Revisiting Higher-Order Differential(-Linear) Attacks from an Algebraic Perspective

Applications to Ascon, Grain v1, Xoodoo, and ChaCha

Kai Hu and Thomas Peyrin

School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
`kai.hu@ntu.edu.sg, thomas.peyrin@ntu.edu.sg`

Abstract. The higher-order differential-linear (HDL) attack was studied for the first time by Biham, Dunkelman, and Keller at FSE 2005, where a linear approximation is appended to a higher-order differential (HD) transition. It is a natural generalization of the differential-linear (DL) attack. Due to some restrictions in practical usage, unfortunately, the HDL cryptanalysis has attracted much less attention compared to its DL counterpart since its proposal. Inspired by the algebraic perspective on DL attacks recently proposed at CRYPTO 2021, in this paper we show that the HDL attack can be made much more practical with an algebraic treatment, turning this 17-year-old attack into another go-to tool for cryptanalysts.

Unsurprisingly, HD/HDL attacks have the potential to be more effective than their simpler differential/DL counterpart. We provide three novel methods to detect possible HD/HDL distinguishers, including: (a) an estimation of the algebraic degree of the differential supporting function (DSF); (b) the higher-order algebraic transitional form (HATF); (c) experimental methods based on cube testers. With these methods, we greatly improve the distinguishing attacks on the 8-round Ascon permutation under the black-box model from 2^{130} to 2^{46}. Also, we give a new zero-sum distinguisher for a full 12-round Ascon permutation with only 2^{55} time/data complexity, improving over the previous best one that requires 2^{130} calls (we make clear that this does not impact the full Ascon AEAD scheme). For the 4-round Ascon initialization, a deterministic 2nd order HDL distinguisher is proposed with only four nonces. Besides the distinguishers, the HATF technique allows us to handle the probabilistic HD/HDL properties of cryptographic primitives. This leads to a conditional HDL attack on 5-round Ascon initialization that can recover all the key bits, performing 8 times faster than the conditional DL attack. To the best of our knowledge, this is the first theoretical work to propose a probabilistic HDL attack since it was first published. All our attacks in this paper apply to both Ascon-128 and Ascon-128a. We also give a conditional HD approximation for 130-round Grain v1 (reaching 5 more rounds than the previous best conditional differential approximation) and new 4-round deterministic HDL distinguishers for the Xoodoo permutation with only four chosen plaintexts. Finally, we applied our strategy to the ARX-based cipher ChaCha, obtaining 3.5-, 4- and 4.5-round distinguishers and again improving over the state-of-the-art. Our cryptanalyses do not threaten the security of the ciphers mentioned in this paper.

Keywords: Higher-Order Differential, Higher-Order Differential-Linear, Ascon, Xoodoo, Grain v1, ChaCha

1 Introduction

1.1 Differential and Linear Cryptanalysis

Differential cryptanalysis was proposed in [BS90] as an approach to analyzing the security of DES-like cryptosystems. In a differential attack, the attacker seeks a fixed input difference Δ_I that propagates through the target cipher to a fixed output difference Δ_O with a high probability. The so-called differential is denoted by $\Delta_I \xrightarrow{p} \Delta_O$, where p is the probability $\Pr[C \oplus C' = \Delta_O | P \oplus P' = \Delta_I]$ and C/C' being the ciphertexts corresponding to the plaintexts P/P' respectively. If p is significantly larger than 2^{1-n}, where n is the block size of the cipher, the differential can be used for distinguishing it from a random permutation.

Linear cryptanalysis [Mat93] was also originally proposed to attack the DES cipher. In linear cryptanalysis, the attacker studies the bias of the approximation between the parity of some plaintext and ciphertext bits. The bias q with the input and output masks (λ_I, λ_O) can be computed with $\Pr[P \cdot \lambda_I = C \cdot \lambda_O] = \frac{1}{2} + q$, where $a \cdot b = \bigoplus_{i=0}^{n-1} a[i]b[i]$ for $a, b \in \mathbb{F}_2^n$. Such a linear approximation is denoted by $\lambda_I \xrightarrow{q} \lambda_O$. If $|q|$ is significantly larger than 0, it is possible to distinguish the cipher from a random permutation.

1.2 Differential-Linear Cryptanalysis

Differential and linear cryptanalysis have been the fundamental methods for evaluating the security of a cipher. Nowadays, all new schemes are requested to claim resistance against these two attacks, *e.g.*, [DR02,BJK+16]. However, resistance against the plain differential and linear cryptanalysis does not necessarily lead to a resistance against variants of these two attacks. 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]. 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 $|q'|$ is significantly larger than 0, we can distinguish the cipher from a random permutation.

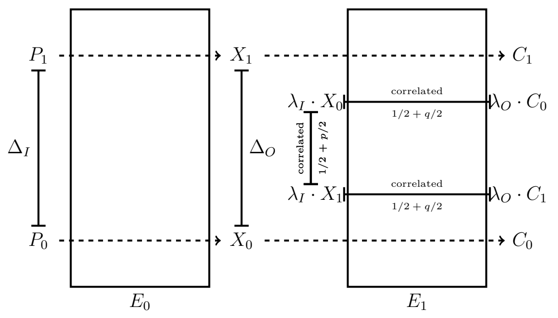


Fig. 1: The differential-linear cryptanalysis on a cipher $E = E_1 \circ E_0$, with a differential $\Delta_I \xrightarrow{E_0} \Delta_O$ whose probability is p and a linear approximation $\lambda_I \xrightarrow{E_1} \lambda_O$ whose bias is q.

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 there is a differential $\Delta_I \xrightarrow{p} \Delta_O$ for E_0 and a linear approximation $\lambda_I \xrightarrow{q} \lambda_O$ for E_1. The DL bias q' can be analyzed as follows (see Figure 1 for the illustration). Let (P, P') be the chosen plaintext pair with difference Δ_I, (X, X') and (C, C') be the corresponding intermediate state pair (between E_0 and E_1) and ciphertext pair. The DL approximation for E then combines three approximations: the values of $\lambda_O \cdot C$ and $\lambda_O \cdot C'$ are correlated to $\lambda_I \cdot X$ and $\lambda_I \cdot X'$, respectively, by $\lambda_I \xrightarrow{q} \lambda_O$ for E_1; the values $\lambda_I \cdot X$ and $\lambda_I \cdot X'$ are correlated, as consequences of $\Delta_I \xrightarrow{p} \Delta_O$ for E_0. Under two assumptions: (a) E_0 and E_1 are independent; (b) When $X \oplus X' \neq \Delta_O$, $\lambda_O \cdot C$ and $\lambda_O \cdot C'$ are correlated to $\lambda_I \cdot X$ and $\lambda_I \cdot X'$ with probability $\frac{1}{2}$ respectively, the overall bias q' can be computed with $q' = (-1)^{\Delta_O \cdot \lambda_I} 2pq^2$ with the well-known piling-up lemma [Mat93].

