditions into the first two rounds of the HATF. The conditions are provided in Section F of Supplementary Material. 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 6^{th} 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 $\texttt{0x3} \rightarrow \texttt{0x10}$ 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 pilling-up

lemma 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 [DHP+20]. Xoodyak uses Xoodoo as its underlying cryptographic permutation, which is a family of 384-bit to 384-bit permutations [DHAK18b]. 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 ι operation is the i-round constant, which can be found in [DHAK18b]. Note that χ 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.

6.1 HDL Distinguishers for Xoodyak and Xoodoo

Application 1: Revisiting the DL Distinguishers for 4-round Xoodyak. In [DW22], 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 Section 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 $-2^{-5.36}$ [DW22]. 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 Δ_0 that is active in (u_0, u_{128}), Δ_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: 4^{th}-order HDL distinguisher for 4-round Xoodyak. Unlike Ascon, the nonlinear function of Xoodoo (χ) is after the linear operation. Hence if we select low-weight input differences before θ, the input differences into χ are complicated. Thus, we also tried selecting low-weight differences before χ, then we can compute the actual differences back through $(\theta \circ \rho_{west} \circ \iota)^{-1}$. In our 4^{th}-order distinguisher, we choose four differences such as

(Let S be the input state): Δ_0 is active in $(S[0][0][0], S[0][2][0])$, Δ_1 is active in $(S[2][0][7], S[2][2][7])$, Δ_2 is active in $(S[2][0][15], S[2][2][15])$, and Δ_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 $(\theta \circ \rho_{west} \circ \iota)^{-1}(\Delta_0, \Delta_1, \Delta_2, \Delta_3)$, the bias of $S^{(4)}[0][1][1]$ is 2^{-1}. Note that $(\theta \circ \rho_{west} \circ \iota)^{-1}(\Delta_0, \Delta_1, \Delta_2, \Delta_3)$ will be active in all three planes, so this HDL distinguisher is under the related-key model.

Application 4: 2^{nd}-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 2^{nd}-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 Section 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: 2^{nd}- and 3^{rd}-order HDL distinguishers for 4- and 5-round XOODOO. Besides XOODYAK, XOODOO also plays an important role in other schemes such as XOOFFF [DHAK18a]. 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 non-linear operations in the first round. Let S be the input state of the first χ. 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 ℓ^{th}-order HDL attack, we choose the input differences $\Delta(i_0, i_1, \ldots, i_{\ell-1})$ that have 2ℓ active bits of $S[0][0][i_j]$ and $S[0][2][i_j]$.

For 4-round XOODOO, we choose the input difference as $\Delta(0, 20)$, 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 $\Delta(0, 13, 14)$, 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 3^{rd}-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 χ after θ and ρ_{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 Δ_0 is active in (u_0, u_{128}), Δ_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 Section 2.4, 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 [DW22].

Application 7: Theoretical 3^{rd}-order HDL key-recovery attack on 5-round XOODYAK under the single-key model. In [DW22], 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 [ZLD+20]. 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 Δ_0 is active in (u_0, u_{128}), Δ_1 is active in (u_9, u_{137}), and Δ_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 Section G of Supplementary Material. 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 according to Section 2.4. 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 Section 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 $f : \mathbb{F}_2^n \to \mathbb{F}_2$ and an ℓ^{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 ℓ^{th}*-order differential supporting function* (DSF) *of* f *with respect to* $(X, \boldsymbol{\Delta})$*. When the order* ℓ *is clear from context, we will ignore it in the notation, i.e.,* $\mathrm{DSF}_{f,X,\boldsymbol{\Delta}}(\boldsymbol{x})$.

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 2^{64} can only reach 7 rounds. The only integral distinguishers given by Todo [Tod15] 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. 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 ℓ. 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 ℓ^{th}-order difference yielding the following property with probability 1, *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[7]. The 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 [Tod15,TM16] 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*

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

[7] 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.

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 according to Propositions 3 and 4 given in Section H of Supplementary Material. 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[8]. 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 Section 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 ℓ'^{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 [BLNS21]. With $\ell' = 1$, our 64^{th} 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 Δ'_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 use Propositions 3 and 4 to update the degree matrix 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 is smaller than 64, we find useful 64^{th} HD distinguishers for r-round Ascon permutation. The process is illustrated by Algorithm 5 in Section H of Supplementary Material.

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

$$(\bar{X}, \bar{\Delta}) \in \left\{ \begin{matrix} (\texttt{0x6}, \texttt{0x13}), (\texttt{0xa}, \texttt{0x13}), (\texttt{0xc}, \texttt{0x17}), (\texttt{0xf}, \texttt{0x18}), \\ (\texttt{0x15}, \texttt{0x13}), (\texttt{0x17}, \texttt{0x18}), (\texttt{0x19}, \texttt{0x13}), (\texttt{0x1b}, \texttt{0x17}) \end{matrix} \right\}. \tag{10}$$

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 Equation 10. As is seen, for 7

[8] 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 Equation 10. 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

rounds, the degree upper bound of $S^{(7)}[4]$ is only 22, so 2^{23} chosen texts are enough to enforce the zero output difference in this word. We practically verified the algebraic degrees in Table 4 for $(X, \boldsymbol{\Delta}) = (\texttt{0x6}, \texttt{0x13})$ up to 7 rounds. According to Propositions 3 and 4, the degree upper bounds in Table 4 for 8 rounds is also verified.

Therefore, if we choose 2^{46} plaintexts in any 46-dimensional affine space defined by values in Equation 10, the summation of all ciphertexts will be zero with probability of 1. Given a random permutation, the probability that the summation of such 2^{46} ciphertexts will be zero is only 2^{-64}. Thus, 2^{46} chosen plaintexts are enough to distinguish the 8-round Ascon permutation from a random permutation.

Applying a similar method to the inverse of Ascon permutation, we can obtain a zero-sum distinguisher for full-round Ascon permutation with 2^{55} time/-data complexities. Due to the page limit, we provide this part in Section I of Supplementary Material. We stress that these two distinguishers work for the Ascon permutation rather than the keyed mode, so it does not threaten the security of Ascon-AEAD or Ascon-Hash.

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 ℓ^{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 Section J). 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.

References

AFK+08. Jean-Philippe Aumasson, Simon Fischer, Shahram Khazaei, Willi Meier, and Christian Rechberger. New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. In Kaisa Nyberg, editor, *Fast Software Encryption - FSE 2008*, volume 5086 of *LNCS*, pages 470–488. Springer, 2008.

AM09. Jean-Philippe Aumasson and Willi Meier. Zero-sum distinguishers for reduced keccak-f and for the core functions of luffa and hamsi. *rump session of Cryptographic Hardware and Embedded Systems-CHES*, 2009:67, 2009.

BAK98. Eli Biham, Ross J. Anderson, and Lars R. Knudsen. Serpent: A New Block Cipher Proposal. In Serge Vaudenay, editor, *Fast Software Encryption - FSE '98*, volume 1372 of *LNCS*, pages 222–238. Springer, 1998.

BC10. Christina Boura and Anne Canteaut. A zero-sum property for the keccak-f permutation with 18 rounds. In *IEEE International Symposium on Information Theory, ISIT 2010, June 13-18, 2010, Austin, Texas, USA, Proceedings*, pages 2488–2492. IEEE, 2010.

BCP22. Jules Baudrin, Anne Canteaut, and Léo Perrin. Practical cube-attack against nonce-misused ascon. In *NIST Lightweight Cryptography Workshop*, 2022.

BDK02. Eli Biham, Orr Dunkelman, and Nathan Keller. Enhancing Differential-Linear Cryptanalysis. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 254–266. Springer, 2002.

BDK05. Eli Biham, Orr Dunkelman, and Nathan Keller. New Combined Attacks on Block Ciphers. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption - FSE 2005*, volume 3557 of *LNCS*, pages 126–144. Springer, 2005.

BDK07. Eli Biham, Orr Dunkelman, and Nathan Keller. A New Attack on 6-Round IDEA. In Alex Biryukov, editor, *Fast Software Encryption - FSE 2007*, volume 4593 of *LNCS*, pages 211–224. Springer, 2007.

BDKW19. Achiya Bar-On, Orr Dunkelman, Nathan Keller, and Ariel Weizman. DLCT: A New Tool for Differential-Linear Cryptanalysis. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019*, volume 11476 of *LNCS*, pages 313–342. Springer, 2019.

BDP+18. Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, Ronny Van Keer, and Benoît Viguier. Kangarootwelve: Fast hashing based on keccak-p. In Bart Preneel and Frederik Vercauteren, editors, *Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings*, volume 10892 of *Lecture Notes in Computer Science*, pages 400–418. Springer, 2018.

BLN17. Céline Blondeau, Gregor Leander, and Kaisa Nyberg. Differential-Linear Cryptanalysis Revisited. *J. Cryptol.*, 30(3):859–888, 2017.

BLNS21. X. Bonnetain, G. Leurent, M. Naya-Plasencia, and A. Schrottenloher. Quantum linearization attacks. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021*, volume 13090 of *Lecture Notes in Computer Science*, pages 422–452. Springer, 2021.

BS90. Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90*, volume 537 of *LNCS*, pages 2–21. Springer, 1990.

CHK22. Donghoon Chang, Deukjo Hong, and Jinkeon Kang. Conditional cube attacks on ascon-128 and ascon-80pq in a nonce-misuse setting. *IACR Cryptol. ePrint Arch.*, page 544, 2022.

CKT+22. Donghoon Chang, Jinkeon Kang, Meltem Sönmez Turan, et al. A new conditional cube attack on reduced-round ascon-128a in a nonce-misuse setting. NIST LWC Workshop, May 2022.

DEMS15. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Cryptanalysis of Ascon. In Kaisa Nyberg, editor, *Topics in Cryptology - CT-RSA 2015*, volume 9048 of *LNCS*, pages 371–387. Springer, 2015.

DEMS21. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight Authenticated Encryption and Hashing. *J. Cryptol.*, 34(3):33, 2021.

DHAK18a. Joan Daemen, Seth Hoffert, Gilles Van Assche, and Ronny Van Keer. The design of xoodoo and xoofff. *IACR Trans. Symmetric Cryptol.*, 2018(4):1–38, 2018.

DHAK18b. Joan Daemen, Seth Hoffert, Gilles Van Assche, and Ronny Van Keer. Xoodoo cookbook. *IACR Cryptol. ePrint Arch.*, page 767, 2018.

DHP+20. Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Xoodyak, a lightweight cryptographic scheme. *IACR Trans. Symmetric Cryptol.*, 2020(S1):60–87, 2020.

DS09. Itai Dinur and Adi Shamir. Cube Attacks on Tweakable Black Box Polynomials. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 278–299. Springer, 2009.

DW22. Orr Dunkelman and Ariel Weizman. Differential-linear cryptanalysis on xoodyak. In *NIST Lightweight Cryptography Workshop*, 2022.

GPT21. David Gérault, Thomas Peyrin, and Quan Quan Tan. Exploring Differential-Based Distinguishers and Forgeries for ASCON. *IACR Trans. Symmetric Cryptol.*, 2021(3):102–136, 2021.

Kec. Keccak Team. Note on zero-sum distinguishers of Keccak-f, https://keccak.team/files/NoteZeroSum.pdf.

Knu94. Lars R. Knudsen. Truncated and Higher Order Differentials. In Bart Preneel, editor, *Fast Software Encryption - FSE'94*, volume 1008 of *LNCS*, pages 196–211. Springer, 1994.

Lai94. Xuejia Lai. Higher order derivatives and differential cryptanalysis. In *Communications and cryptography*, pages 227–233. Springer, 1994.

LDW17. Zheng Li, Xiaoyang Dong, and Xiaoyun Wang. Conditional Cube Attack on Round-Reduced ASCON. *IACR Trans. Symmetric Cryptol.*, 2017(1):175–202, 2017.

LH94. Susan K. Langford and Martin E. Hellman. Differential-Linear Cryptanalysis. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94*, volume 839 of *LNCS*, pages 17–25. Springer, 1994.

LLL21. Meicheng Liu, Xiaojuan Lu, and Dongdai Lin. Differential-Linear Cryptanalysis from an Algebraic Perspective. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021*, volume 12827 of *LNCS*, pages 247–277. Springer, 2021.

LM90. Xuejia Lai and James L. Massey. A Proposal for a New Block Encryption Standard. In Ivan Damgård, editor, *Advances in Cryptology - EUROCRYPT '90*, volume 473 of *LNCS*, pages 389–404. Springer, 1990.

LSL21. Yunwen Liu, Siwei Sun, and Chao Li. Rotational Cryptanalysis from a Differential-Linear Perspective - Practical distinguishers for round-reduced friet, xoodoo, and alzette. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021*, volume 12696 of *LNCS*, pages 741–770. Springer, 2021.

Mat93. Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93*, volume 765 of *LNCS*, pages 386–397. Springer, 1993.

SSS+19. Danping Shi, Siwei Sun, Yu Sasaki, Chaoyun Li, and Lei Hu. Correlation of quadratic boolean functions: Cryptanalysis of all versions of full \mathsf MORUS. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 180–209. Springer, 2019.

TM16. Yosuke Todo and Masakatu Morii. Bit-Based Division Property and Application to Simon Family. *IACR Cryptol. ePrint Arch.*, page 285, 2016.

Tod15. Yosuke Todo. Structural Evaluation by Generalized Integral Property. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 287–314. Springer, 2015.

Vau98. Serge Vaudenay. Provable Security for Block Ciphers by Decorrelation. In Michel Morvan, Christoph Meinel, and Daniel Krob, editors, *Annual Symposium on Theoretical Aspects of Computer Science - STACS 98*, volume 1373 of *LNCS*, pages 249–275. Springer, 1998.

Wag99. David A. Wagner. The Boomerang Attack. In Lars R. Knudsen, editor, *Fast Software Encryption - FSE '99*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.

WGR18. Qingju Wang, Lorenzo Grassi, and Christian Rechberger. Zero-sum partitions of PHOTON permutations. In Nigel P. Smart, editor, *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*, volume 10808 of *Lecture Notes in Computer Science*, pages 279–299. Springer, 2018.

YLW+19. Hailun Yan, Xuejia Lai, Lei Wang, Yu Yu, and Yiran Xing. New zero-sum distinguishers on full 24-round keccak-f using the division property. *IET Inf. Secur.*, 13(5):469–478, 2019.

ZLD+20. Haibo Zhou, Zheng Li, Xiaoyang Dong, Keting Jia, and Willi Meier. Practical key-recovery attacks on round-reduced ketje jr, xoodoo-ae and xoodyak. *Comput. J.*, 63(8):1231–1246, 2020.

Supplementary Material

A Brief Specification of ASCON

ASCON, designed by Dobraunig, Eichlseder, Mendel, and Schläffer, is a family of AEAD and hash algorithms. At a high level, the ASCON AEAD takes as input a nonce N, a secret key K, an associated data A and a plaintext or message M, and produces a ciphertext C and a tag T. The authenticity of the associated data and message can be verified against the tag T. Table 5 lists the variants of ASCON AEAD along with the recommended parameter sets.

Table 5: ASCON variants and their recommended parameters

Name	State size	Rate r	Size of			Rounds	
			Key	Nonce	Tag	p^a	p^b
ASCON-128	320	64	128	128	128	12	6
ASCON-128a	320	128	128	128	128	12	8

ASCON adopts a MonkeyDuplex mode with a stronger keyed initialization and keyed finalization phases as illustrated in Figure 3. The underlying permutations p^a and p^b are iterative designs, whose round function p is based on the substitution permutation network design paradigm and consists of three simple steps p_C, p_S, and p_L. We now describe the round function p and each step in detail.

The round function $p = p_L \circ p_S \circ p_C$ operates on a 320-bit state arranged into five 64-bit words. The input state to the round function at r^{th} round is denoted by $S^{(r)} = S^{(r)}[0]\|S^{(r)}[1]\|S^{(r)}[2]\|S^{(r)}[3]\|S^{(r)}[4]$, the j^{th} bit of $S^{(r)}[i]$ is denoted by $S^{(r)}[i][j]$ where $0 \leq i < 5, 0 \leq j < 64$. We use $S^{(r.5)}$ to represent the state after p_S of the r^{th} round, $r \geq 0$.