As pointed out in [BDK02], these two assumptions may fail sometimes, so experiments are required to verify the estimated bias when possible. There are two main refined methods to avoid the assumptions issue. One is from Blondeau *et al.* [BLN17], where an accurate formula for q' is given under only the first assumption. The other, proposed by Bar-On *et al.* [BDKW19] at EUROCRYPT 2019, is called the differential-linear connectivity table (DLCT) technique 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

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

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.*, $x\Delta_I = (\Delta_I[0] \cdot x, \dots, \Delta_I[n-1] \cdot x)$. Given a Boolean function $g(a_0, a_1, \dots, a_{n-1})$ 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, 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),

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

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.

1.3 Higher-Order Differential(-Linear) Cryptanalysis

Inspired by the boomerang and DL cryptanalysis, other combined attacks were studied by Biham *et al.* [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 $\boldsymbol{\Delta}_I = (\Delta_0, \Delta_1, \dots, \Delta_{\ell-1})$ where $\Delta_0, \Delta_1, \dots, \Delta_{\ell-1}$ are linearly independent, the ℓ-th derivative of a (partial) cipher E with respect to $\boldsymbol{\Delta}_I$ studies the probability

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

where $\mathcal{L}(\boldsymbol{\Delta}_I)$ is the linear span of $(\Delta_0, \Delta_1, \dots, \Delta_{\ell-1})$, 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. In [Tie17], Tiessen pointed that a HD is a cluster of so-called d-differentials from polytopic cryptanalysis [Tie16]. However, since the number of d-differentials is exponential and every single d-difference has an extremely low probability, it is computationally impossible to calculate the probability of an HD or even some useful lower bounds in a differential-like way. Therefore, usually

[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.

only the deterministic property from the algebraic degree of a cipher [BCD11], or the integral attack [KW02] are considered in previous HD cryptanalysis.

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, as we mentioned, 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].

The ideas of HDL were also ever used in the context of cryptanalysis of Salsa [Ber08b] and ChaCha [Ber08a] although no one explicitly called it HDL cryptanalysis. In [SZFW12], Shi *et al.* provided several highly-biased 2nd derivatives with one active output bit for 4-round Salsa and 3-round ChaCha based on experiments. As pointed in [BLN17,LLL21], differentials with one active output bit are DL distinguishers with unit output masks, the observation works for the HD as well. Thus, Shi *et al.*'s distinguishers are also HDL distinguishers. In [CM16], Choudhuri and Maitra extended Shi *et al.*'s HD by appending a linear approximation, which is the typical case of HDL (note that they did not call it HDL nor gave many discussions on this topic).

1.4 Motivation and Contributions

Considering that DL attacks have been found to be efficient for many important primitives, such as Ascon [DEMS21] and Salsa/ChaCha [Ber08b,Ber08a], we are naturally interested in whether the HDL attack could achieve even better performance. However, as we mentioned, we do not have many tools to study the general form of the HD properties of a cipher, especially the probabilistic ones. Consequently, the HDL cryptanalysis is currently far less practical than its differential counterpart, *i.e.*, the DL cryptanalysis.

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]. We note that their method is based on some intuitive observations and is limited to the first-order case. In this paper, we study the underlying algebraic structure of this new perspective and finally we manage to generalize it to the more general higher-order cases, *i.e.*, HDL cryptanalysis.

Our contributions. Our results range from theory to application, so our contributions are generally two-fold. In the theoretical part, we revisit the HD/HDL cryptanalysis of a Boolean function from the algebraic perspective, which provides a novel method to study HD and HDL cryptanalysis. Especially, as far as we know, our method is the first theoretical tool for studying the *probabilistic* HD and HDL distinguishers. On the applications side, we give three methods for HD/HDL cryptanalysis based on the study of the so-called *differential supporting function* (DSF). Improvements over the state-of-the-art for several primitives are thus obtained. We are confident that with the techniques we propose in this paper, HDL cryptanalysis is now another handy weapon in the cryptanalyst's toolbox.

1. By twisting the input set of a Boolean function f from an ℓ dimensional affine space to an ℓ dimensional linear space, we succeed in transforming an HD of f to a standard integral/cube attack. Thus, any HD attack on f is equivalent to the cube or integral attack on this Boolean function's DSF. This gives us a unified viewpoint for differential, HD, cube, and integral attacks. Based on the DSF, we can analyze the algebraic properties of a cryptographic primitive in a more general and systematic way, while previous methods seem more intuitive and empirical. This greatly deepens our general understanding of the relationship between these algebraic attacks.
2. We provide three methods to mount HD attacks on a Boolean function f by analyzing its DSF (which can be used to mount HDL attacks on concrete instances as well). All HD/HDL approximations for various primitives we obtained in this paper (as well as their previous DL approximations) are summarized in Table 1.
   (a) Instead of using the degree evaluation on f to derive HD distinguishers, we can evaluate the algebraic degree or find integral distinguishers for 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 an integral distinguisher for the DSF. After that, we can conveniently transform it into an HD distinguisher for f. With this technique, we significantly improve the best-known distinguishing attacks on round-reduced Ascon permutation [DEMS21]. A 46th HD will lead to a zero output difference (in 64 bits) for 8 rounds, *i.e.*, 2^{46} plaintexts are enough to distinguish an 8-round Ascon permutation from a random permutation (the previous best distinguisher requires 2^{130} computations [Tod15]). This is the first distinguisher with complexity being lower than 2^{64} for 8-round Ascon permutation. 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. We

also give a 2nd order HDL distinguisher for 4 rounds of the Ascon initialization with bias equal to $\frac{1}{2}$, which means we can use 4 nonces to distinguish it from a random permutation. These distinguishers are demonstrated in the distinguisher part of Table 2. We emphasize that our results do not threaten the security of the Ascon AEAD scheme.

(b) We propose the higher-order algebraic transitional form (HATF) to estimate the bias of a probabilistic HDL approximation, which is the first systematic method to handle this long-standing problem. With HATF, we can construct efficiently a transitional expression of the DSF and then evaluate its bias. We detected four conditional HDL approximations with a bias of only 2^{-2} for 5-round Ascon initialization. By analyzing the conditions in this distinguisher, we could obtain the best key-recovery attack on 5-round Ascon with time/data complexity 2^{23}, which is 8 times faster than its DL counterpart. We also found an HDL approximation with 2^{-30} bias for 6-round Ascon initialization that is outside the scope of DL cryptanalysis. With some reasonable assumptions, we could mount a key-recovery attack on 6-round Ascon initialization. This is the first time that DL-like attacks succeed for more than 5 rounds besides the cube-like attacks [LDW17,RHSS21]. A summary of these key-recovery attacks is given in the key-recovery part of Table 2. Note that these attacks apply to both Ascon-128 and Ascon-128a. To further illustrate the powerfulness of the HATF, we applied it to Grain v1 [HJM07] to get a conditional HD approximation for 130 rounds, which is 5 rounds longer than the previous best conditional DL approximation [LLL21]. We remark that since the HATF technique is a heuristic method based on some new assumptions, we provide experimental data to back our theoretical results when possible.

(c) Finally, we applied the cube tester to the DSF. This is an experimental method that was used in the scope of DL cryptanalysis in many previous works. This method works for all kinds of primitives. We first applied cube testers to the initialization of the Ascon AEAD and found more highly-biased HDL approximations. For example, we detected an 8th order HDL with bias equal to only $2^{-2.46}$ for 5 rounds, and thus we can use about 2^{13} data/time complexity to distinguish 5 rounds of the Ascon initialization (the previous best one requires 2^{16} [RHSS21]). If we impose 16 conditions on the key, the bias can even improve to $\frac{1}{2}$, *i.e.*, for 2^{112} keys, 5-round Ascon initialization could be distinguished with 2^8 data/time complexities. By analyzing the nonlinear operations of Xoodoo [DHAK18], we choose some specific forms of the input values and differences. An exhaustive search within a small space returned a deterministic HDL distinguisher for 4-round Xoodoo (the bias is $\frac{1}{2}$). For ChaCha [Ber08a], we first searched for some efficient HDL distinguishers for 2-, 2.5, and 3-round ChaCha permutation, then appended a 1.5-round linear approximation with bias equal to $\frac{1}{2}$ to extend the distinguishers to 3.5, 4 and 4.5 rounds. The biases of these three HDL approximations

Table 1: Approximation Biases of the differential, DL and HDL for ciphers considered in this paper. Cond. is short for Conditional and means that we impose some conditions on the key bits.

Primitive	**Round**	**Bias** ($-\log$)	**Type**	**Reference**
Ascon Init.	4	2	DL	[DEMS15]
		1	**2nd order HDL**	**Section 4.4**
	5	9	DL	[DEMS15]
		4.54	Con. DL	[LLL21]
		2.46	**8th order HDL**	**Section I.1**
		2	**Con. 2nd order HDL**	**Section 5.3**
		1.04	**11th order HDL**	**Section I.1**
		1	**Con. 8th order HDL**	**Section I.1**
	6	**30**	**Con. 2nd order HDL**	**Sup.Mat. I.1**
Grain v1	120	12.8	Cond. diff.	[LG19]
	125	17.4	Cond. diff.	[LLL21]
	130	**30.18**	**Cond. 2nd order diff.**	**Sup.Mat. H**
Xoodoo	4	1	Rotational DL	[LSL21]
		1	**2nd order HDL**	**Sup.Mat. I.2**
ChaCha	3.5	**1.00**	**2nd order HDL**	**Sup.Mat. I.3**
	4	3.33	DL	[CM16]
		2.21	2nd order HDL	[CM16]
		1.19	**2nd order HDL**	**Sup.Mat. I.3**
	4.5	6.14	DL	[CM16]
		4.81	**2nd order HDL**	**Sup.Mat. I.3**

are significantly higher than DL approximations. A summary of these results is provided in Table 1.

The source codes of this work are provided in the anonymous git repository `https://anonymous.4open.science/r/HDL-CC85`. This can be seen as an extra practical contribution, as the team of [LLL21] did not make their code public.

Outline. In Section 2, we briefly recall the main concepts of the HD and the algebraic perspective on the differential attack. In Section 3, we provide the algebraic perspective on the HD/HDL and give a simple and direct formula for the transformation between HD/HDL and cube/integral cryptanalysis. The definition of the DSF is also introduced in this section. In the following Sections 4, 5 and Section I of Supplementary Material, three novel techniques are provided to detect possible HD/HDL distinguishers based on analyzing the DSF. In Section 6, we do some discussions and conclude this paper.

Table 2: Summary of results on the permutation (black-box mode) (labelled by 🔴), permutation (non-black-box mode) (labelled by 🔷), initialization (labelled by 🟦), encryption (labelled by ▲) and initialization under the weak-key model (labelled by 🟧) of reduced-round Ascon. Time complexities are expressed in number of primitive calls while the data complexities are measured by the number of 128-bit blocks, *i.e.*, 128-bit blocks. Our distinguishers up to 7 rounds have been verified experimentally.

Type	**Rnd**	**Data**(log)	**Time** (log)	**Method**	**Reference**
Distinguisher‡	4	16	16	Rectangle 🔴	[GPT21]
		8	8	Limited-birthday 🔴	[GPT21]
		5	5	Integral 🟦	[RHSS21]
		5	5	DL 🟦	[DEMS15]
		3	**3**	**HD** 🔴	**Section 4.2**
	5	18	18	Integral 🔴	[Tod15]
		18	18	DL 🟦	[DEMS15]
		16	16	Integral 🟦	[RHSS21]
		13	**13**	**HDL** 🟦	**Section I.1**
		9	9	Degree 🟧 🔴	[RS21]
		6	**6**	**HD** 🔴	**Section 4.2**
	6	35	35	Integral 🔴	[Tod15]
		31	31	Integral 🟦	[RHSS21]
		17	17	Degree 🟧 🔴	[RS21]
		12	**12**	**HD** 🔴	**Section 4.2**
	7	65	65	Integral 🔴	[Tod15]
		60	60	Integral 🟦	[RHSS21]
		33	33	Degree 🟧 🔴	[RS21]
		23	**23**	**HD** 🔴	**Section 4.2**
	8	130	130	Integral 🔴	[Tod15]
		46	**46**	**HD** 🔴	**Section 4.2**
	12	130	130	Zero-sum (partition)†🔷	[DEMS15]
		55	**55**	**Zero-Sum** 🔷	**Section 4.3**
Key-Recovery	5	36	36	DL 🟦	[DEMS15]
		26	26	Cond. DL 🟦	[LLL21]
		24	24	Cond. Cube 🟦	[LDW17]
		23	**23**	**Cond. HDL** 🟦	**Section 5.3**
	6	40	40	Cond. Cube 🟦	[LDW17]
		74	**74**	**Cond. HDL** 🟦	**Section 5.3**
	7	77	103	Cond. Cube 🟦	[LDW17]
		64	123	Cube 🟦	[RHSS21]