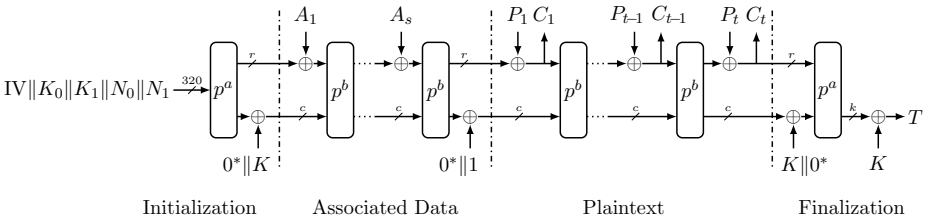


Fig. 3: The encryption algorithm of ASCON

Addition of constants (p_C). An 8-bit constant is XORed to the bit positions $56, \cdots, 63$ of the 64-bit word $S^{(r)}[2]$ at each round.

Substitution layer (p_S). Update each slice of the 320-bit state by applying the 5-bit Sbox $\mathcal{S} : \mathbb{F}_2^5 \to \mathbb{F}_2^5$ defined by the following algebraic normal forms:

$$\begin{cases} y_0 = x_4x_1 + x_3 + x_2x_1 + x_2 + x_1x_0 + x_1 + x_0 \\ y_1 = x_4 + x_3x_2 + x_3x_1 + x_3 + x_2x_1 + x_2 + x_1 + x_0 \\ y_2 = x_4x_3 + x_4 + x_2 + x_1 + 1 \\ y_3 = x_4x_0 + x_4 + x_3x_0 + x_3 + x_2 + x_1 + x_0 \\ y_4 = x_4x_1 + x_4 + x_3 + x_1x_0 + x_1 \end{cases} \tag{11}$$

The ANF of the inverse of the Sbox is as follows,

$$\begin{cases} y_0 = x_4x_3x_2 + x_4x_3x_1 + x_4x_3x_0 + x_3x_2x_0 + x_3x_2 + x_3 + x_2 + x_1x_0 + x_1 + 1 \\ y_1 = x_4x_2x_0 + x_4 + x_3x_2 + x_2x_0 + x_1 + x_0 \\ y_2 = x_4x_3x_1 + x_4x_3 + x_4x_2x_1 + x_4x_2x_0 + x_4x_2 + x_4 + x_3x_2 + x_3x_1x_0 \\ \qquad\qquad + x_3x_1 + x_2x_1x_0 + x_2x_1 + x_2x_0 + x_2 + x_1 + x_0 + 1 \\ y_3 = x_4x_2x_1 + x_4x_2x_0 + x_4x_2 + x_4x_1 + x_4 + x_3 + x_2x_1 + x_2x_0 + x_1 \\ y_4 = x_4x_3x_2 + x_4x_2x_1 + x_4x_2x_0 + x_4x_2 + x_3x_2x_0 + x_3x_2 + x_3 + x_2x_1 + x_2x_0 + x_1x_0 \end{cases} \tag{12}$$

Linear diffusion layer (p_L). Apply a linear transformation Σ_i to each 64-bit word $S^{r.5}[i]$ with $0 \leq i < 5$, where Σ_i is defined as

$$\begin{cases} y_0 \leftarrow \Sigma_0(x_0) = x_0 + (x_0 \ggg 19) + (x_0 \ggg 28) \\ y_1 \leftarrow \Sigma_1(x_1) = x_1 + (x_1 \ggg 61) + (x_1 \ggg 39) \\ y_2 \leftarrow \Sigma_2(x_2) = x_2 + (x_2 \ggg \ 1) + (x_2 \ggg \ 6) \\ y_3 \leftarrow \Sigma_3(x_3) = x_3 + (x_3 \ggg 10) + (x_3 \ggg 17) \\ y_4 \leftarrow \Sigma_4(x_4) = x_4 + (x_4 \ggg \ 7) + (x_4 \ggg 41) \end{cases} \tag{13}$$

In this paper, when we attack r rounds of the Ascon permutation, we can operate all 320 input bits S^0 and observe all 320 output bits of $S^{(r)}$ or $S^{(r.5)}$. When we attack r rounds of the Ascon initialization, we can operate only $S^{(0)}[3]$ and $S^{(0)}[4]$ and observe $S^r[0]$.

B Algorithm for Injecting Conditions into HATF

Algorithm 3 Injecting conditions into the first R_0 rounds of the HATF of $\mathcal{E}^\ell$ from a cipher E

Input: 1. the ANFs of components of $E = E_{R-1} \circ \cdots E_0$,
2. the order ℓ,
3. the block size n,
4. an ℓ^{th}-order difference $(\Delta_0, \ldots, \Delta_{\ell-1})$,
5. an input value X
6. the round R_0

Output: a set of conditions I

Let $\alpha_0^{(0)} = X$, $\alpha_{e_i}^{(0)} = \Delta_i$, $\Delta_u^{(0)} = 0$ for all $wt(u) \geq 2$
$I \leftarrow \varnothing$
for $1 \leq r \leq R_0$ **do**
  Calculate $E_r\left(\bigoplus_{u \in \mathbb{F}_2^n} \alpha_u^{(r)} \boldsymbol{x}^u\right)$
  for $0 \leq i < n$ **do**
    for $0 \leq u < 2^\ell$ **do**
      if $\mathsf{Coe}\left(E_r\left(\bigoplus_{u \in \mathbb{F}_2^n} \alpha_u^{(r)} \boldsymbol{x}^u\right), \boldsymbol{x}^u\right) \notin \{0, 1\}$ **then**
        $I \leftarrow \mathsf{Coe}\left(E_r\left(\bigoplus_{u \in \mathbb{F}_2^n} \alpha_u^{(r)} \boldsymbol{x}^u\right), \boldsymbol{x}^u\right)$
        for $0 \leq i < n$ **do** ▷ Reduce all the coefficients
          $E_r\left(\bigoplus_{u \in \mathbb{F}_2^n} \alpha_u^{(r)} \boldsymbol{x}^u\right) = E_r\left(\bigoplus_{u \in \mathbb{F}_2^n} \alpha_u^{(r)} \boldsymbol{x}^u\right) \boldsymbol{x}^u \mod I$
        end for
      end if
    end for
  end for
end for
return I

C 7 Conditions Used for Partitioning in the DL Distinguisher for 5-round Ascon. ASCON Initialization

$$
\begin{aligned}
k_{64} &= 0 \\
k_0 &= 0 \\
u_{86} &= u_{19}k_{19} \oplus u_{19}k_{83} \oplus u_{19} \oplus u_{22}k_{22} \oplus u_{22}k_{86} \oplus u_{22} \oplus u_{44}k_{44} \oplus u_{44}k_{108} \oplus u_{44} \\
&\quad \oplus u_{83} \oplus u_{108} \oplus k_{19}k_{83} \oplus k_{19} \oplus k_{22}k_{86} \oplus k_{22} \oplus k_{44}k_{108} \oplus k_{44} \oplus k_{83} \oplus k_{86} \\
&\quad \oplus k_{108} \\
u_{23} &= u_{47} \oplus u_{54} \oplus u_{57} \oplus u_{87}k_{23} \oplus u_{87} \oplus u_{111} \oplus u_{118} \oplus u_{121}k_{57} \oplus u_{121} \oplus k_{23} \\
&\quad \oplus k_{47} \oplus k_{54} \oplus k_{57} \oplus k_{111} \oplus k_{118} \\
u_{95} &= u_{28}k_{28} \oplus u_{28}k_{92} \oplus u_{28} \oplus u_{31}k_{31} \oplus u_{31}k_{95} \oplus u_{31} \oplus u_{53}k_{53} \oplus u_{53}k_{117} \oplus u_{53} \\
&\quad \oplus u_{92} \oplus u_{117} \oplus k_{28}k_{92} \oplus k_{28} \oplus k_{31}k_{95} \oplus k_{31} \oplus k_{53}k_{117} \oplus k_{53} \oplus k_{92} \oplus k_{95} \\
&\quad \oplus k_{117} \\
u_{67} &= u_3k_3 \oplus u_3k_{67} \oplus u_3 \oplus u_{25}k_{25} \oplus u_{25}k_{89} \oplus u_{25} \oplus u_{89} \oplus k_3k_{67} \oplus k_3 \oplus k_{25}k_{89} \\
&\quad \oplus k_{25} \oplus k_{67} \oplus k_{89} \oplus 1 \\
u_2 &= u_{12} \oplus u_{42} \oplus u_{66} \oplus u_{76}k_{12} \oplus u_{76} \oplus u_{83}k_{19} \oplus u_{106}k_{42} \oplus u_{106} \oplus k_2 \oplus k_9 \oplus k_{12} \\
&\quad \oplus k_{42} \oplus k_{66} \oplus k_{73} \oplus k_{83}
\end{aligned}
$$

D Tables of Biases in Second-Order and 8^{th}-Order HDL Distinguishers for Ascon

Table 6: Theoretical and experimental biases of 4-round Ascon initialization with the input difference $\Delta(0, 60)$. Note since our experiments are done with 2^{26} samples, only those experimental biases that are significantly greater than 2^{-13} are reliable.

Bit	0	1	2	3	4	5	6	7	8	9
Theory	$2^{-5.45}$	$2^{-3.48}$	$2^{-3.48}$	$2^{-7.49}$	$2^{-2.22}$	$2^{-2.42}$	$2^{-4.87}$	$2^{-3.97}$	$2^{-1.19}$	$2^{-2.68}$
Expr.	$2^{-5.60}$	$2^{-3.39}$	$2^{-3.71}$	$2^{-5.73}$	$2^{-2.19}$	$2^{-2.42}$	$2^{-4.03}$	$2^{-3.63}$	$2^{-1.19}$	$2^{-2.68}$
Bit	10	11	12	13	14	15	16	17	18	19
Theory	$2^{-5.27}$	$2^{-2.99}$	$2^{-3.19}$	$2^{-5.90}$	$2^{-2.30}$	$2^{-3.68}$	$2^{-3.46}$	$2^{-2.66}$	$2^{-2.07}$	$2^{-3.71}$
Expr.	$2^{-4.17}$	$2^{-2.86}$	$2^{-2.68}$	$2^{-5.34}$	$2^{-2.14}$	$2^{-3.61}$	$2^{-3.22}$	$2^{-2.45}$	$2^{-2.07}$	$2^{-3.71}$
Bit	20	21	22	23	24	25	26	27	28	29
Theory	$2^{-2.52}$	$2^{-2.57}$	$2^{-3.81}$	$2^{-1.68}$	$2^{-4.08}$	$2^{-3.50}$	$2^{-1.42}$	$2^{-1.30}$	$2^{-3.71}$	$2^{-2.02}$
Expr.	$2^{-2.44}$	$2^{-2.43}$	$2^{-3.57}$	$2^{-1.54}$	$2^{-3.36}$	$2^{-3.07}$	$2^{-1.42}$	$2^{-1.30}$	$2^{-3.72}$	$2^{-2.02}$
Bit	30	31	32	33	34	35	36	37	38	39
Theory	$2^{-1.09}$	$2^{-3.09}$	$2^{-3.19}$	$2^{-2.51}$	$2^{-2.30}$	$2^{-6.00}$	$2^{-4.95}$	$2^{-3.62}$	$2^{-3.69}$	$2^{-3.00}$
Expr.	$2^{-1.09}$	$2^{-3.09}$	$2^{-3.19}$	$2^{-2.45}$	$2^{-2.30}$	$2^{-6.67}$	$2^{-5.91}$	$2^{-2.91}$	$2^{-3.25}$	$2^{-3.00}$
Bit	40	41	42	43	44	45	46	47	48	49
Theory	$2^{-3.54}$	$2^{-5.38}$	$2^{-6.61}$	$2^{-2.71}$	$2^{-3.09}$	$2^{-3.31}$	$2^{-1.42}$	$2^{-2.70}$	$2^{-1.61}$	$2^{-2.93}$
Expr.	$2^{-4.00}$	$2^{-4.76}$	$2^{-4.84}$	$2^{-2.62}$	$2^{-2.98}$	$2^{-2.60}$	$2^{-1.42}$	$2^{-2.29}$	$2^{-1.61}$	$2^{-2.29}$
Bit	50	51	52	53	54	55	56	57	58	59
Theory	$2^{-1.00}$	$2^{-2.36}$	$2^{-2.79}$	$2^{-4.00}$	$2^{-2.40}$	$2^{-3.92}$	$2^{-2.90}$	$2^{-4.68}$	$2^{-5.14}$	$2^{-1.61}$
Expr.	$2^{-1.00}$	$2^{-2.34}$	$2^{-2.45}$	$2^{-4.00}$	$2^{-2.19}$	$2^{-3.54}$	$2^{-2.61}$	$2^{-4.68}$	$2^{-6.20}$	$2^{-1.56}$
Bit	60	61	62	63						
Theory	$2^{-4.81}$	$2^{-4.48}$	$2^{-3.96}$	$2^{-2.95}$						
Expr.	$2^{-3.54}$	$2^{-4.48}$	$2^{-4.42}$	$2^{-2.59}$						

Table 7: Theoretical and experimental biases of 5-round Ascon initialization with the input difference $\Delta(0, 8, 9, 13, 14, 26, 43, 60)$. Note since our experiments are done with 2^{26} samples, only those experimental biases that are significantly greater than 2^{-13} are reliable.

Bit	0	1	2	3	4	5	6	7	8	9
Theory	$2^{-6.68}$	$2^{-7.17}$	$2^{-7.05}$	$2^{-5.57}$	$2^{-7.10}$	$2^{-6.13}$	$2^{-6.67}$	$2^{-7.42}$	$2^{-6.06}$	$2^{-4.88}$
Expr.	$2^{-4.19}$	$2^{-4.34}$	$2^{-4.34}$	$2^{-3.74}$	$2^{-4.81}$	$2^{-3.64}$	$2^{-4.11}$	$2^{-4.42}$	$2^{-3.94}$	$2^{-3.36}$
Bit	10	11	12	13	14	15	16	17	18	19
Theory	$2^{-9.12}$	$2^{-7.74}$	$2^{-5.50}$	$2^{-7.95}$	$2^{-8.44}$	$2^{-7.14}$	$2^{-5.32}$	$2^{-8.87}$	$2^{-7.32}$	$2^{-7.48}$
Expr.	$2^{-5.09}$	$2^{-4.52}$	$2^{-3.51}$	$2^{-4.25}$	$2^{-4.90}$	$2^{-4.24}$	$2^{-3.47}$	$2^{-5.05}$	$2^{-4.48}$	$2^{-4.27}$
Bit	20	21	22	23	24	25	26	27	28	29
Theory	$2^{-6.39}$	$2^{-6.82}$	$2^{-6.69}$	$2^{-5.17}$	$2^{-11.49}$	$2^{-7.17}$	$2^{-8.69}$	$2^{-6.00}$	$2^{-6.30}$	$2^{-5.26}$
Expr.	$2^{-3.78}$	$2^{-4.30}$	$2^{-4.16}$	$2^{-3.45}$	$2^{-5.51}$	$2^{-4.80}$	$2^{-5.12}$	$2^{-3.32}$	$2^{-4.45}$	$2^{-3.97}$
Bit	30	31	32	33	34	35	36	37	38	39
Theory	$2^{-7.94}$	$2^{-6.78}$	$2^{-6.67}$	$2^{-5.35}$	$2^{-6.29}$	$2^{-10.32}$	$2^{-5.50}$	$2^{-4.52}$	$2^{-4.75}$	$2^{-6.88}$
Expr.	$2^{-4.45}$	$2^{-4.69}$	$2^{-4.32}$	$2^{-3.64}$	$2^{-4.10}$	$2^{-5.09}$	$2^{-3.53}$	$2^{-2.92}$	$2^{-3.47}$	$2^{-4.25}$
Bit	40	41	42	43	44	45	46	47	48	49
Theory	$2^{-7.18}$	$2^{-5.47}$	$2^{-6.98}$	$2^{-7.78}$	$2^{-7.87}$	$2^{-5.39}$	$2^{-6.74}$	$2^{-7.83}$	$2^{-7.53}$	$2^{-7.34}$
Expr.	$2^{-4.29}$	$2^{-3.51}$	$2^{-4.00}$	$2^{-4.55}$	$2^{-4.56}$	$2^{-3.75}$	$2^{-4.34}$	$2^{-4.34}$	$2^{-4.70}$	$2^{-4.57}$
Bit	50	51	52	53	54	55	56	57	58	59
Theory	$2^{-4.73}$	$2^{-6.52}$	$2^{-9.17}$	$2^{-9.35}$	$2^{-8.56}$	$2^{-5.53}$	$2^{-7.90}$	$2^{-8.69}$	$2^{-5.25}$	$2^{-4.69}$
Expr.	$2^{-3.35}$	$2^{-4.21}$	$2^{-5.44}$	$2^{-5.28}$	$2^{-5.29}$	$2^{-3.77}$	$2^{-4.42}$	$2^{-4.77}$	$2^{-3.45}$	$2^{-3.14}$
Bit	60	61	62	63						
Theory	$2^{-8.80}$	$2^{-9.87}$	$2^{-5.90}$	$2^{-5.90}$						
Expr.	$2^{-4.30}$	$2^{-5.34}$	$2^{-3.90}$	$2^{-3.94}$						

E Details of HDL Key-Recovery attack on the 5-Round Ascon Initialization

Although Ascon initialization is not strictly rotation-invariant due to the constant additions and IV, the biases truly follow some rotation-invariant properties according to our experiments. Thus, in the following we only discuss the $\Delta(0, 9)$ case as an example. It can be easily adapted to other cases.