‡ Due to space limitations, we did not list all existing distinguishers for 4 round 5 rounds of Ascon primitives. Please refer to [DEMS21,GPT21,Tez16] for more details on them.

† The zero-sum distinguisher in [DEMS15] can be further extended to a zero-sum partition distinguisher.

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 $\boldsymbol{x} = (x_0, x_1, \dots, x_{\ell-1}) \in (\mathbb{F}_2^n)^\ell$, 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 $\Delta \in \mathbb{F}_2^n$, $x\Delta = (\Delta[0]x, \Delta[1]x, \dots, \Delta[n-1]x)$[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 $a \cdot b = \bigoplus_{0 \leq i < n} a[i]b[i]$.

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)

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

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 $\pi_u(\boldsymbol{x})$ of f is denoted by $\mathsf{Coe}(f, \pi_u(\boldsymbol{x}))$. 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. Therefore, in this paper, we usually explain our theories with Boolean functions rather than concrete primitive instances.

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 method to deal with 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

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

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 proof nor clear motivation was given in their article. In the next section, we will make it clearer by introducing the algebraic perspective on the ℓ-th order HD.

3 HD/HDL Cryptanalysis from an Algebraic Perspective

In this section, we show how to treat the ℓ-th derivative of a Boolean function f with respect to an ℓ-th order difference from an algebraic perspective by establishing a bijective mapping to twist the input set of f. Liu *et al.*'s algebraic perspective on DL is then a special case of our theory when the order $\ell = 1$. Interestingly, from our algebraic treatment, the underlying rationale of Equation (3) including the reason for introducing the auxiliary variable x becomes natural and clear.

3.1 HD/HDL Cryptanalysis from an Algebraic Perspective

Given a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ and an input ℓ-th order 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 of f $X \in \mathbb{F}_2^n$. 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 we are operating an ℓ-dimensional affine space $\mathbb{A}^\ell = X \oplus \mathcal{L}(\boldsymbol{\Delta})$. An important observation is that 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 $\mathbb{F}_2^\ell$. With *a method of undetermined coefficients*, one choice of $\mathcal{M}^\ell$ can be

$$\begin{aligned} \mathcal{M}^\ell : \mathbb{F}_2^\ell &\to \mathbb{A}^\ell \\ (x_0, x_1, \ldots, x_{\ell-1}) &\mapsto X \oplus x_0\Delta_0 \oplus x_1\Delta_1 \oplus \cdots \oplus x_{\ell-1}\Delta_{\ell-1} \triangleq X \oplus \boldsymbol{x\Delta} \end{aligned} \tag{4}$$

We define a new function $f_{\boldsymbol{\Delta}}$ from f with the twisted input set as $f_{\boldsymbol{\Delta}} \triangleq f(X \oplus \boldsymbol{x\Delta})$:

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

If we let $D_{\boldsymbol{x}} f_{\boldsymbol{\Delta}}$ represent the coefficient of the maxterm in $f_{\boldsymbol{\Delta}}$, *i.e.*, $D_{\boldsymbol{x}} f_{\boldsymbol{\Delta}} = \mathsf{Coe}\left(f(X \oplus \boldsymbol{x\Delta}), \pi(\boldsymbol{x})\right)$, 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 (4), 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,

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

□

When the information of the order ℓ is clear in the context, we will use $\mathcal{M}$ to represent the mappings defined in Equation (4). According to Proposition 1, we know that the ℓ-th derivative of f is equivalent to the coefficient of the maxterm of $f \circ \mathcal{M}$. Obviously, $f \circ \mathcal{M}$ plays an important role in (higher-order) differential cryptanalysis, thus we give it a formal definition:

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

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

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

We provide an example to illustrate the usage of the DSF in differential cryptanalysis.

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 2nd derivative of f at a point $X = (X_0, X_1, X_2) \in (\mathbb{F}_2)^3$. According to Equation (4), $\mathcal{M}(x_0, x_1) = X \oplus x_0\Delta_0 \oplus x_1\Delta_1 = (X_0 \oplus x_0 \oplus x_1, X_1 \oplus x_1, X_2 \oplus x_0 \oplus x_1)$. The composition of f and $\mathcal{M}$ is then

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

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

With the establishment of the DSF, a HD property of f is transformed into an integral/cube property for a related Boolean function $f_{\boldsymbol{\Delta}}$. Such transformations bring nice convenience to our study on the HD attacks, as will be shown in the following sections. Before we introduce our applications, we first make it clearer about the differences between the conceptions of HD and HDL in this paper.

The terminology of HD and HDL cryptanalysis in this paper. One nice aspect of this work is that HD and HDL can be treated almost similarly. Indeed, for a Boolean function, the HD and HDL are the same. For a cryptographic primitive with multiple output bits, however, there are slight differences between the two terminologies. Assume a cryptographic primitive E with n output bits ($n > 1$), which can be seen as a set of n Boolean functions $(f_0, f_1, \ldots, f_{n-1})$. In this paper, HDL cryptanalysis will refer to the HD properties of only one f_i or the sum of several f_i, while HD cryptanalysis will refer to at least two different f_i simultaneously. Typically, HDL will consider the bias of a single Boolean expression equal to one f_i or to the sum of several f_i, while HD will consider the probability that a certain set of f_i will each be equal to a certain Boolean value.

4 HD/HDL Cryptanalysis Based on Degree Estimation of the DSF

In this section, we show how to obtain HD distinguishers for a Boolean function by analyzing the algebraic degree of its DSF. Our first application is to Ascon [DEMS21]. Ascon is the first choice for lightweight applications recommended by the CAESAR competition[3] and now one of the NIST LWC[4] finalists. Thus, it has already attracted a lot of attention in cryptographic community and undergone repeated cryptanalysis. In this section, we present new and improved distinguishers for its permutation and initialization, which essentially reduce the complexities. Due to page limits, the description of the Ascon AEAD and its permutation is provided in Section B of Supplementary Material, we also recommend that readers refer to [DEMS21] for the whole specification.

Notations used for describing the Ascon permutation. For the Ascon permutation, 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 and 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 $S^r = (p_L \circ p_S \circ p_C)^r(S^0)$. We use $S^{r.5}$ to represent the state $p_S \circ p_C(S^r)$. For example, $S^{3.5}$ represents the state after p_S of the round 3, *i.e.*, 4 rounds without the last p_L.

4.1 Degree Matrix Transition of the Ascon Permutation

Before we introduce our core theory about the degree estimation of the DSF, we first introduce an efficient way to trace the update of algebraic degrees of the Ascon permutation state, *i.e.*, given the degrees or the upper bounds of bits in S^r, we can quickly calculate the degree upper bounds of bits in S^{r+1}. This will be useful in our HD and HDL cryptanalysis of the Ascon permutation in the remaining part of this section. To easily describe the degrees or their upper

[3] https://competitions.cr.yp.to/caesar-submissions.html
[4] https://csrc.nist.gov/News/2021/lightweight-crypto-finalists-announced

bounds of the ASCON permutation state bits, we introduce the definition of the *degree matrix*.

Definition 2 (Degree Matrix of S^r). *The algebraic degrees of the bits in the state S^r are called a degree matrix of S^r, denoted by*

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

Proposition 2 (Degree Matrix Transition over p_S). *With the knowledge of* $\mathrm{DM}(S) = (d_{i,j}, 0 \leq i < 5, 0 \leq j < 64)$*, we have* $\mathrm{DM}(p_S(S)) = (d'_{i,j}, 0 \leq i < 5, 0 \leq j < 64)$*, where* $d'_{i,j}, 0 \leq i < 5, 0 \leq j < 64$ *are computed as*

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

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

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

Proof (for Propositions 2 and 3). It is clear that if $y = x_0 \oplus x_1$, $\deg(y) \leq \max(\deg(x_0), \deg(x_1))$; if $y = x_0 x_1$, $\deg(y) \leq \deg(x_0) + \deg(x_1)$. Then from the ANFs of p_S (Equation (7)) and p_L (Equation (9)), we directly derive the formulas in Proposition 2 and Proposition 3.

Although Propositions 2 and 3 are very simple, they achieve a quite precise estimation of the upper bounds on algebraic degrees of the state bits when dealing with the ASCON permutation (which is sometimes even as good as division properties [Tod15,TM16] according to our experiments).

4.2 HD Distinguishers for the ASCON Permutation

The cryptographic permutation plays an important role in the permutation-based ciphers, so analyzing the security strength of these permutations is currently an important topic. For example, in the specification of ASCON [DEMS21], only analyses of the ASCON permutation are given including the differential/linear cryptanalysis and degree estimation (see Table 2). The target of this subsection is the ASCON permutation under the black-box model, *i.e.*, we can access the whole 320-bit input and observe the 320-bit output. 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). In this subsection, based on 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 1, $\boldsymbol{x}$ are variables while X and $\boldsymbol{\Delta}$ are parameters. Hence, different X and $\boldsymbol{\Delta}$ will lead to different DSF. Some combinations of $(X, \boldsymbol{\Delta})$ may make $\mathrm{DSF}_{f,X,\boldsymbol{\Delta}}$ simpler. 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

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

To estimate the degree upper bound of a DSF, we cut the 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 $\mathrm{DSF}_{f,X,\boldsymbol{\Delta}}$.

Regarding 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[5]. The remaining $(r-2.5)$-round permutation is seen as f^1, and the method introduced in Section 4.1 is a suitable method for f^1 to update the degrees of the output of f^0. 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. 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 64th 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:

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

[5] 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.

Algorithm 1 Detect HD Distinguishers (up to 64th order) for the Ascon permutation

```
Input: r-round Ascon permutation, $r \geq 4$
Output: $(\bar{X}, \bar{\Delta})$ leading to HD distinguishers (up to 64th order) for r-round Ascon
    permutation, the order of the HD
 1: degree = 64
 2: for $\bar{X}$ from 0 to 31 do
 3:     for $\bar{\Delta}$ from 1 to 31 do
 4:         for $i$ from 0 to 63 do
 5:             for $j$ from 0 to 4 do
 6:                 $S^0[j][i] = X[j] \oplus x_i \Delta[j]$
 7:         Compute the exact ANF of $S^{2.5}$ and compute $\mathrm{DM}(S^{2.5})$
 8:         Compute the degree matrix of $S^r$ from $S^{2.5}$ using Propositions 2 and 3
 9:         if  $\min(\mathrm{DM}(S^r)) <$ degree  then
10:             degree = $\min(\mathrm{DM}(S^r))$                ▷ To find the best distinguisher
11: return $(\bar{X}, \bar{\Delta}, \mathrm{DM}(S^r))$
```

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

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 $0 \leq i < 64$. 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 $f^0(X \oplus \boldsymbol{x\Delta})$, then derive the degree matrix of its output. After that we use Propositions 2 and 3 to update the degree matrix to calculate the degree matrix of S^r (for $r \geq 4$) which is the degree upper bound of the corresponding DSF. If the degree of a certain DSF is smaller than 64, we find useful 64th HD distinguishers for r-round Ascon permutation. The process is illustrated by Algorithm 1.

Algorithm 1 is practical. 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

$$(\bar{X}, \bar{\Delta}) \in \left\{ \begin{matrix} (\texttt{0x6}, \texttt{0x13}), (\texttt{0xa}, \texttt{0x13}), (\texttt{0xc}, \texttt{0x17}), (\texttt{0xf}, \texttt{0x18}), \\ (\texttt{0x15}, \texttt{0x13}), (\texttt{0x17}, \texttt{0x18}), (\texttt{0x19}, \texttt{0x13}), (\texttt{0x1b}, \texttt{0x17}) \end{matrix} \right\}. \quad (5)$$

In Table 3, 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 (5). As is seen, for 7 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 3 for $(X, \boldsymbol{\Delta}) = (\texttt{0x6}, \texttt{0x13})$ up to 7 rounds. According to Propositions 2 and 3, the degree upper bounds in Table 3 for 8 rounds is also verified.