Key-recovery process. From the condition injection, we obtain 14 conditions, 3 among them are found a bit redundant so we remove them without affecting the bias significantly. The 11 conditions are given as follows,

$$k_0 = 0 \tag{14}$$
$$k_{64} = 0 \tag{15}$$
$$k_9 = 0 \tag{16}$$
$$k_{73} = 0 \tag{17}$$
$$\begin{aligned} u_{86} = {} & u_{19}k_{19} \oplus u_{19}k_{83} \oplus u_{19} \oplus u_{22}k_{22} \oplus u_{22}k_{86} \oplus u_{22} \oplus u_{44}k_{44} \oplus u_{44}k_{108} \oplus u_{44} \\ & \oplus u_{83} \oplus u_{108} \oplus k_{19}k_{83} \oplus k_{19} \oplus k_{22}k_{86} \oplus k_{22} \oplus k_{44}k_{108} \oplus k_{44} \oplus k_{83} \oplus k_{86} \\ & \oplus k_{108} \end{aligned} \tag{18}$$
$$\begin{aligned} u_{23} = {} & u_{47} \oplus u_{54} \oplus u_{57} \oplus u_{87}k_{23} \oplus u_{87} \oplus u_{111} \oplus u_{118} \oplus u_{121}k_{57} \oplus u_{121} \oplus k_{23} \\ & \oplus k_{47} \oplus k_{54} \oplus k_{57} \oplus k_{111} \oplus k_{118} \end{aligned} \tag{19}$$
$$\begin{aligned} u_{98} = {} & u_{12}k_{12} \oplus u_{12}k_{76} \oplus u_{12} \oplus u_{34}k_{34} \oplus u_{34}k_{98} \oplus u_{34} \oplus u_{76} \oplus k_{12}k_{76} \oplus k_{12} \\ & \oplus k_{34}k_{98} \oplus k_{34} \oplus k_{76} \oplus k_{98} \oplus 1 \end{aligned} \tag{20}$$
$$\begin{aligned} u_{42} = {} & u_2 \oplus u_{12} \oplus u_{66} \oplus u_{76}k_{12} \oplus u_{76} \oplus u_{83}k_{19} \oplus u_{106}k_{42} \oplus u_{106} \oplus k_2 \oplus k_{12} \oplus k_{42} \\ & \oplus k_{66} \oplus k_{83} \end{aligned} \tag{21}$$
$$\begin{aligned} u_{95} = {} & u_{28}k_{28} \oplus u_{28}k_{92} \oplus u_{28} \oplus u_{31}k_{31} \oplus u_{31}k_{95} \oplus u_{31} \oplus u_{53}k_{53} \oplus u_{53}k_{117} \oplus u_{53} \\ & \oplus u_{92} \oplus u_{117} \oplus k_{28}k_{92} \oplus k_{28} \oplus k_{31}k_{95} \oplus k_{31} \oplus k_{53}k_{117} \oplus k_{53} \oplus k_{92} \oplus k_{95} \\ & \oplus k_{117} \end{aligned} \tag{22}$$
$$\begin{aligned} u_{11} = {} & u_{18} \oplus u_{21} \oplus u_{51} \oplus u_{75} \oplus u_{82} \oplus u_{85}k_{21} \oplus u_{85} \oplus u_{92}k_{28} \oplus u_{115}k_{51} \oplus u_{115} \\ & \oplus k_{11} \oplus k_{18} \oplus k_{51} \oplus k_{75} \oplus k_{82} \oplus k_{92} \oplus 1 \end{aligned} \tag{23}$$
$$\begin{aligned} u_{32} = {} & u_2 \oplus u_{56} \oplus u_{63} \oplus u_{66}k_2 \oplus u_{66} \oplus u_{96}k_{32} \oplus u_{96} \oplus u_{120} \oplus u_{127} \oplus k_2 \oplus k_{32} \\ & \oplus k_{56} \oplus k_{63} \oplus k_{120} \oplus k_{127} \oplus 1 \end{aligned} \tag{24}$$

When all these 11 conditions hold, the bias is about 0.373. When the Condition 22 or Condition 24 doesn't hold, the bias is about 0.287. In other cases, the biases are significantly smaller. Especially, if any of Conditions 14–17 does not hold, the bias is close to zero. Based on these conditions and biases, the key-recovery attack is performed as follows,

1. Note that we do not have means to control Conditions 14–17, so that we have to check if $k_0 = k_{64} = k_9 = k_{73} = 0$ in the first step. For Conditions 18–24,

we first fix all the nonce variables in the right parts as 0, *i.e.*, variables in

$$V_r = \left\{ \begin{array}{r} u_{120}, u_{19}, u_{127}, u_{118}, u_{121}, u_{12}, u_{56}, u_{87}, u_{85}, u_{76}, u_{44}, u_{82}, u_{22}, u_{54}, \\ u_{34}, u_{57}, u_{117}, u_{31}, u_{92}, u_{53}, u_{51}, u_{18}, u_{111}, u_{66}, u_{47}, u_2, u_{21}, u_{83}, \\ u_{106}, u_{28}, u_{96}, u_{108}, u_{115}, u_{75}, u_{63} \end{array} \right\}$$

will be fixed as 0. Next we traverse all possible assignments for the left variables, *i.e.*, variables in

$$V_l = \left\{ u_{86}, u_{23}, u_{98}, u_{42}, u_{95}, u_{11}, u_{32} \right\}.$$

There must be one assignment of V_l making Conditions 18–24 hold. If Conditions 14–17 also hold ($k_0 = k_9 = k_{64} = k_{73} = 0$), we will see a high bias in $S^{(5)}[0][27]$, *i.e.*, 0.373. Otherwise, for all assignments of V_l, the bias of $S^{(5)}[0][27]$ will be close to 0.
To detect this bias, we do a statistical test. Suppose that we use N samples and denote the times of $S^{(5)}[0][27] = 0$ by T.
When $k_0 = k_{64} = k_9 = k_{73} = 0$, for at least one assignment of V_l,

$$T \sim \mathcal{N}_0(N \times 0.373, N \times 0.373 \times (1 - 0.373)).$$

If $k_0 = k_{64} = k_9 = k_{73} = 0$ is not true, for all assignments,

$$T \sim \mathcal{N}_1(N \times 0.5, N \times 0.25).$$

We use the theory of Section 2.4 to distinguish the two normal distributions $\mathcal{N}_0$ and $\mathcal{N}_1$. Let α_0, α_1 be the probability of the Type-1 and Type-2 errors, respectively. We set the Type-2 error as $\alpha_1 = 0.05$ (with 95% probability, the right assignment can be identified). To make sure that all wrong assignments are identified with probability 95%, α_0 needs to satisfy $1-(1-\alpha_0)^{128} = 0.05$, thus $\alpha_0 \approx 2^{-11.285}$. According to Equation 5, we need $N = 2^{5.15}$ samples, the threshold is $\tau = 28$. In other words, for each of the 2^7 assignments, we encrypt $2^{5.15}$ samples. If there is one assignment such that the number of $S^{(5)}[0][27] = 0$ is larger than 28, we determine $k_0 = k_{64} = k_9 = k_{73} = 0$. Otherwise, $k_0 = k_{64} = k_9 = k_{73} = 0$ is not true.

2. If $k_0 = k_{64} = k_9 = k_{73} = 0$ is not true, we cannot proceed with key-recovery attack. We have to check other input differences $\Delta(i, i+9)$ until we find one satisfying $k_i = k_{i+64} = k_{i+9} = k_{i+73} = 0$. Due to the rotation-invariant property of Ascon (in terms of HDL attacks), we can try at most 64 possible $\Delta(i, i+9)$ for $0 \leq i < 64$. Since the probability of $k_i = k_{i+64} = k_{i+9} = k_{i+73} = 0$ is 2^{-4}, so on average, we will detect 4 input differences that satisfy our requirements. Our experiments with 2^{26} randomly-generated keys show that with about 54% probability, there will be 4 input differences satisfying $k_i = k_{i+64} = k_{i+9} = k_{i+73} = 0$ (we can improve this probability by using more HDL approximations later). In the following, we assume $k_0 = k_{64} = k_9 = k_{73} = 0$ to discuss the key recovery attack.

3. When we have known $k_0 = k_{64} = k_9 = k_{73} = 0$, we need to find out the exact one right assignment making Conditions 18–24 hold. We again do the statistical test as above step. In Step 2, only those assignments of V_l that surpass the threshold will be examined in this step. If not all conditions are satisfied, the bias of $S^{(5)}[0][27]$ is at most 0.287. Thus, for a wrong assignment, $T \sim \mathcal{N}_2(N \times 0.287, N \times 0.287 \times (1 - 0.287))$. To distinguish $\mathcal{N}_0$ and $\mathcal{N}_2$, we need $2^{8.96}$ samples with the threshold being 423.
   Once we obtain the right assignment of V_l, we simultaneously obtain 11 bits of key information (including $k_0 = k_{64} = k_9 = k_{73} = 0$).
4. Furthermore, we can obtain more key information. Note that among the 35 variables in V_r, 18 variables are multiplied with certain key bits in Conditions 18–24 (these variables have been labeled by red). By flipping each of the 18 variables (recall they are set as 0 in the first step) and observing whether the bias is still 0.373, we can obtain 18 more key equations. For example, in the Step 2, we have known proper u_{86} satisfying Condition 18 with all nonce bits in the right side being 0, *i.e.*,

   $$u_{86} = k_{19}k_{83} \oplus k_{19} \oplus k_{22}k_{86} \oplus k_{22} \oplus k_{44}k_{108} \oplus k_{44} \oplus k_{83} \oplus k_{86} \oplus k_{108}$$

   Next, we flip u_{19} from 0 to 1, then the equation becomes to

   $$u_{86} = u_{19}k_{19} \oplus u_{19}k_{83} \oplus u_{19} \oplus k_{19}k_{83} \oplus k_{19} \oplus k_{22}k_{86} \oplus k_{22} \oplus k_{44}k_{108} \oplus k_{44} \oplus k_{83} \oplus k_{86} \oplus k_{108}$$

   If $k_{19} \oplus k_{83} \oplus 1 = 0$, u_{19} will not affect Condition 18, so the bias is still high. However, if $k_{19} \oplus k_{83} \oplus 1 = 1$, $u_{19} = 1$ will break the conditions, then the bias is reduced significantly.
   Finally, we can recover 29 bits of key information in total. The whole process is given in Algorithm 4. The complexities occur in the statistical tests in the Step 1, 3, 4. In Step 1, we need to encrypt $64 \times 2^7 \times 2^{5.15} \approx 2^{18.15}$ samples. In Step 3, we need to test fewer assignments of V_l that passed the filters of Step 2. The complexity of this step is significantly smaller. In Step 4, we need to encrypt $18 \times 2^{8.96} \approx 2^{13.13}$ samples. Since each sample contains 4 nonces, the whole complexity is about $2^{18.15+2} = 2^{20.15}$. We have implemented this attack, for 1000 random keys, we can recover the correct 29 bits of key in more than 950 experiments, which meets our expectations.
5. Given 4 opportunities satisfying $k_i = k_{i+64} = k_{i+9} = k_{i+73} = 0$, we expect to recover $29 \times 4 = 116$ bits of key information at most. However, the probability that we have 4 opportunities is only 54%, and it is possible that some key expressions are related. So we need to use more HDL approximations to make sure we can recover all the keys. As we mentioned, when $\Delta(i, j) = \Delta(i', i' + 24), 0 < i' < 64$, $\texttt{Bias}(S^{(5)}[0][51]) = 0.313$ under 16 conditions. We can apply the similar process as $\Delta(i', i' + 9)$. Firstly, we select 12 conditions from the 16 including the 4 conditions of $k_i = k_{i+24} = k_{i+64} = k_{i+88} = 0$ that still keep the bias as 0.313. When not all the 12 conditions are satisfied, the bias is at most 0.236. When $k_i = k_{i+24} = k_{i+64} = k_{i+88} = 0$ is not true, the bias is close to zero. The 12 conditions are listed as follows,

   $k_0 = 0$

Algorithm 4 Recover 29 bits of key information for 5-round Ascon

Input: Ascon initialization oracle
Output: 29 bits of key information

Assign 0 to all variables in V_r
for $0 \leq i < 63$ **do**
  Allocate $C = \varnothing$
  for Each of the 2^7 assignments of V_l, denoted by v **do**
    Encrypt $2^{5.15}$ samples with the input difference $\Delta(i, i+9)$
    if There is more than 28 times that the output difference of $S^{(5)}[0][27]$ is zero **then**
      $C \leftarrow v_r$
    end if
  end for
  if C is not $\varnothing$ **then**
    Test each of assignment in C with $2^{8.96}$ samples, find the one that makes $S^{(5)}[0][27]$ is zero for more than 423 times. Denote this assignment as $v^\star$
    for Each variable labeled by red in V_r **do**
      Flip this variable from 0 to 1, see if the bias of $S^{(5)}[0][27]$ is still $2^{8.96}$ with $v^\star$. Derive the corresponding key expressions
    end for
  end if
end for
return 29 key expressions derived

$$
\begin{aligned}
k_{64} &= 0 \\
k_{24} &= 0 \\
k_{88} &= 0 \\
u_{77} &= u_{13}k_{13} \oplus u_{13}k_{77} \oplus u_{13} \oplus u_{52}k_{52} \oplus u_{52}k_{116} \oplus u_{52} \oplus u_{55}k_{55} \oplus u_{55}k_{119} \\
&\quad \oplus u_{55} \oplus u_{116} \oplus u_{119} \oplus k_{13}k_{77} \oplus k_{13} \oplus k_{52}k_{116} \oplus k_{52} \oplus k_{55}k_{119} \oplus k_{55} \\
&\quad \oplus k_{77} \oplus k_{116} \oplus k_{119} \oplus 1; \\
u_{108} &= u_{19}k_{19} \oplus u_{19}k_{83} \oplus u_{19} \oplus u_{22}k_{22} \oplus u_{22}k_{86} \oplus u_{22} \oplus u_{44}k_{44} \oplus u_{44}k_{108} \\
&\quad \oplus u_{44} \oplus u_{83} \oplus u_{86} \oplus k_{19}k_{83} \oplus k_{19} \oplus k_{22}k_{86} \oplus k_{22} \oplus k_{44}k_{108} \oplus k_{44} \\
&\quad \oplus k_{83} \oplus k_{86} \oplus k_{108}; \\
u_{118} &= u_{23} \oplus u_{47} \oplus u_{54} \oplus u_{57} \oplus u_{87}k_{23} \oplus u_{87} \oplus u_{111} \oplus u_{121}k_{57} \oplus u_{121} \oplus k_{23} \\
&\quad \oplus k_{47} \oplus k_{54} \oplus k_{57} \oplus k_{111} \oplus k_{118}; \\
u_{117} &= u_{28}k_{28} \oplus u_{28}k_{92} \oplus u_{28} \oplus u_{31}k_{31} \oplus u_{31}k_{95} \oplus u_{31} \oplus u_{53}k_{53} \oplus u_{53}k_{117} \\
&\quad \oplus u_{53} \oplus u_{92} \oplus u_{95} \oplus k_{28}k_{92} \oplus k_{28} \oplus k_{31}k_{95} \oplus k_{31} \oplus k_{53}k_{117} \oplus k_{53} \\
&\quad \oplus k_{92} \oplus k_{95} \oplus k_{117}; \\
u_{89} &= u_3k_3 \oplus u_3k_{67} \oplus u_3 \oplus u_{25}k_{25} \oplus u_{25}k_{89} \oplus u_{25} \oplus u_{67} \oplus k_3k_{67} \oplus k_3 \\
&\quad \oplus k_{25}k_{89} \oplus k_{25} \oplus k_{67} \oplus k_{89} \oplus 1; \\
u_{78} &= u_7 \oplus u_{14} \oplus u_{17} \oplus u_{47} \oplus u_{71} \oplus u_{81}k_{17} \oplus u_{81} \oplus u_{111}k_{47} \oplus u_{111} \oplus k_7
\end{aligned}
$$

$$\oplus\, k_{14} \oplus k_{17} \oplus k_{47} \oplus k_{71} \oplus k_{78} \oplus 1;$$
$$u_{82} = u_{11} \oplus u_{18} \oplus u_{21} \oplus u_{51} \oplus u_{75} \oplus u_{85}k_{21} \oplus u_{85} \oplus u_{92}k_{28} \oplus u_{115}k_{51} \oplus u_{115} \oplus k_{11} \oplus k_{18} \oplus k_{51} \oplus k_{75} \oplus k_{82} \oplus k_{92} \oplus 1;$$
$$u_{99} = u_{11} \oplus u_{35} \oplus u_{42} \oplus u_{45} \oplus u_{75}k_{11} \oplus u_{75} \oplus u_{106} \oplus u_{109}k_{45} \oplus u_{109} \oplus u_{116}k_{52} \oplus k_{11} \oplus k_{35} \oplus k_{42} \oplus k_{45} \oplus k_{99} \oplus k_{106} \oplus k_{116} \oplus 1;$$

Thus, in Step 1, to detect the possible positions satisfying $k_i = k_{i+24} = k_{i+64} = k_{i+88} = 0$ requires the statistical test with $2^{5.90}$ samples. So the complexity of Step 1 is about $64 \times 2^{5.90} \times 2^8 = 2^{19.90}$ samples. The complexity of Step 3 is significantly smaller. In Step 4, since there are 21 nonce variables that are multiplied with keys, the complexity is about $21 \times 2^{8.30} \approx 2^{12.69}$ samples. Thus, the whole complexity is about $2^{19.90}$ samples, which is $2^{21.90}$ nonces. we can obtain in total 33 bits of key information. Together with the case of $\Delta(0, 9)$, we can expect to obtain enough key expressions, then all keys can be recovered. The whole complexity is estimated as about $2^{22.28}$.