Therefore, if we choose 2^{46} plaintexts in any 46-dimensional affine space defined by values in Equation 5, the summation of all ciphertexts will be zero

Table 3: Upper bounds on the algebraic degree of the DSF of the ASCON permutation with $(X, \boldsymbol{\Delta})$ in Equation (5). 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

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.

4.3 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 4.1 is also applicable to the inverse operations of the ASCON permutation. The basic process is very similar to that stated in Section 4.2, so we provide the details in Section C of Supplementary Material. Here we only give the results, which have been abstracted into Table 4[6].

According to Tables 3 and 4, 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

[6] Recalling Section 4.2, we ignore the first p_C for the forward direction. Here we include this p_C in the backward direction.

Table 4: 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.

4.4 HDL Distinguisher for Ascon Initialization and Encryption

Algorithm 1 can be easily adapted to detect HDL distinguishers for the initialization of Ascon. When targeting the initialization, we are only allowed to access the fourth and fifth words of the state and observe the first word of the output. The first word of input is filled with real IV values according to the Ascon specification, the second and third words are represented by 128 free variables for the 128-bit keys. The first p_C should also be included in the computation. This means that X is limited to $\{0, 1, 2, 3\}$ while $\varDelta$ is limited to $\{1, 2, 3\}$ in Algorithm 1, for the bottom two bits of each Sbox.

In this part, we focus on the 2nd order HDL. In other words, in line 4 of Algorithm 1, we do not fill all 64 positions, instead, we only choose 2 different positions (i_0, i_1) to impose differences and let the other positions be filled with free variables. We found many different index pairs (i_0, i_1) and $(\bar{X}, \bar{\varDelta})$ that make the algebraic degrees of some bits after 3.5-round initialization to only 1. For example, when $(i_0, i_1) = (0, 60)$ and $(\bar{X}, \bar{\varDelta}) = (0, 3)$, $\deg(S^{3.5}[50]) \leq 1$. Thus, we obtain a deterministic 2nd order HDL approximation for 4-round Ascon. One sample is enough to distinguish 4-round Ascon initialization from a random permutation (the Ascon initialization will never be judged as a random permutation). One sample contains 4 texts, so the data and time complexity is 4.

We would like to mention that one can also adapt Algorithm 1 to check the encryption phase of Ascon where we can access the first word of the input (the other four words are filled with free variables) and observe the first word of the output. For 4 rounds of encryption, when we impose differences into positions of $(0, 22)$ and $(\bar{X}, \bar{\varDelta}) = (\texttt{0x0}, \texttt{0x10})$, the degree of $S^{3.5}[0][22]$ is 1. All the HD or HDL distinguishers obtained in this section have been listed in Table 2 in Section 1.

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 $(X, \boldsymbol{\varDelta})$ 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.

5 Probabilistic HDL Cryptanalysis Based on HATF

In Section 4, we exhibited a method for deterministic HD/HDL distinguishers. In this section, we give a strategy to measure the bias of an probabilistic HDL approximation, which is the first time we have a theoretical and systematic method to handle it. In [LLL21], Liu *et al.* invented a method called algebraic transitional form (ATF) to handle the probabilistic DL approximation. Thanks to the new insights into the HD/HDL cryptanalysis we introduced in Section 3, we can extend the ATF technique to a higher-order case, which is named as the higher-order algebraic transitional form (HATF).

5.1 Algebraic Transitional Form

In this subsection, we briefly recall the basic ideas of the ATF including how to construct the ATF of an output bit of a cipher and how to estimate the DL bias from the ATF. We remind that since our HATF technique is a general form of the ATF, readers are safe not to understand all details of the ATF in this subsection. Indeed, when the order $\ell = 1$, the HATF is just the same as the ATF.

Basic idea of algebraic transitional forms. Equation (3) tells us that if we can (a) calculate out 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 as a transitional expression of the exact ANF of f_{Δ}, for which it is easier to compute the bias. The ATF of a Boolean function f is denoted by $\mathcal{A}(f)$. From $\mathcal{A}(f_{\Delta})$, we hope to obtain a simple transitional expression of $D_x f_{\Delta}$, say $D_x\mathcal{A}(f_{\Delta})$. Finally, the bias of $D_x\mathcal{A}(f_{\Delta})$ will be regarded as an estimation of the real bias.

Construction of algebraic transitional forms. The core of the ATF technique is to substitute some parts of a Boolean function with new variables to simplify its form. Since almost all symmetric-key primitives are iterated designs, each of their output bits can be represented as a composite Boolean function such as

$$f = f^{r-1} \circ f^{r-2} \circ \cdots \circ f^0,$$

where $f^i : \mathbb{F}_2^n \to \mathbb{F}_2^n$ for $0 \leq i < r-1$ and $f^{r-1} : \mathbb{F}_2^n \to \mathbb{F}_2$. Since we want to construct a transitional expression for $D_x f_\Delta$, we need to be careful not to bury the variable x during the substitution operations. Therefore, we only substitute the expressions in f which are independent of x. Following this principle, we introduce the transitional variables $\alpha^i, \beta^i \in \mathbb{F}_2^n$ for $F^i = f^i \circ \cdots \circ f^0(X \oplus x\Delta)$ for $0 \leq i < r$. For the j-th bit of the output of F^i, which can be written as $F^i[j] = (F^i[j])''x + (F^i[j])'$, we let

$$\begin{cases} \alpha^i[j] \overset{s}{=}_Q (F^i[j])'' \\ \beta^i[j] \overset{s}{=}_Q (F^i[j])' \end{cases}, 0 \leq j < n$$

where "$\overset{s}{=}_Q$" means that we substitute $(F^i[j])''$ and $(F^i[j])'$ with new variables $\alpha^i[j]$ and $\beta^i[j]$, respectively, and store the key-value pairs $\{\alpha^i[j] : (F^i[j])''\}$ and $\{\beta^i[j] : (F^i[j])'\}$ into a substitution dictionary Q, for all j. Since the goal of the substitution is to simplify the ANF of f, we apply it only when $(F^i[j])''$ or $(F^i[j])'$ contains at least two different variables (no need to enforce substitutions if the expressions are simple enough).