F Conditions used in Conditional HDL Approximation on 6-Round Ascon

$$k_{64} = 0$$
$$k_{94} = 0$$
$$k_0 = 0$$
$$k_{30} = 0$$
$$k_{125} = 0$$
$$k_{61} = 0$$
$$u_{86} = u_{22}k_{22} \oplus u_{22}k_{86} \oplus u_{22} \oplus k_{22}k_{86} \oplus k_{22} \oplus k_{86} \oplus 1$$
$$u_{80} = u_{16}k_{16} \oplus u_{16}k_{80} \oplus u_{16} \oplus u_{19}k_{19} \oplus u_{19}k_{83} \oplus u_{19} \oplus u_{41}k_{41} \oplus u_{41}k_{105} \oplus u_{41} \oplus u_{83} \oplus u_{105} \oplus k_{16}k_{80} \oplus k_{16} \oplus k_{19}k_{83} \oplus k_{19} \oplus k_{41}k_{105} \oplus k_{41} \oplus k_{80} \oplus k_{83} \oplus k_{105} \oplus 1$$
$$u_{122} = u_{19}k_{19} \oplus u_{19}k_{83} \oplus u_{19} \oplus u_{58}k_{58} \oplus u_{58}k_{122} \oplus u_{83} \oplus k_{19}k_{83} \oplus k_{19} \oplus k_{58}k_{122} \oplus k_{83} \oplus k_{122} \oplus 1$$
$$u_{108} = u_{19}k_{19} \oplus u_{19}k_{83} \oplus u_{19} \oplus u_{22}k_{22} \oplus u_{22}k_{86} \oplus u_{22} \oplus u_{44}k_{44} \oplus u_{44}k_{108} \oplus u_{44} \oplus u_{83} \oplus u_{86} \oplus k_{19}k_{83} \oplus k_{19} \oplus k_{22}k_{86} \oplus k_{22} \oplus k_{44}k_{108} \oplus k_{44} \oplus k_{83} \oplus k_{86} \oplus k_{108} \oplus 1$$
$$u_{47} = u_{23} \oplus u_{54} \oplus u_{57} \oplus u_{87}k_{23} \oplus u_{87} \oplus u_{111} \oplus u_{118} \oplus u_{121}k_{57} \oplus u_{121} \oplus k_{23} \oplus k_{47} \oplus k_{54} \oplus k_{57} \oplus k_{111} \oplus k_{118}$$
$$u_{17} = u_{41} \oplus u_{48} \oplus u_{51} \oplus u_{81}k_{17} \oplus u_{81} \oplus u_{105} \oplus u_{112} \oplus u_{115}k_{51} \oplus u_{115} \oplus u_{122}k_{58} \oplus k_{17} \oplus k_{41} \oplus k_{48} \oplus k_{51} \oplus k_{105} \oplus k_{112} \oplus k_{122}$$
$$u_2 = u_{12} \oplus u_{42} \oplus u_{66} \oplus u_{76}k_{12} \oplus u_{76} \oplus u_{83}k_{19} \oplus u_{106}k_{42} \oplus u_{106} \oplus k_2 \oplus k_9$$

$$
\begin{aligned}
&\oplus k_{12} \oplus k_{42} \oplus k_{66} \oplus k_{73} \oplus k_{83} \\
u_{97} &= u_{33}k_{33} \oplus u_{33}k_{97} \oplus u_{33} \oplus u_{55}k_{55} \oplus u_{55}k_{119} \oplus u_{55} \oplus u_{119} \oplus k_{33}k_{97} \oplus k_{33} \\
&\oplus k_{55}k_{119} \oplus k_{55} \oplus k_{97} \oplus k_{119} \oplus 1 \\
u_{32} &= u_8 \oplus u_{39} \oplus u_{42} \oplus u_{72}k_8 \oplus u_{72} \oplus u_{96} \oplus u_{103} \oplus u_{106}k_{42} \oplus u_{106} \oplus u_{113}k_{49} \\
&\oplus k_8 \oplus k_{32} \oplus k_{39} \oplus k_{42} \oplus k_{96} \oplus k_{103} \oplus k_{113} \oplus 1 \\
u_{20} &= u_{44} \oplus u_{51} \oplus u_{54} \oplus u_{84}k_{20} \oplus u_{84} \oplus u_{108} \oplus u_{115} \oplus u_{118}k_{54} \oplus u_{118} \oplus k_{44} \\
&\oplus k_{51} \oplus k_{54} \oplus k_{108} \oplus k_{115} \oplus 1 \\
u_{15} &= u_8 \oplus u_{18} \oplus u_{48} \oplus u_{72} \oplus u_{79} \oplus u_{82}k_{18} \oplus u_{82} \oplus u_{89}k_{25} \oplus u_{112}k_{48} \oplus u_{112} \\
&\oplus k_8 \oplus k_{15} \oplus k_{18} \oplus k_{48} \oplus k_{72} \oplus k_{79} \oplus k_{89} \oplus 1 \\
u_{74} &= u_{10}k_{10} \oplus u_{10}k_{74} \oplus u_{10} \oplus u_{49}k_{49} \oplus u_{49}k_{113} \oplus u_{49} \oplus u_{52}k_{52} \oplus u_{52}k_{116} \oplus u_{52} \\
&\oplus u_{113} \oplus u_{116} \oplus k_{10}k_{74} \oplus k_{10} \oplus k_{49}k_{113} \oplus k_{49} \oplus k_{52}k_{116} \oplus k_{52} \oplus k_{74} \\
&\oplus k_{113} \oplus k_{116} \\
u_{67} &= u_3k_3 \oplus u_3k_{67} \oplus u_3 \oplus u_{25}k_{25} \oplus u_{25}k_{89} \oplus u_{25} \oplus u_{89} \oplus k_3k_{67} \oplus k_3 \oplus k_{25}k_{89} \\
&\oplus k_{25} \oplus k_{67} \oplus k_{89} \oplus 1 \\
u_{11} &= u_{18} \oplus u_{21} \oplus u_{51} \oplus u_{75} \oplus u_{82} \oplus u_{85}k_{21} \oplus u_{85} \oplus u_{92}k_{28} \oplus u_{115}k_{51} \oplus u_{115} \\
&\oplus k_{11} \oplus k_{18} \oplus k_{51} \oplus k_{75} \oplus k_{82} \oplus k_{92} \oplus 1 \\
u_{114} &= u_{25}k_{25} \oplus u_{25}k_{89} \oplus u_{25} \oplus u_{28}k_{28} \oplus u_{28}k_{92} \oplus u_{28} \oplus u_{50}k_{50} \oplus u_{50}k_{114} \oplus u_{50} \\
&\oplus u_{89} \oplus u_{92} \oplus k_{25}k_{89} \oplus k_{25} \oplus k_{28}k_{92} \oplus k_{28} \oplus k_{50}k_{114} \oplus k_{50} \oplus k_{89} \oplus k_{92} \\
&\oplus k_{114} \\
u_{95} &= u_{28}k_{28} \oplus u_{28}k_{92} \oplus u_{28} \oplus u_{31}k_{31} \oplus u_{31}k_{95} \oplus u_{31} \oplus u_{53}k_{53} \oplus u_{53}k_{117} \oplus u_{53} \\
&\oplus u_{92} \oplus u_{117} \oplus k_{28}k_{92} \oplus k_{28} \oplus k_{31}k_{95} \oplus k_{31} \oplus k_{53}k_{117} \oplus k_{53} \oplus k_{92} \oplus k_{95} \\
&\oplus k_{117} \\
u_6 &= u_9 \oplus u_{39} \oplus u_{63} \oplus u_{70} \oplus u_{73}k_9 \oplus u_{73} \oplus u_{80}k_{16} \oplus u_{103}k_{39} \oplus u_{103} \oplus u_{127} \\
&\oplus k_6 \oplus k_{39} \oplus k_{63} \oplus k_{70} \oplus k_{80} \oplus k_{127} \oplus 1 \\
u_{13} &= u_{23} \oplus u_{53} \oplus u_{77} \oplus u_{87}k_{23} \oplus u_{87} \oplus u_{117}k_{53} \oplus u_{117} \oplus k_{13} \oplus k_{20} \oplus k_{23} \oplus k_{53} \\
&\oplus k_{77} \oplus k_{84} \oplus 1
\end{aligned}
$$

G Conditions used in Condition HDL Approximation on 5-Round Xoodyak

$$
\begin{aligned}
u_{93} &= k_{79} \oplus k_{93} \oplus u_{79} \oplus u_{107} \oplus u_{116} \oplus u_{207} \oplus u_{221} \\
u_{70} &= k_{93} \oplus k_{70} \oplus u_{93} \oplus u_{107} \oplus u_{198} \oplus u_{221} \oplus 1 \\
u_7 &= k_{16} \oplus k_7 \oplus u_{16} \oplus u_{135} \oplus u_{144} \oplus u_{181} \\
u_4 &= k_{27} \oplus k_{41} \oplus k_4 \oplus u_{27} \oplus u_{132} \oplus u_{155} \oplus 1 \\
u_{54} &= k_{63} \oplus k_{68} \oplus k_{54} \oplus u_{63} \oplus u_{182} \oplus u_{191} \oplus 1 \\
u_{97} &= k_{106} \oplus u_{15} \oplus k_{97} \oplus u_{106} \oplus u_{225} \oplus u_{234} \oplus 1
\end{aligned}
$$

$$
\begin{aligned}
u_{43} &= k_{52} \oplus k_{43} \oplus u_{52} \oplus u_{171} \oplus u_{180} \oplus u_{217} \\
u_{25} &= k_{25} \oplus k_{16} \oplus u_{16} \oplus u_{144} \oplus u_{153} \oplus u_{190} \\
u_{111} &= k_{20} \oplus k_{102} \oplus k_{111} \oplus u_{102} \oplus u_{230} \oplus u_{239} \\
u_{18} &= k_{18} \oplus k_{27} \oplus k_{32} \oplus u_{27} \oplus u_{146} \oplus u_{155} \oplus 1 \\
u_{125} &= k_{11} \oplus k_{102} \oplus k_{125} \oplus u_{102} \oplus u_{230} \oplus u_{253} \\
u_{1} &= k_{1} \oplus k_{10} \oplus k_{47} \oplus u_{10} \oplus u_{129} \oplus u_{138}
\end{aligned}
$$

H Updating Degree Matrix Transition of the Ascon Permutation

Proposition 3 (Degree Matrix Transition over p_S). *With the knowledge of* $\mathrm{DM}(S) = (d_{i,j}, 0 \le i < 5, 0 \le j < 64)$*, we have* $\mathrm{DM}(p_S(S)) = (d'_{i,j}, 0 \le i < 5, 0 \le j < 64)$*, where* $d'_{i,j}, 0 \le i < 5, 0 \le j < 64$ *are computed as*

$$\begin{aligned}
d'_{0,j} &= \max(d_{4,j} + d_{1,j}, d_{3,j}, d_{2,j} + d_{1,j}, d_{2,j}, d_{0,j} + d_{1,j}, d_{1,j}, d_{0,j}) \\
d'_{1,j} &= \max(d_{4,j}, d_{3,j} + d_{2,j}, d_{3,j} + d_{1,j}, d_{3,j}, d_{2,j} + d_{1,j}, d_{2,j}, d_{1,j}, d_{0,j}) \\
d'_{2,j} &= \max(d_{4,j} + d_{3,j}, d_{4,j}, d_{2,j}, d_{1,j}, 0) \qquad\qquad , 0 \le j < 64 \\
d'_{3,j} &= \max(d_{4,j} + d_{0,j}, d_{4,j}, d_{3,j} + d_{0,j}, d_{3,j}, d_{2,j}, d_{1,j}, d_{0,j}) \\
d'_{4,j} &= \max(d_{4,j} + d_{1,j}, d_{4,j}, d_{3,j}, d_{1,j} + d_{0,j}, d_{1,j})
\end{aligned}$$

Proposition 4 (Degree Matrix Transition over p_L). *With the knowledge of* $\mathrm{DM}(S) = (d'_{i,j}, 0 \le i < 5, 0 \le j < 64)$*, we have* $\mathrm{DM}(p_L(S)) = (d''_{i,j}, 0 \le i < 5, 0 \le j < 64)$*, where* $d''_{i,j}, 0 \le i < 5, 0 \le j < 64$ *are computed as*

$$\begin{aligned}
d''_{0,j} &= \max(d'_{0,j+0}, d'_{0,j-19 \bmod 64}, d'_{0,j-28 \bmod 64}) \\
d''_{1,j} &= \max(d'_{1,j+0}, d'_{1,j-61 \bmod 64}, d'_{1,j-39 \bmod 64}) \\
d''_{2,j} &= \max(d'_{2,j+0}, d'_{2,j-\ 1 \bmod 64}, d'_{2,j-\ 6 \bmod 64}), 0 \le j < 64 \\
d''_{3,j} &= \max(d'_{3,j+0}, d'_{3,j-10 \bmod 64}, d'_{3,j-17 \bmod 64}) \\
d''_{4,j} &= \max(d'_{4,j+0}, d'_{4,j-\ 7 \bmod 64}, d'_{4,j-41 \bmod 64})
\end{aligned}$$

Proof (for Propositions 3 and 4). It is clear that if $y = x_0 \oplus x_1$, $\deg(y) \le \max(\deg(x_0), \deg(x_1))$; if $y = x_0 x_1$, $\deg(y) \le \deg(x_0) + \deg(x_1)$. Then from the ANFs of p_S (Equation 11) and p_L (Equation 13), we directly derive the formulas in Proposition 3 and Proposition 4.

I Zero-Sum Distinguishers for Full Ascon Permutation

The *zero-sum distinguisher* was first proposed to study the non-ideal property of the Keccak-f permutation [AM09,BC10,YLW+19], which was also used to distinguish the (12-round) Ascon permutation by its designers [DEMS15]. It studies the following question. Given a permutation $P : \mathbb{F}_2^n \to \mathbb{F}_2^n$, can we create a set of inputs, I, such that $\bigoplus_{x \in I} x = \bigoplus_{x \in I} P(x) = 0$? Currently, the best result of the zero-sum distinguisher for the 12-round Ascon permutation costs 2^{130} calls [DEMS21]. In this subsection, we show how to use our HD distinguisher to build a zero-sum distinguisher for a 12-round Ascon permutation with only 2^{55} calls.

Note that the idea of the degree matrix transition method introduced in Section 7 is also applicable to the inverse operations of the Ascon permutation. We deduce the transitional rules for the inverse operations of the Ascon permutation. Thereby we can give two corollaries of Propositions 3 and 4. According to the ANFs of the inverse Sbox of Ascon in Equation 12, we deduce the following corollary,

Algorithm 5 Detect HD Distinguishers (up to 64^{th}-order) for the ASCON permutation

Input: r-round ASCON permutation, $r \geq 4$
Output: $(\bar{X}, \bar{\Delta})$ leading to HD distinguishers (up to 64^{th}-order) for r-round ASCON permutation, the order of the HD

```
degree = 64
for X̄ from 0 to 31 do
    for Δ̄ from 1 to 31 do
        for i from 0 to 63 do
            for j from 0 to 4 do
                S^0[j][i] = X[j] ⊕ x_i Δ[j]
            end for
        end for
        Compute the exact ANF of S^2.5 and compute DM(S^2.5)
        Compute the degree matrix of S^r from S^2.5 using Propositions 3 and 4
        if min(DM(S^r)) < degree then
            degree = min(DM(S^r))                ▷ To find the best distinguisher
        end if
    end for
end for
return (X̄, Δ̄, DM(S^r))
```

Corollary 1 (Degree Matrix Transition over p_S^{-1}). *With the knowledge of* $\mathrm{DM}(S) = (d_{i,j}, 0 \leq i < 5, 0 \leq j < 64)$*, we have* $\mathrm{DM}(p_S^{-1}(S)) = (d'_{i,j}, 0 \leq i < 5, 0 \leq j < 64)$*, where* $d'_{i,j}, 0 \leq i < 5, 0 \leq j < 64$ *are computed as*

$$
\begin{aligned}
d'_{0,j} &= \max(d_{4,j} + d_{3,j} + d_{2,j}, d_{4,j} + d_{3,j} + d_{1,j}, d_{4,j} + d_{3,j} + d_{0,j}, d_{3,j} + d_{2,j} + d_{0,j},\\
&\quad d_{3,j} + d_{2,j}, d_{3,j}, d_{2,j}, d_{1,j} + d_{0,j}, d_{1,j}, 0)\\
d'_{1,j} &= \max(d_{4,j} + d_{2,j} + d_{0,j}, d_{4,j}, d_{3,j} + d_{2,j}, d_{2,j} + d_{0,j}, d_{1,j}, d_{0,j})\\
d'_{2,j} &= \max(d_{4,j} + d_{3,j} + d_{1,j}, d_{4,j} + d_{3,j}, d_{4,j} + d_{2,j} + d_{1,j}, d_{4,j} + d_{2,j} + d_{0,j},\\
&\quad d_{4,j} + d_{2,j}, d_{4,j}, d_{3,j} + d_{2,j}, d_{3,j} + d_{1,j} + d_{0,j}, d_{3,j} + d_{1,j}, d_{2,j} + d_{1,j} + d_{0,j},\\
&\quad d_{2,j} + d_{1,j}, d_{2,j} + d_{0,j}, d_{1,j}, d_{0,j}, 0)\\
d'_{3,j} &= \max(d_{4,j} + d_{2,j} + d_{1,j}, d_{4,j} + d_{2,j} + d_{0,j}, d_{4,j} + d_{2,j}, d_{4,j} + d_{1,j}, d_{4,j}, d_{3,j},\\
&\quad d_{2,j} + d_{1,j}, d_{2,j} + d_{0,j}, d_{1,j})\\
d'_{4,j} &= \max(d_{4,j} + d_{3,j} + d_{2,j}, d_{4,j} + d_{2,j} + d_{1,j}, d_{4,j} + d_{2,j} + d_{0,j}, d_{4,j} + d_{2,j},\\
&\quad d_{3,j} + d_{2,j} + d_{0,j}, d_{3,j} + d_{2,j}, d_{3,j}, d_{2,j} + d_{1,j}, d_{2,j} + d_{0,j}, d_{1,j} + d_{0,j})
\end{aligned}
$$

The ANF of p_L^{-1} is a little complicated, so we introduce a simpler version of the degree matrix transition for p_L^{-1}.

Corollary 2 (Simplified Degree Matrix Transition over p_L^{-1}). *With the knowledge of* $\mathrm{DM}(S) = (d'_{i,j}, 0 \leq i < 5, 0 \leq j < 64)$*, we have* $\mathrm{DM}(p_L^{-1}(S)) =$

$(d''_{i,j}, 0 \le i < 5, 0 \le j < 64)$, *where* $d''_{i,j}, 0 \le i < 5, 0 \le j < 64$ *are computed as*

$$\begin{aligned}
d''_{0,j} &= \max_{0 \le k < 64} (d'_{0,k}) \\
d''_{1,j} &= \max_{0 \le k < 64} (d'_{1,k}) \\
d''_{2,j} &= \max_{0 \le k < 64} (d'_{2,k}), 0 \le j < 64 \\
d''_{3,j} &= \max_{0 \le k < 64} (d'_{3,k}) \\
d''_{4,j} &= \max_{0 \le k < 64} (d'_{4,k})
\end{aligned}$$

It is easy to verify that Corollary 1 and 2 give an upper bound on the degree of output bits of p_S^{-1} and p_L^{-1}. Thus, we can replay the calculation of Section 7 to the inverse Ascon permutation.

Considering that with $(X, \boldsymbol{\Delta})$ in Equation 10, the upper bounds on the degree of the five words of the output after 8 rounds are (47, 47, 45, 47, 47), we only test the $(X, \boldsymbol{\Delta})$ for the 4-round inverse Ascon permutation. Note that in the forward direction, we did not include the first p_C, thus we add it to the backward calculation. In other words, the four rounds of inverse Ascon permutation is

$$P_b = (p_C \circ p_S^{-1} \circ p_L^{-1})^4 \circ p_C.$$

We first calculate the exact ANFs of the output of $p_L^{-1} \circ p_C \circ p_S^{-1} \circ p_L^{-1} \circ p_C(X \oplus \boldsymbol{x}\boldsymbol{\Delta})$, then apply Corollary 1 and 2 to calculate the degree upper bounds for 4 rounds of inverse Ascon permutation, *i.e.*, P_b. Finally, with $(X, \boldsymbol{\Delta}) \in \{(\texttt{0xf}, \texttt{0x18}), (\texttt{0x17}, \texttt{0x18})\}$, the degree upper bounds are shown in Table 8.

Thus, we choose 55 positions from $\{0, 1, \ldots, 63\}$, traverse the variables, and keep the remaining $64 - 55 = 9$ positions as constants for the state after p_C of the fifth round of the 12-round Ascon permutation. The corresponding plaintext and ciphertext sets are zero-sum. Thus we obtain a zero-sum distinguisher for 12-round Ascon permutation, with complexity of 2^{55}. Similarly, with 7 forward rounds and 3 backward rounds, we can construct a zero-sum distinguisher for 10 rounds with 2^{25} complexity; with 8 forward rounds and 3 backward rounds, we can construct a zero-sum distinguisher for 11 rounds with 2^{48} complexity. We experimentally verified the 7-round zero-sum distinguisher.

Here we only give the results, which have been abstracted into Table 8[9].

According to Tables 4 and 8, we choose 55 positions from $\{0, 1, \ldots, 63\}$, traverse the variables, and keep the remaining $64 - 55 = 9$ positions as constants for the state after p_C of the fifth round of the 12-round Ascon permutation. The corresponding plaintext and ciphertext sets are zero-sum. Thus we obtain a zero-sum distinguisher for a 12-round Ascon permutation, with a complexity of 2^{55}. Similarly, with 8 forward rounds and 3 backward rounds, we can construct a zero-sum distinguisher for 11 rounds with 2^{48} complexity; with 7 forward rounds and 3 backward rounds, we can construct a zero-sum distinguisher for 10

[9] Recalling Section 7, we ignore the first p_C for the forward direction. Here we include this p_C in the backward direction.

Table 8: Upper bounds on the algebraic degree of the DSF of the inverse Ascon permutation with $(X, \boldsymbol{\Delta}) \in \{(\texttt{0xf}, \texttt{0x18}), (\texttt{0x17}, \texttt{0x18})\}$. We experimentally verified the upper bounds on degrees up to 3 inverse rounds.

Round r	**Upper bounds on the algebraic degree**				
	$S[0]$	$S[1]$	$S[2]$	$S[3]$	$S[4]$
1	2	1	2	0	2
2	4	6	6	6	6
3	18	16	18	18	18
4	54	54	54	54	54

rounds with 2^{25} complexity; We experimentally verified the 10-round zero-sum distinguisher.

The impact of the zero-sum distinguisher. In [DEMS15], the designers gave a zero-sum distinguisher for 12 rounds with complexity 2^{130} and noted: *"The non-ideal properties of the permutation do not seem to affect the security of* Ascon*. In particular, the complexity of* 2^{130} *is above the cipher's claimed security level."* Yet, we show that our 12-round distinguisher requires a much lower complexity than the cipher's claimed security level (2^{128}).

However, although these zero-sum distinguishers require low complexities, their actual impact on the security of the Ascon AEAD and Hash are very likely non-existent or at best not clear. As discussed in [WGR18,GPT21,Kec], the advantage of the zero-sum distinguisher for Ascon permutation and a perfect permutation is very small, usually falling under a factor of 2 (our zero-sum approach follows the same philosophy).

Yet, zero-sum distinguishers still represent some non-ideal property of the target permutation. We can mention that the Keccak team decided to increase the number of rounds of Keccak-f (*e.g.*, for Keccak-f[1600] from 18 to 24 rounds) in round 2 of the SHA-3 competition, even though they judged as very unlikely that the zero-sum distinguishers on the full Keccak-f permutation can result in actual attacks against the global Keccak scheme. It is also worth mentioning that in [BDP+18], the Keccak team presented the fast hash scheme KangarooTwelve that is based on the 12-round Keccak-f[1600].

Comparison with the zero-sum distinguisher in [DEMS15]. The zero-sum distinguisher in [DEMS15] that needs 2^{130} calls can be further adapted into a zero-sum partition distinguisher of the full Ascon permutation. That is, by enumerating the constant bits in the middle, we can divide the whole input-output space into $2^{320-130}$ subspaces, and the plaintexts/ciphertexts in each subspace present a zero-sum distinguisher. Differently, our zero-sum distinguisher chose a specific initial structure of size 2^{64} in the middle, which means it cannot be adapted into a zero-sum partition distinguisher. The strength of a zero-sum partition distinguisher is that it has a larger advantage from the generic

attack when compared to simple zero-sum distinguishers. However, a noticeable disadvantage is that building or verifying such a zero-sum partition requires an extremely huge computational cost (2^n calls for an n-bit cryptographic primitive) which is actually impossible to perform [GPT21]. Besides, the cost of the best zero-sum partition generic algorithm remains only conjectured. We believe that our distinguishers provide some new insights into the structural properties of the Ascon permutation.

Relationship with the previous structural algebraic distinguishers. In this section, we provided a systematic method to construct an algebraic distinguisher based on analyzing the DSF. Since the DSF is parameterized by the input values and differences, a proper choice of $(X, \boldsymbol{\Delta})$ can reduce the degree of the output bits. Before our work, some similar ideas were proposed to analyze the permutation-based primitives. Usually, by analyzing the algebraic properties of the round functions, an "initial structure" is set as the input values of the target permutation. With this initial structure, the algebraic degree would increase more slowly since some intermediate variables will become linear or quadratic. An example of these attacks is the conditional cube attack [LDW17,CHK22,CKT+22,BCP22], where the influences of the secret keys on the algebraic degrees are captured to perform key recovery attacks. Such a technique is also called the linearization technique in some papers such as [BLNS21]. In terms of the DSF, a good $(X, \boldsymbol{\Delta})$ will also reduce the degrees of intermediate variables (in our case, the degree of f^0 is reduced), thus our method searches for a good initial structure and implicitly uses the linearization technique to slow the degree increase. However, previous methods usually require careful manual analyses of the round functions, which are sometimes tedious, or even impossible for complicated cases. Instead, our method is universal and can consistently analyze their algebraic properties by the DSF. The DSF is an explicit formula of an output HD/HDL, *i.e.*, a Boolean function of both the input difference $(\boldsymbol{\Delta})$ and input value (X), which gives us a unified viewpoint for algebraic attacks on cryptographic primitives.

J Discussion of the Precision of HATF

In this section, we provide three kinds of figures to show the precision of the HATF:

1. In Section J.1, we show the figures of theoretical and experimental biases of the second-order HDL for 4-round Ascon initialization. The second-order differences denoted by $\Delta(0, i), 1 \leq i < 64$ are the captions of each figure.
2. In Section J.2, we show the figures of theoretical and experimental biases of the second-order HDL for 5-round Ascon initialization. The second-order differences denoted by $\Delta(0, i), 1 \leq i < 64$ are the captions of each figure.
3. In Section J.3, we show the figures of theoretical and experimental biases of ℓ^{th}-order ($3 \leq \ell \leq 8$) HDL for 5-round Ascon initialization. The ℓ^{th}-order differences denoted by $\Delta(i_0, i_1, \ldots, i_{\ell-1}), 0 \leq i_0 < i_1 < \cdots < i_{\ell-1} < 64$ are the captions of each figure.

The theoretical biases obtained in Sections J.1,J.2 and J.3 are obtained with the partition technique: When the input difference is $\Delta(i_0, i_1, \ldots, i_{\ell-1})$ for an ℓ^{th} HDL cryptanalysis, the 2ℓ bits of keys

$$S^{(0)}[1][i_0], S^{(0)}[2][i_0], \ldots, S^{(0)}[1][i_{\ell-1}], S^{(0)}[2][i_{\ell-1}] \tag{25}$$

are used to partition the whole input space into $2^{2\ell}$ subspaces. For the experiments, we use 2^{26} samples to obtain the data.

Next, based on these figures, we discuss the precision of HATF and the impact of the partitioning technique.

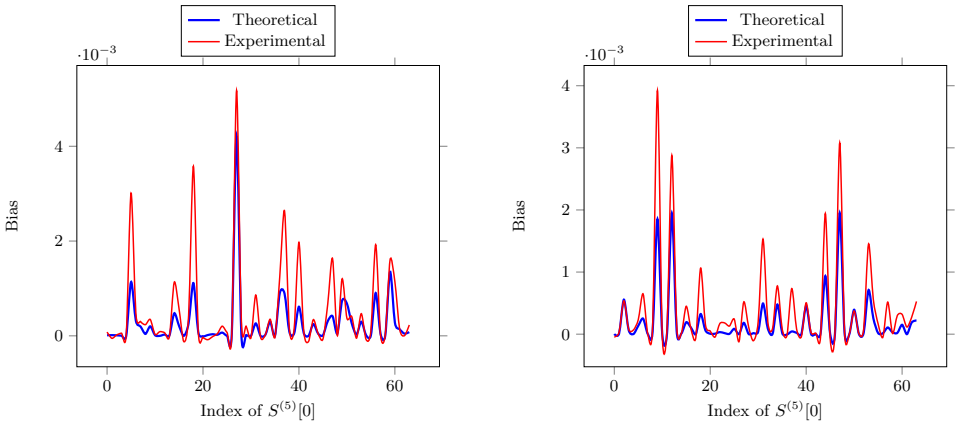
1. From the second-order HDL distinguishers for 4-round Ascon in Section J.1, we can find that the curves of theoretical and experimental results coincide almost exactly. That is to say, HATF predicts the second-order HDL biases almost perfectly for 4-round Ascon.
2. From the second-order HDL distinguishers for 5-round Ascon in Section J.2, we can see that the precision of HATF drops. Not all biased bits and their biases are predicted correctly. However, there are two important points:
   (a) The shapes of the curve of theoretical results are related to those of the experimental results: one peak of the theoretical curve is always related to one experimental peak, although the former ones might be weaker than the latter ones.
   (b) The absolute values of the biases in most cases are smaller than the experimental ones, except for the $\Delta(0, 26)$ (the leftmost peak) and $\Delta(0, 38)$ (the peak at index 40). Note that even for the two exceptions, HATF correctly predicts that the bits are significantly biased, only the biases are overestimated. Later, with an advanced partitioning technique, we can overcome the overestimations.

   Considering the two points, we are confident that HATF is very useful to find second-order distinguishers for the 5-round Ascon. If HATF predicts that a bit is biased, then there is a high probability that the bit is truly biased. And there is a high probability that the value of the bias is greater than the prediction.

3. From the ℓ^{th}-order HDL distinguishers for 5-round Ascon, $3 \leq \ell \leq 8$ in Section J.3, Firstly, we can also observe the similar two points with the second-order HDL distinguishers for 5-round Ascon. Secondly, as the number of orders increases, the real biases of output bits become higher. HATF performs better for larger biases than smaller ones.

Comparing the precision of the second-order HATF for 4-round and 5-round Ascon, we guess that the reduction of the precision of HATF is mainly caused by the more complicated HATF. More complicated HATF assumes more independent intermediate state bits and fail more easily.