After the substitution, the ANF of each coordinate of F^i is simplified to $F^i[j] = \alpha^i[j]x \oplus \beta^i[j]$. For readability, we ignore their indexes and write them as $F^i = \alpha^i x \oplus \beta^i$, for all $0 \leq j < n$, and this is called the *ATF of* F^i, denoted by $\mathcal{A}(F^i)$. From $\mathcal{A}(F^i)$, we calculate $\mathcal{A}(F^{i+1})$ similarly. Finally, $\mathcal{A}(f)$ can be computed from $\mathcal{A}(F^{r-2})$ as $f = f^{r-1}(\alpha^{r-2}x \oplus \beta^{r-2})$. The algorithm for constructing $\mathcal{A}(f_\Delta)$ is given in [LLL21, Algorithm 1].

Evaluating the bias of $\mathcal{A}(f_\Delta)$. Since we have obtained $\mathcal{A}(f_\Delta)$, we can calculate $D_x\mathcal{A}(f_\Delta)$ (we can do this because we do substitutions for expressions independent of x) as an transitional expression of $D_x f_\Delta$. In [LLL21], the bias of $D_x f_\Delta$ is estimated from $D_x\mathcal{A}(f_\Delta)$. Suppose $D_x\mathcal{A}(f_\Delta) = p_n \oplus p_l$ where p_l is the XOR of linearly isolated monomials of $D_x\mathcal{A}(f_\Delta)$ and p_n is the remaining part. The bias of p_n, denoted by $\texttt{Bias}(p_n)$, is calculated directly from the definition of the bias by counting the number of inputs leading to a zero output although there may be transitional variables in p_n. If p_l contains any transitional variables, we expand it with the corresponding expressions in Q. We repeat the estimation for this new expression until there are no transitional variables in the linearly isolated part. The biases obtained along the way are used with the piling-up lemma to calculate the bias of $D_x\mathcal{A}(f_\Delta)$. The algorithm for computing this bias is given in [LLL21, Algorithm 2]. Since part of our work directly utilizes this method to estimate the bias of the HDL approximation, to make this paper self-contained we borrow this algorithm and present it in Algorithm 4 in Section A of Supplementary Material.

Precision of the ATF. The ATF technique has achieved better performances for Ascon, Serpent and Grain v1 than previous methods. However, the ATF technique is actually a heuristic technique that relies on some assumptions such as the independence of transitional variables. The validity of these assumptions will need time to be tested on more primitives. To alleviate the worries about the precision of the ATF, Liu *et al.* have performed experiments to verify the biases they obtained, up to computational feasibility. Our HATF technique is a generalization of the ATF to the higher-order case, so it suffers similar problems of precision. Thus we also need to provide experimental verifications for our theoretical results as long as it is possible.

5.2 Higher-Order ATF Technique

According to Definition 1, to evaluate the bias of a HDL approximation of a Boolean function f, we need to evaluate the bias of the coefficient of the max-term of its DSF. Recall that the ATF technique [LLL21] gives a transitional expression of $\mathsf{Coe}\left(f_{\Delta}, x\right)$ for the 1st order DL. We can adapt the ATF technique to the ℓ-th order situation, *i.e.*, we try to construct a transitional expression for $\mathsf{Coe}\left(\mathrm{DSF}_{f,X,\boldsymbol{\Delta}}(\boldsymbol{x}), \pi(\boldsymbol{x})\right)$. After that, we can estimate the bias of the transitional expression using Algorithm 4.

Constructing the HATF of a composite Boolean function. Consider a composite Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ represented as

$$f = f^{r-1} \circ f^{r-2} \circ \cdots \circ f^0, f^i : \mathbb{F}_2^n \to \mathbb{F}_2^n, 0 \le i < r-1, f^{r-1} : \mathbb{F}_2^n \to \mathbb{F}_2.$$

Since we need a transitional expression of $\mathsf{Coe}\left(f, \pi(\boldsymbol{x})\right)$, we need to retain the variables in $\boldsymbol{x}$. Therefore, we introduce 2^ℓ transitional variables, *i.e.*, $\boldsymbol{\alpha}^i = (\alpha_0^i, \alpha_1^i, \ldots, \alpha_{2^l-1}^i) \in (\mathbb{F}_2^n)^{2^\ell}$, to substitute all the coefficients of monomials $\pi_u(\boldsymbol{x}) = \prod_{0\le i<l} x_i^{u[i]}$ for $u \in \mathbb{F}_2^\ell$ in $F^i = f^i \circ f^{i-1} \circ \cdots \circ f^0(\boldsymbol{x})$ as follows:
The ANF of the j-th bit of the output of F^i can be written as

$$F^i[j] = \bigoplus_{u\in\mathbb{F}_2^l} \mathsf{Coe}\left(F^i[j], \pi_u(\boldsymbol{x})\right) \pi_u(\boldsymbol{x}),$$

We use the transitional variable a_u^i to substitute the coefficient of the monomial $\pi_u(\boldsymbol{x})$ as follows,

$$\alpha_u^i[j] \overset{s}{=}_Q \mathsf{Coe}\left(F^i[j], \pi_u(\boldsymbol{x})\right), 1 \le j \le n$$

Again, "$\overset{s}{=}_Q$" means we use a new variable to substitute an expression, and store the key-value pair into a dictionary Q. After that, the HATF of $F^i[j]$ is

$$F^i[j] = \bigoplus_{u\in\mathbb{F}_2^l} \alpha_u^i[j]\ \pi_u(\boldsymbol{x}).$$

Similarly to the ATF, we do the variable substitution only when the number of variables in $\mathsf{Coe}\left(F^i[j], \pi_u(\boldsymbol{x})\right)$ is at least 2 (when $\mathsf{Coe}\left(F^i[j], \pi_u(\boldsymbol{x})\right)$ contains only one variable, it is simple enough and there is no need to introduce new transitional variables to simplify it). For readability, we ignore their indexes and write them as $F^i = \bigoplus_{u\in\mathbb{F}_2^l} \alpha_u^i \pi_u(\boldsymbol{x})$, for all $0 \le j < n$, and this is called the *higher-order ATF (HATF) of* F^i. We also denote the HATF of f by $\mathcal{A}(f)$ since the ATF [LLL21] is only a special case of the HATF when $\ell = 1$. From $\mathcal{A}(F^i)$,