Recall that all theoretical results in Sections J.1,J.2 and J.3 are obtained with partitioning the input subspaces based on the 2ℓ key bits in Equation 25. A better partition of the input space can simplify the HATF drastically in each subspace. For example, for the second-order HDL cryptanalysis on 5-round Ascon with the input difference $\Delta(0, 9)$, we can use the 10 Equations 14–24 to partition the input space, which is obtained by injecting conditions into the first two rounds of the HATF. The improved theoretical result with $\Delta(0, 9)$ for 5-round Ascon is given in Figure 4a. With the improved partition, it can be seen the precision increases significantly. Similarly, we can obtain 16 conditions into the first two rounds of the HATF concerning $\Delta(0, 26)$, which are used to partition the whole input space into 2^{16} subspaces. With the improved partition, the theoretical curve of $\Delta(0, 26)$ is shown in Figure 4b. It can be seen, the precision also increases significantly, and the leftmost peak is also predicted correctly. Thus, we can conclude that a better partition can improve the precision significantly.

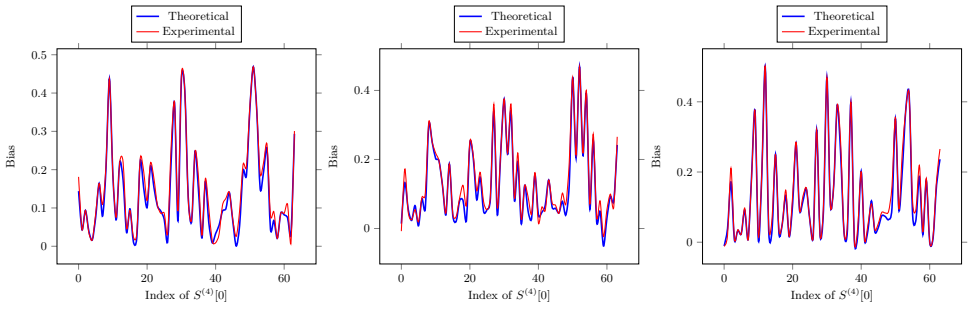


(a) Second-order HDL biases for 5-R Ascon with improved partition with $\Delta(0, 9)$

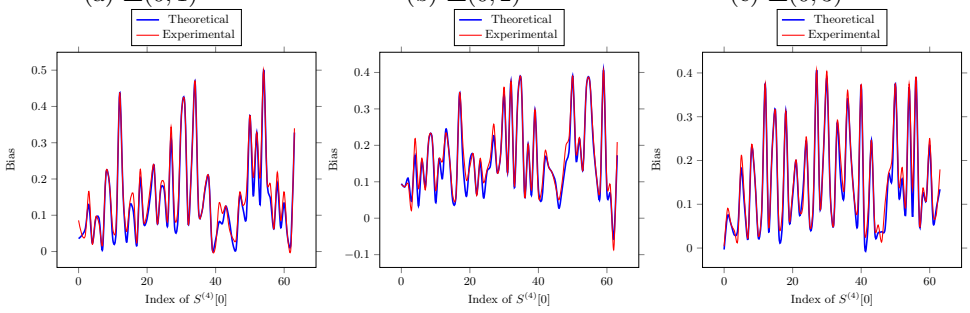
(b) Second-order HDL biases for 5-R Ascon with improved partition with $\Delta(0, 26)$

Fig. 4: The theoretical and experimental biases curves for 5-round Ascon with improved partition.

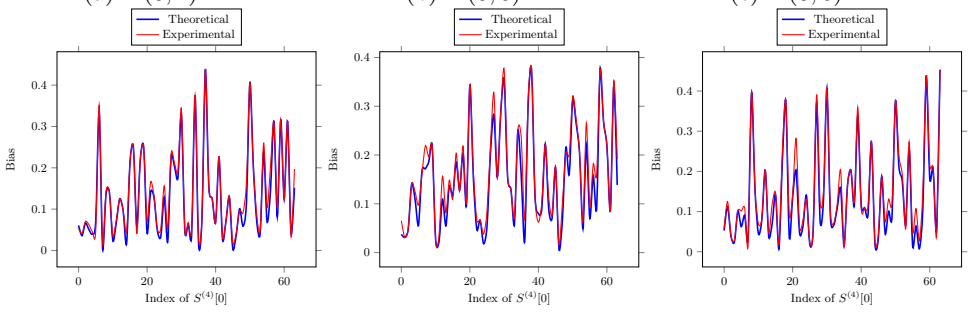
J.1 Figures of Second-Order HDL Biases for 4-Round Ascon



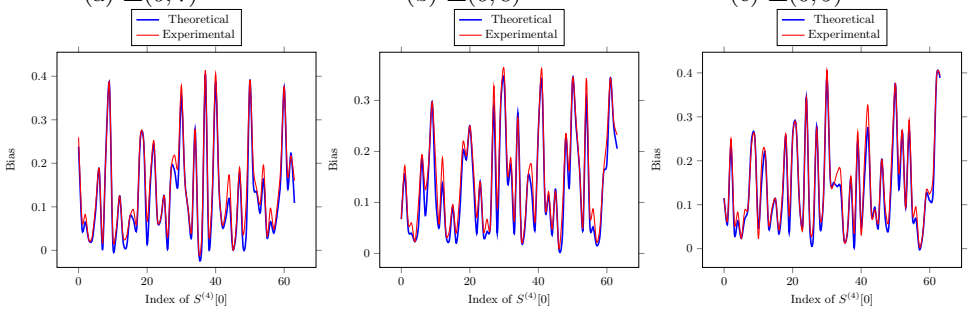
(a) $\Delta(0,1)$ (b) $\Delta(0,2)$ (c) $\Delta(0,3)$



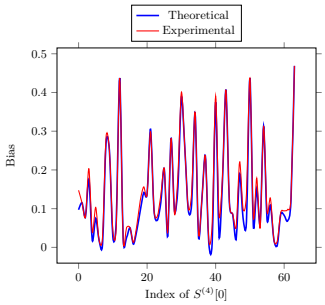
(a) $\Delta(0,4)$ (b) $\Delta(0,5)$ (c) $\Delta(0,6)$



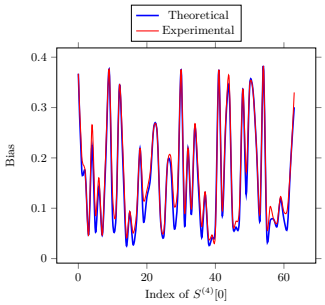
(a) $\Delta(0,7)$ (b) $\Delta(0,8)$ (c) $\Delta(0,9)$



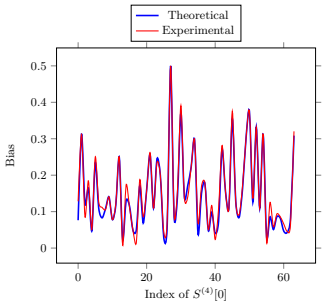
(a) $\Delta(0,10)$ (b) $\Delta(0,11)$ (c) $\Delta(0,12)$



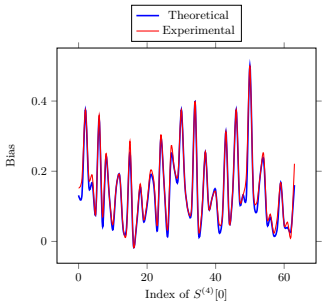
(a) $\Delta(0, 13)$



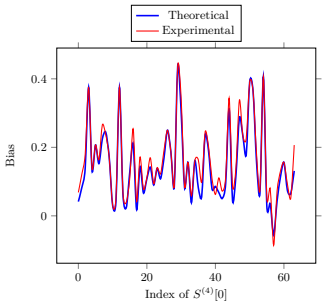
(b) $\Delta(0, 14)$



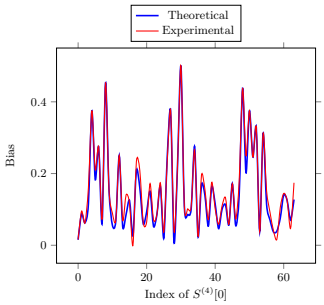
(c) $\Delta(0, 15)$



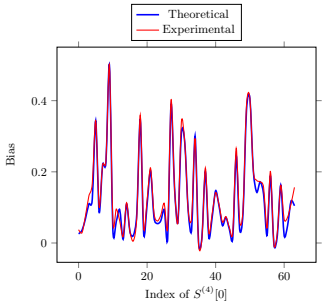
(a) $\Delta(0, 16)$



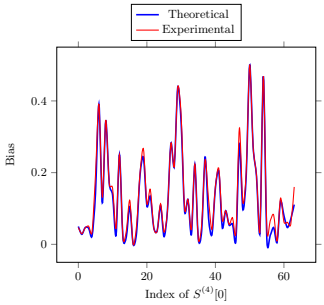
(b) $\Delta(0, 17)$



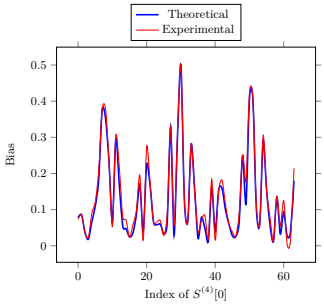
(c) $\Delta(0, 18)$



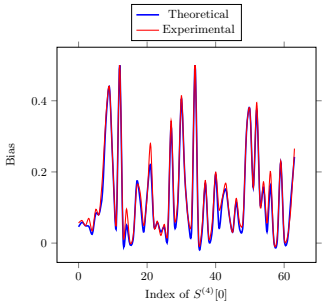
(a) $\Delta(0, 19)$



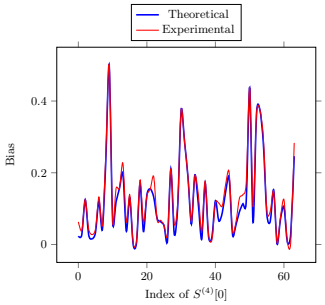
(b) $\Delta(0, 20)$



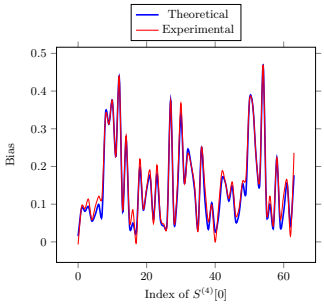
(c) $\Delta(0, 21)$



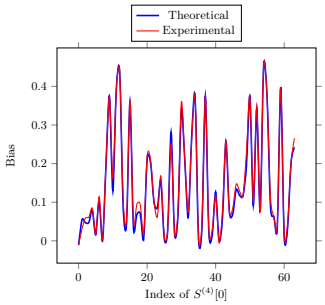
(a) $\Delta(0, 22)$



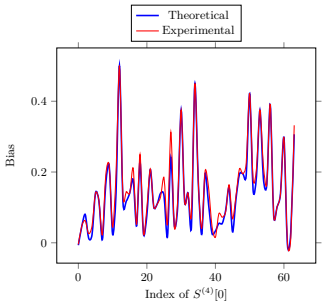
(b) $\Delta(0, 23)$



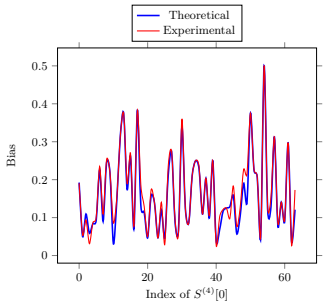
(c) $\Delta(0, 24)$



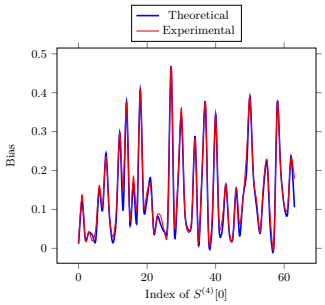
(a) $\Delta(0, 25)$



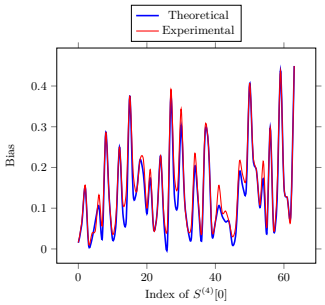
(b) $\Delta(0, 26)$



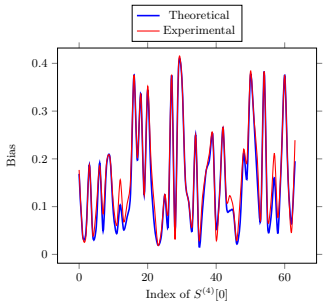
(c) $\Delta(0, 27)$



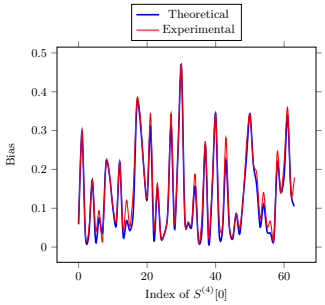
(a) $\Delta(0, 28)$



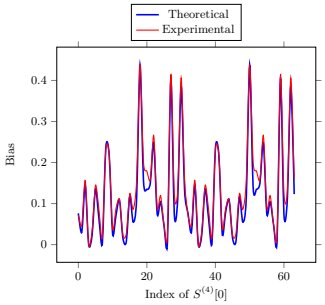
(b) $\Delta(0, 29)$



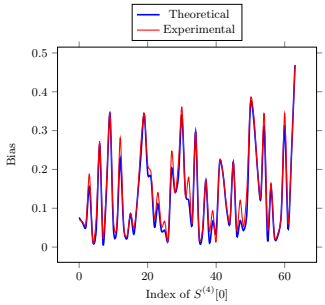
(c) $\Delta(0, 30)$



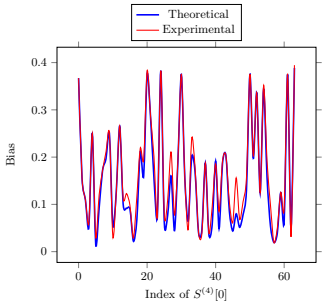
(a) $\Delta(0, 31)$



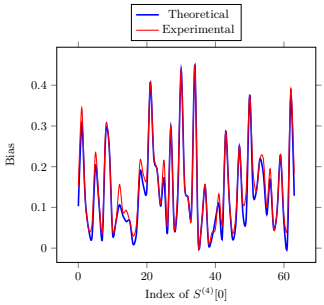
(b) $\Delta(0, 32)$



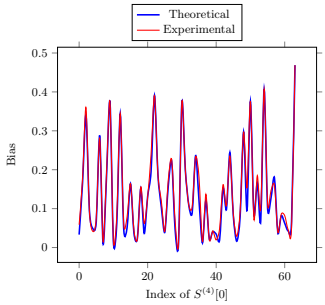
(c) $\Delta(0, 33)$



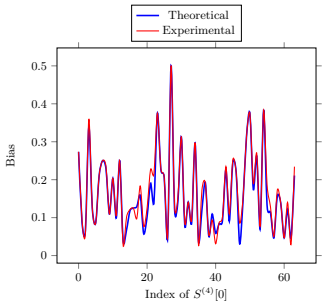
(a) $\Delta(0, 34)$



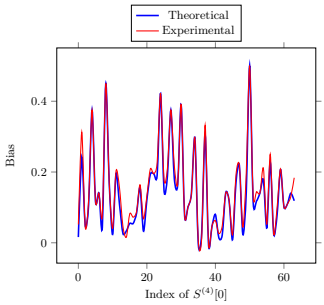
(b) $\Delta(0, 35)$



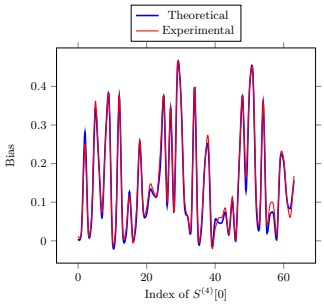
(c) $\Delta(0, 36)$



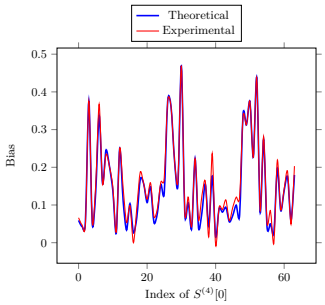
(a) $\Delta(0, 37)$



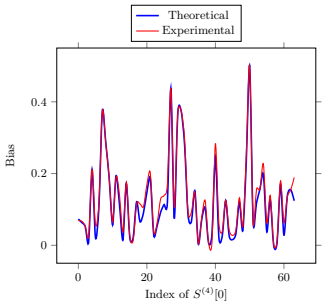
(b) $\Delta(0, 38)$



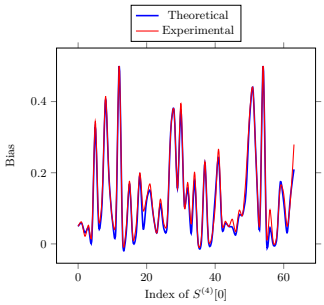
(c) $\Delta(0, 39)$



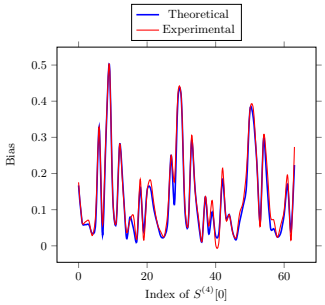
(a) $\Delta(0, 40)$



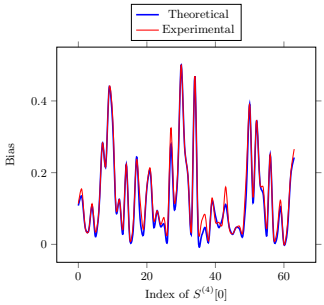
(b) $\Delta(0, 41)$



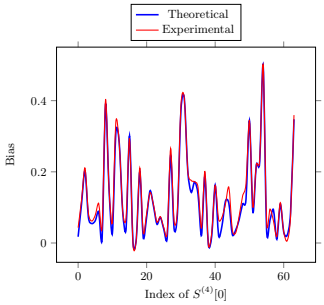
(c) $\Delta(0, 42)$



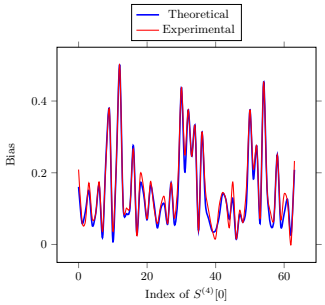
(a) $\Delta(0, 43)$



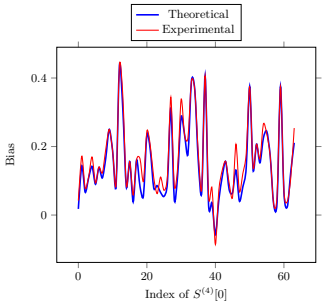
(b) $\Delta(0, 44)$



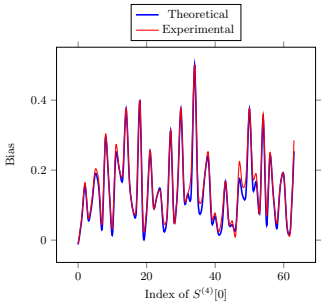
(c) $\Delta(0, 45)$



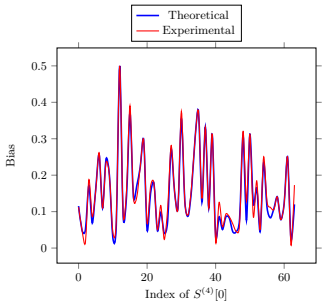
(a) $\Delta(0, 46)$



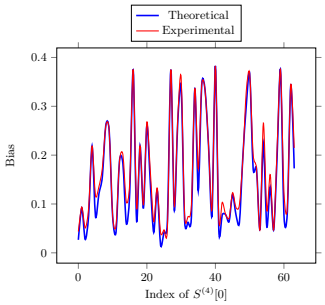
(b) $\Delta(0, 47)$



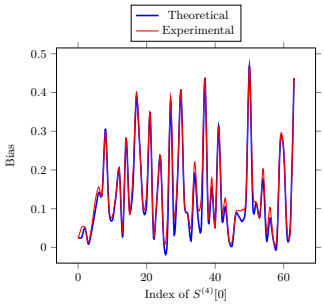
(c) $\Delta(0, 48)$



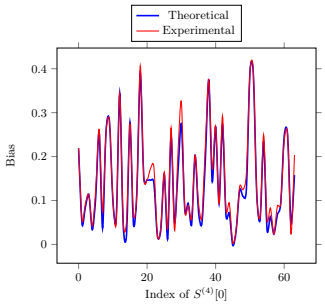
(a) $\Delta(0, 49)$



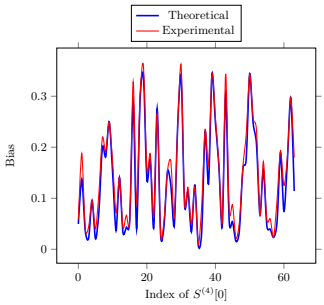
(b) $\Delta(0, 50)$



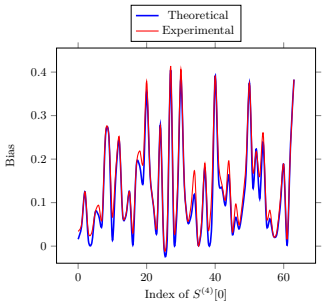
(c) $\Delta(0, 51)$



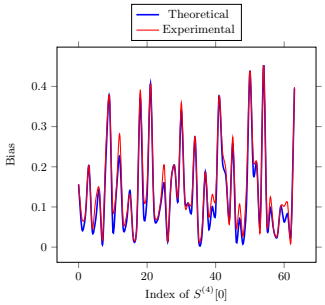
(a) $\Delta(0, 52)$



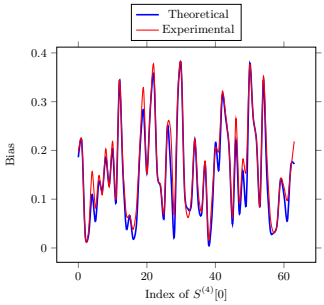
(b) $\Delta(0, 53)$



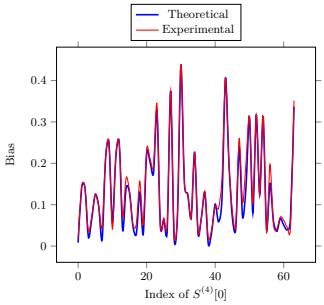
(c) $\Delta(0, 54)$



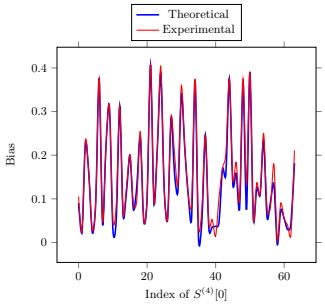
(a) $\Delta(0, 55)$



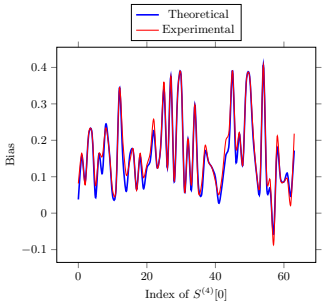
(b) $\Delta(0, 56)$



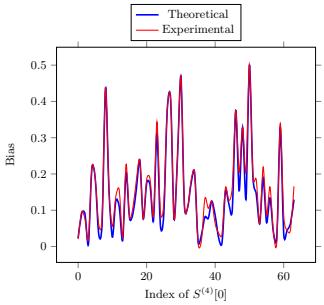
(c) $\Delta(0, 57)$



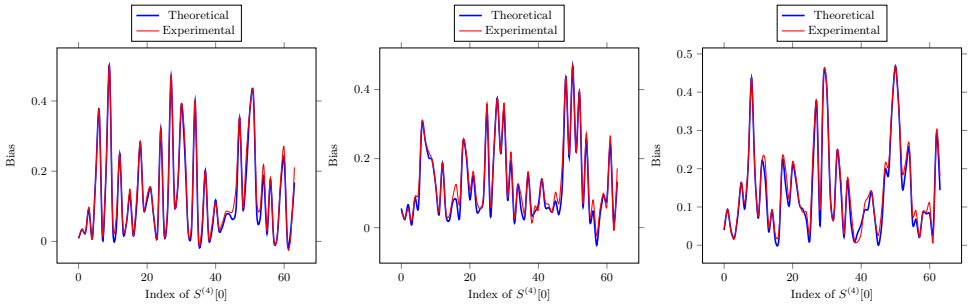
(a) $\Delta(0, 58)$



(b) $\Delta(0, 59)$

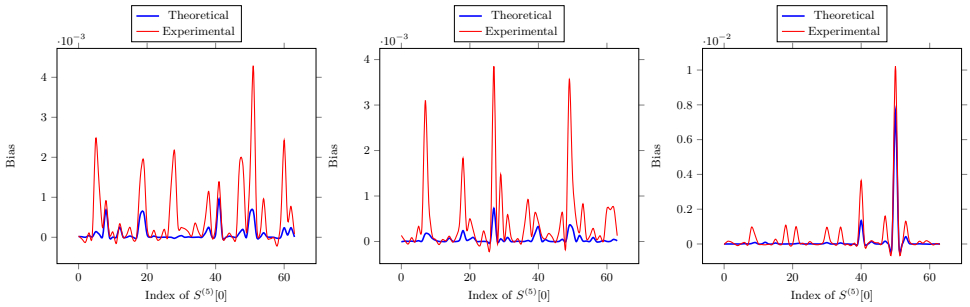


(c) $\Delta(0, 60)$

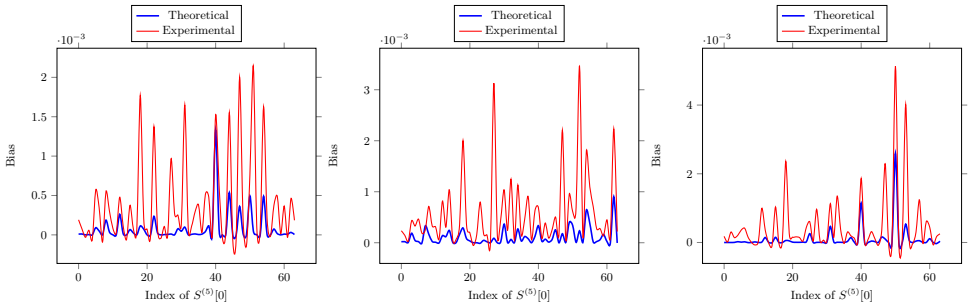


(a) $\Delta(0, 61)$ (b) $\Delta(0, 62)$ (c) $\Delta(0, 63)$

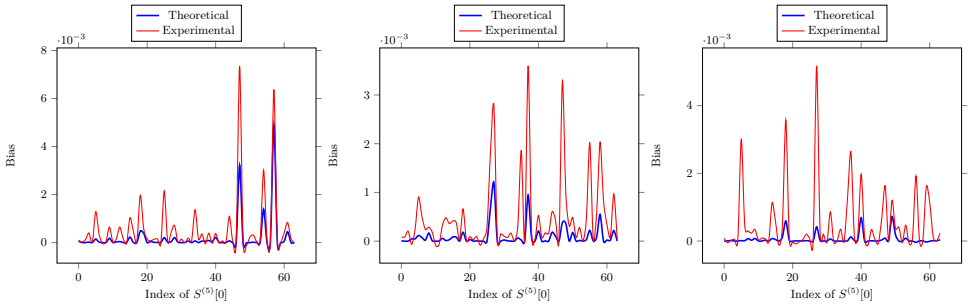
J.2 Figures of Second-Order HDL Biases for 5-Round Ascon



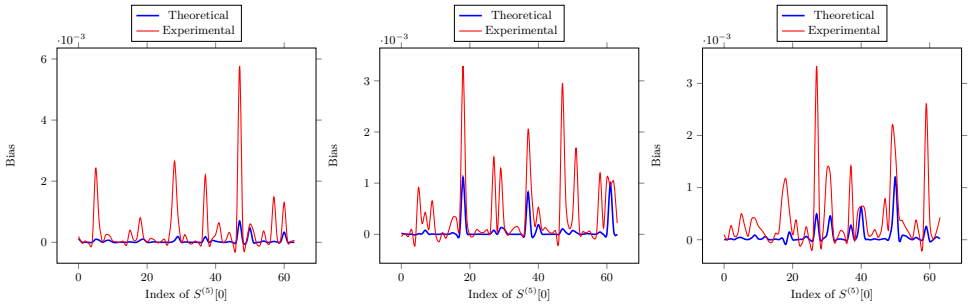
(a) $\Delta(0, 1)$ (b) $\Delta(0, 2)$ (c) $\Delta(0, 3)$



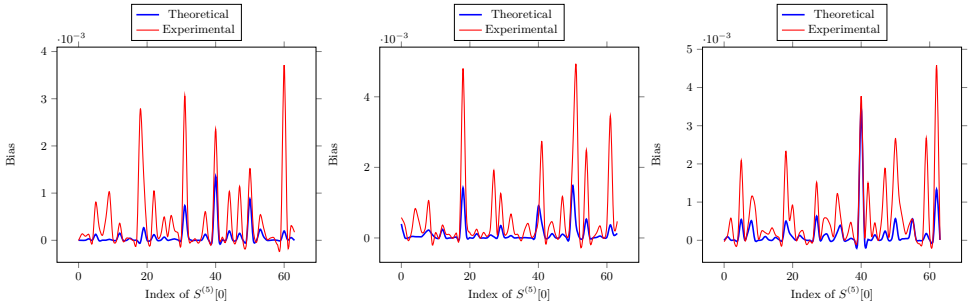
(a) $\Delta(0, 4)$ (b) $\Delta(0, 5)$ (c) $\Delta(0, 6)$



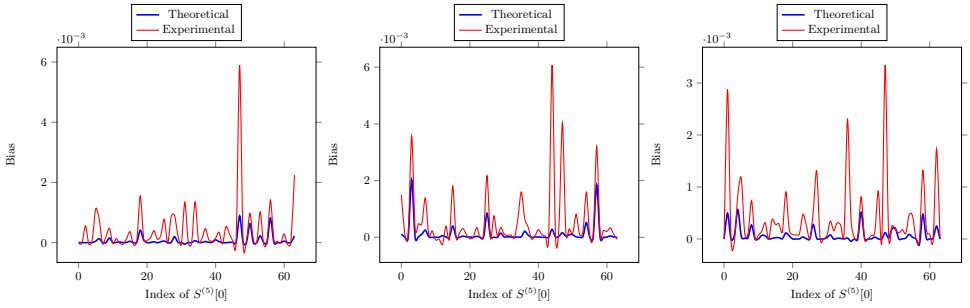
(a) $\Delta(0,7)$ (b) $\Delta(0,8)$ (c) $\Delta(0,9)$



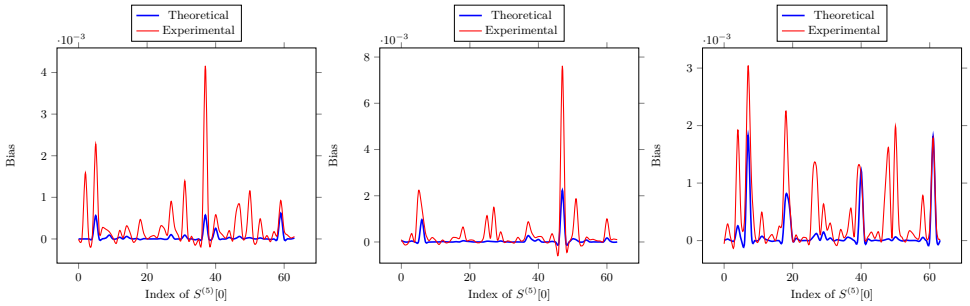
(a) $\Delta(0,10)$ (b) $\Delta(0,11)$ (c) $\Delta(0,12)$



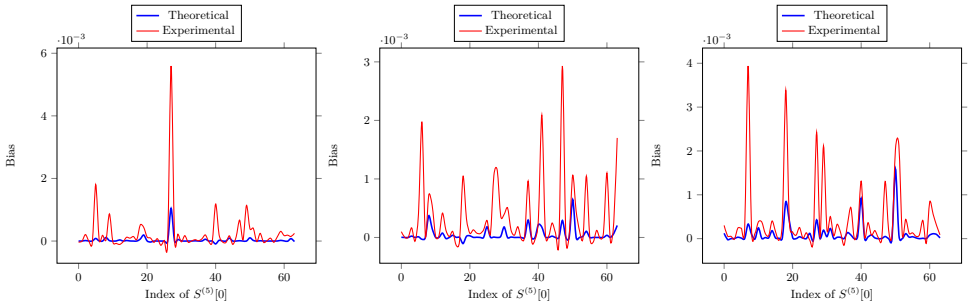
(a) $\Delta(0,13)$ (b) $\Delta(0,14)$ (c) $\Delta(0,15)$



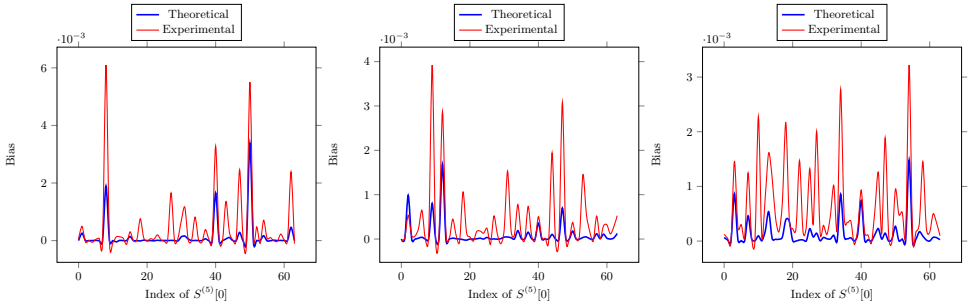
(a) $\Delta(0,16)$ (b) $\Delta(0,17)$ (c) $\Delta(0,18)$



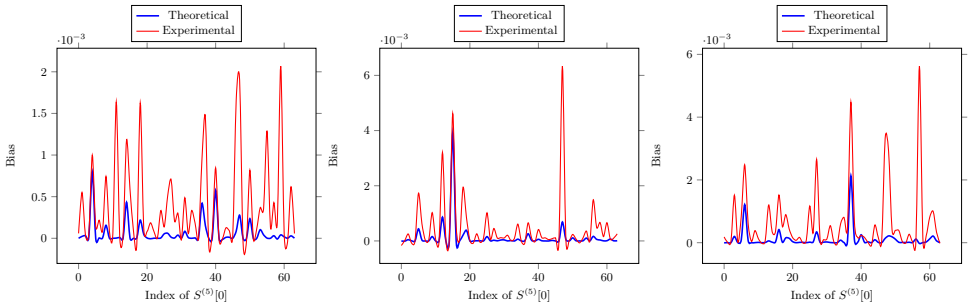
(a) $\Delta(0, 19)$ (b) $\Delta(0, 20)$ (c) $\Delta(0, 21)$



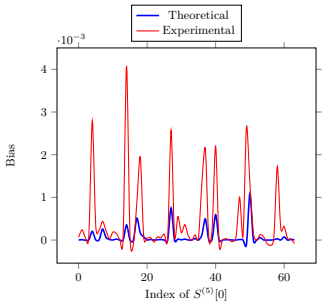
(a) $\Delta(0, 22)$ (b) $\Delta(0, 23)$ (c) $\Delta(0, 24)$



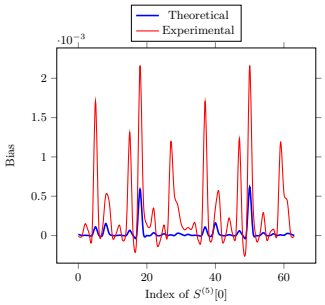
(a) $\Delta(0, 25)$ (b) $\Delta(0, 26)$ (c) $\Delta(0, 27)$



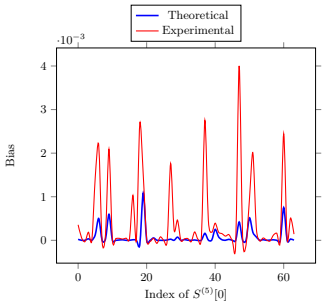
(a) $\Delta(0, 28)$ (b) $\Delta(0, 29)$ (c) $\Delta(0, 30)$



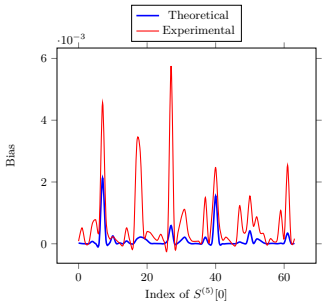
(a) $\Delta(0, 31)$



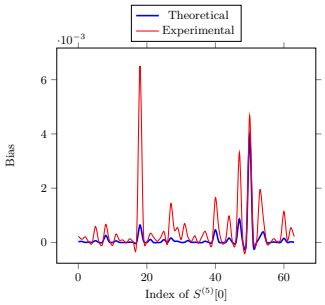
(b) $\Delta(0, 32)$



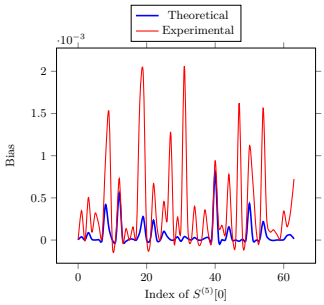
(c) $\Delta(0, 33)$



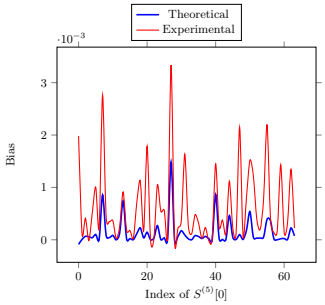
(a) $\Delta(0, 34)$



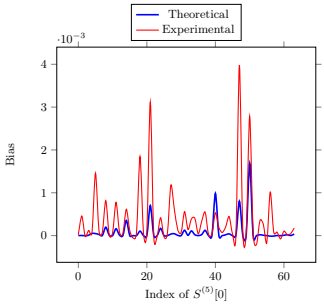
(b) $\Delta(0, 35)$



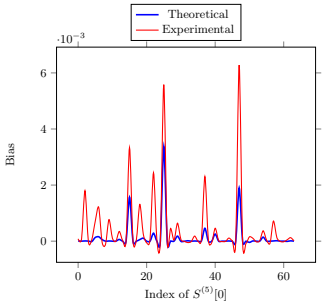
(c) $\Delta(0, 36)$



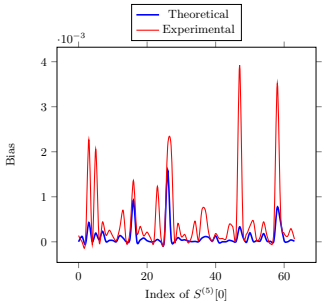
(a) $\Delta(0, 37)$



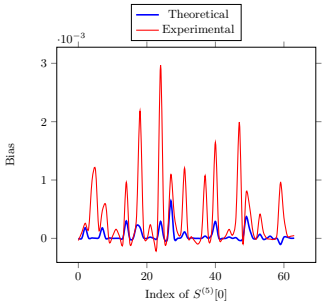
(b) $\Delta(0, 38)$



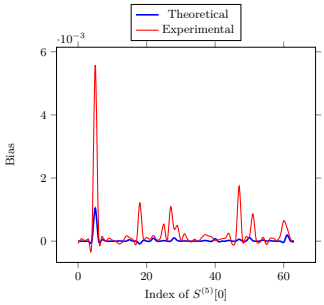
(c) $\Delta(0, 39)$



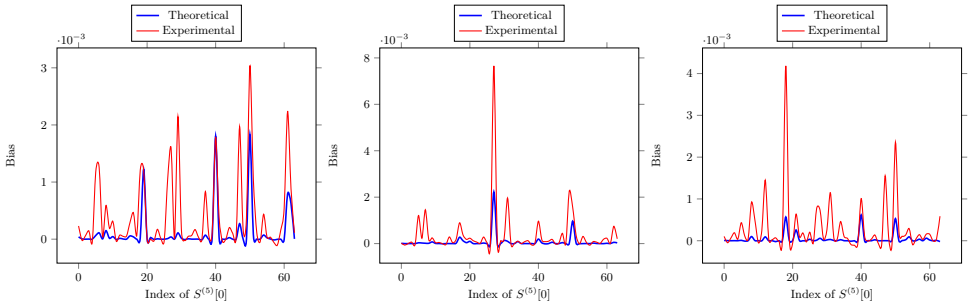
(a) $\Delta(0, 40)$



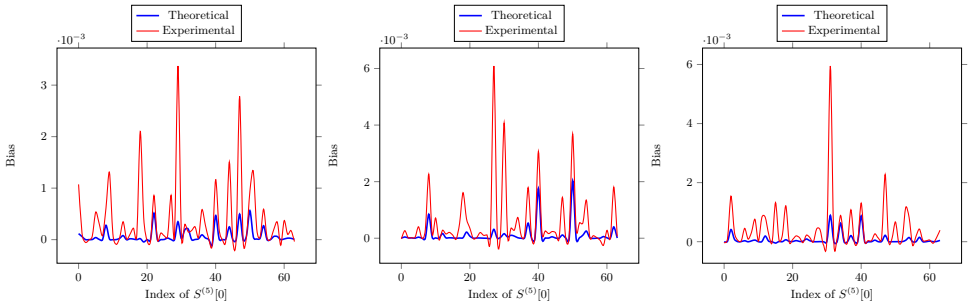
(b) $\Delta(0, 41)$



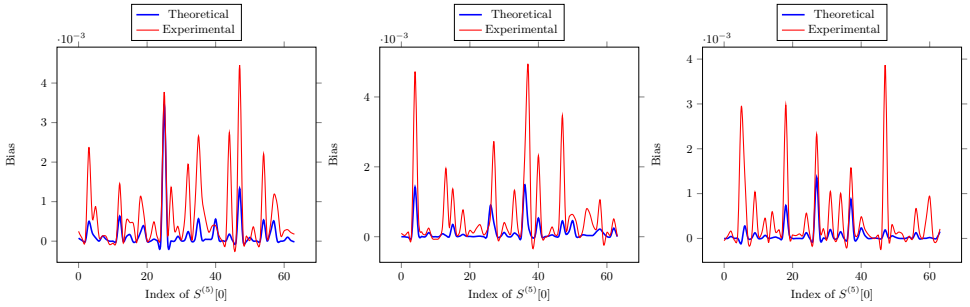
(c) $\Delta(0, 42)$



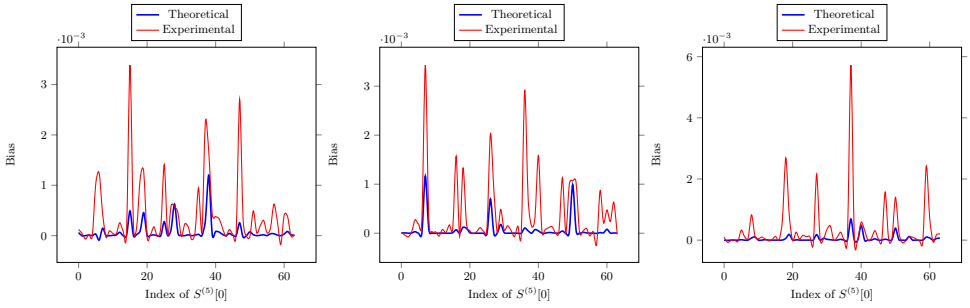
(a) $\Delta(0, 43)$ (b) $\Delta(0, 44)$ (c) $\Delta(0, 45)$



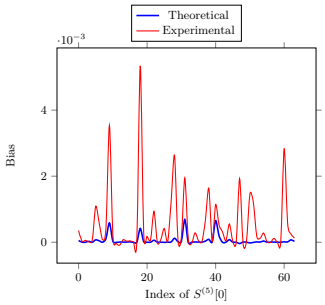
(a) $\Delta(0, 46)$ (b) $\Delta(0, 47)$ (c) $\Delta(0, 48)$



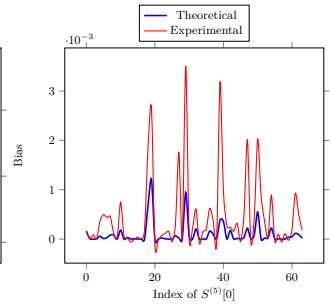
(a) $\Delta(0, 49)$ (b) $\Delta(0, 50)$ (c) $\Delta(0, 51)$



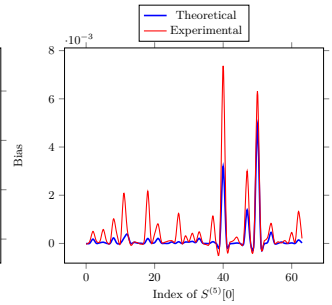
(a) $\Delta(0, 52)$ (b) $\Delta(0, 53)$ (c) $\Delta(0, 54)$



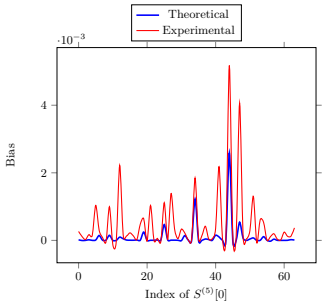
(a) $\Delta(0, 55)$



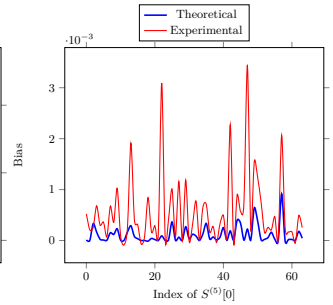
(b) $\Delta(0, 56)$



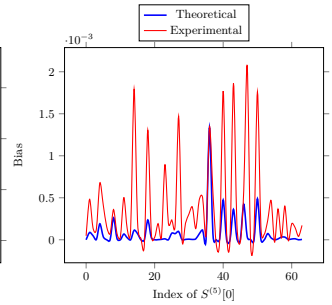
(c) $\Delta(0, 57)$



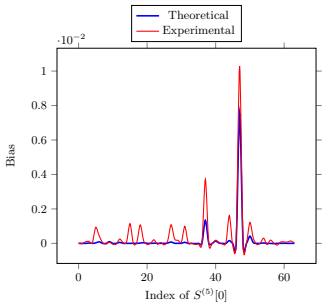
(a) $\Delta(0, 58)$



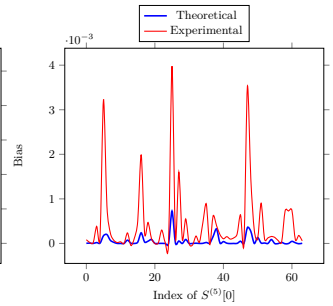
(b) $\Delta(0, 59)$



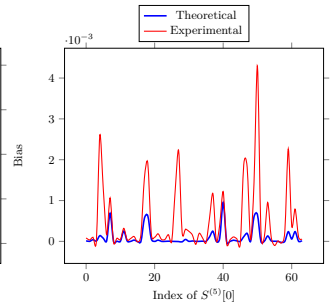
(c) $\Delta(0, 60)$



(a) $\Delta(0, 61)$

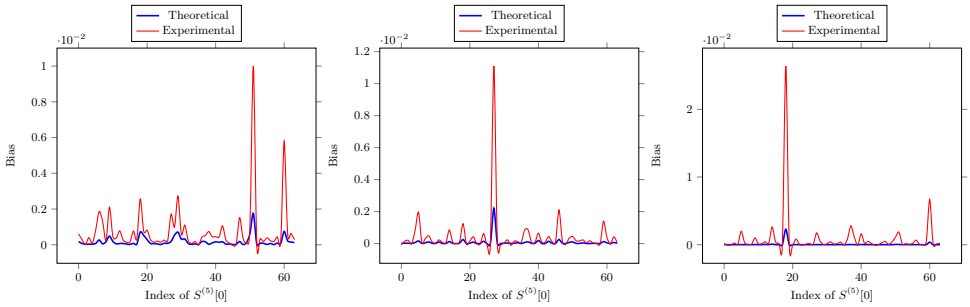


(b) $\Delta(0, 62)$

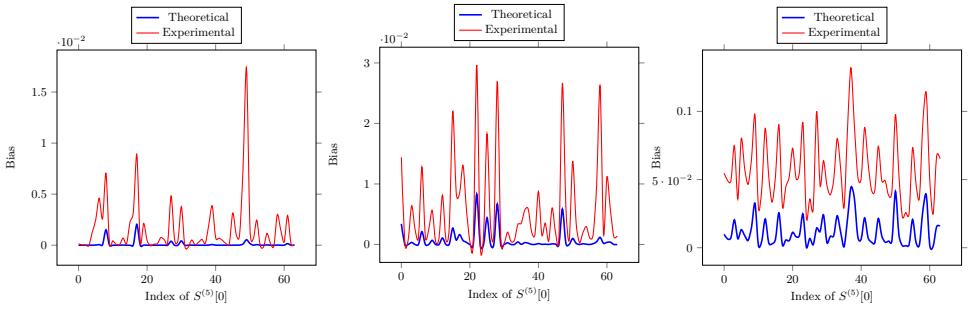


(c) $\Delta(0, 63)$

J.3 Figures of HDL Biases for 5-Round Ascon for 3rd to 8th Order



(a) $\Delta(0, 24, 33)$ (b) $\Delta(0, 9, 15, 41)$ (c) $\Delta(0, 9, 24, 51, 55)$



(a) $\Delta(1, 12, 18, 22, 21, 52)$ (b) $\Delta(10, 13, 21, 31, 49, 55, 61)$ (c) $\Delta(0, 8, 9, 13, 14, 26, 43, 60)$