Algorithm 2 Higher-Order Algebraic Transitional From (HATF)

```
Input: An $\ell$-variable composite Boolean function $f = f^{r-1} \circ f^{r-2} \circ \cdots \circ f^0$
Output: Expression of $\mathcal{A}(f)$ and the variable-substitution dictionary $Q$
1: Initialize the variable-substitution dictionary $Q = \emptyset$
2: Compute $Y^1 = f^0(\boldsymbol{x})$ according to the ANF of $f^0$
3: for $i$ from 1 to $r-1$ do
4:     $\alpha_u^{i-1} \overset{s}{=}_Q \mathsf{Coe}\left(Y^i, \pi_u(\boldsymbol{x})\right)$, for $u \in \mathbb{F}_2^l$ ▷ Substitution and add the key-value pair $\{\alpha_u^{i-1} : \mathsf{Coe}\left(Y^i, \pi_u(\boldsymbol{x})\right)\}$ into $Q$
5:     Compute the HATF of the next round, $Y^{i+1} = f^i\left(\bigoplus_{u \in \mathbb{F}_2^n} \alpha_u^{i-1}\pi_u(\boldsymbol{x})\right)$
6: return $\mathcal{A}(f) = Y^r, Q$
```

we calculate $\mathcal{A}(F^{i+1})$ similarly. Finally, $\mathcal{A}(f)$ can be computed from $\mathcal{A}(F^{r-2})$ as

$$f = f^{r-1}\left(\bigoplus_{u \in \mathbb{F}_2^l} \alpha_u^{r-2}\ \pi_u(\boldsymbol{x})\right).$$

The process of evaluating $\mathcal{A}(f)$ is illustrated in Algorithm 2, which can be seen as a generalized version of [LLL21, Algorithm 1] to the case of higher-order.

In HDL cryptanalysis, we apply Algorithm 2 to $\mathrm{DSF}_{f,X,\boldsymbol{\Delta}}$ to get $\mathcal{A}(f(X \oplus \boldsymbol{x\Delta}))$. After that, we compute $D_{\boldsymbol{x}}\mathcal{A}(f(X \oplus \boldsymbol{x\Delta}))$ as a transitional expression of the HDL expression of f with respect to $\boldsymbol{\Delta}$. The bias of $D_{\boldsymbol{x}}\mathcal{A}(f(X \oplus \boldsymbol{x\Delta}))$ is evaluated by Algorithm 4 which is the same as the ATF technique.

5.3 Application to 5-Round Ascon Initialization

The first of our applications of the HATF is to the 5-round initialization of Ascon. The DL attack based on the ATF technique costs 2^{26} data and time complexities [LLL21]. In this subsection, we show that the HAFT technique can recover the keys 8 times faster than the DL attack.

To apply Algorithm 2, we need to decompose the 5-round Ascon initialization into several small parts. In theory, the more functions are contained in each decomposition part, the higher will be the precision gain. However, this also means heavier or even infeasible computations since the number of transitional variables would increase sharply. In this paper, we take the same method to cut the Ascon functions as the ATF [LLL21].

We divide the Sbox of Ascon into two parts, p_{S_L} and p_{S_N}, as done in [LLL21]. 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 2, we let $f^0 = p_A$, and $f^i = p_A \circ p_B$ for $1 \le i < r-1$, and $f^{r-1} = p_{S_N}$. Thus r-round Ascon (note that we ignore the last diffusion layer)

Algorithm 3 Evaluate the Bias of Conditional HDL Approximations for A Boolean Function

```
Input: An $\ell$-variable composite Boolean function $f = f^{r-1} \circ f^{r-2} \circ \cdots \circ f^0$, a round $r_0$ before which we impose conditions
Output: Expression of $\mathcal{A}(f)$, the variable-substitution dictionary $Q$ and a set conditions $Q_I$
1: Initialize the variable-substitution dictionary $Q = \emptyset$
2: Compute $Y^1 = f^0(\boldsymbol{x})$ according to the ANF of $f^0$
3: for $i$ from 1 to $r-1$ do
4:     if $i \leq r_0$ then
5:         for $u \in \mathbb{F}_2^l$ do                ▷ For coefficient of any $\pi_u(\boldsymbol{x}), u \neq 0$
6:             if $\mathsf{Coe}\left(Y^i, \pi_u(\boldsymbol{x})\right) \notin \{0,1\}$ then
7:                 Add $\mathsf{Coe}\left(Y^i, \pi_u(\boldsymbol{x})\right)$ to $I$
8:                 $Y^i = Y^i \bmod I$
9:     $\alpha_u^{i-1} \overset{s}{=}_Q \mathsf{Coe}\left(Y^i, \pi_u(\boldsymbol{x})\right)$, for $u \in \mathbb{F}_2^l$ ▷ Substitution and add the key-value pair $\{\alpha_u^{i-1} : \mathsf{Coe}\left(Y^{i-1}, \pi_u(\boldsymbol{x})\right)\}$ into $Q$
10:    Compute the HATF of the next round, $Y^{i+1} = f^i\left(\bigoplus_{u \in \mathbb{F}_2^n} \alpha_u^{i-1} \pi_u(\boldsymbol{x})\right)$
11: Dealing with $I$ and obtain a set of expressions in input bits, denoted by $Q_I$
12: return $\mathcal{A}(f) = Y^{i+1}, Q, Q_I$
```

is represented as

$$p_{S_N} \circ (p_A \circ p_B)^{r-2} \circ p_A = f^{r-1} \circ f^{r-1} \circ f^{r-2} \circ \cdots \circ f^1 \circ f^0$$

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

To improve the bias of the HDL approximation, we could impose some conditions I to the first r_0 rounds. The basic principle is to delay the appearance of the maxterm $\pi(\boldsymbol{x})$ in the DSF or reduce the degree of $\mathsf{Coe}\left(\mathrm{DSF}_{f,X,\boldsymbol{\Delta}}, \pi(\boldsymbol{x})\right)$. Thus, in each computation of the ANFs or ATFs in the first r_0 rounds we will extract the coefficients of $\mathsf{Coe}\left(\mathrm{DSF}_{f,X,\boldsymbol{\Delta}}, \pi_u(\boldsymbol{x})\right), u \succ \mathbf{0}$ (when the coefficients are not constants) and put them into a set I as ideal generators. Next, we reduce the polynomials over the ideal generated from I, denoted by "mod I". With the conditions in I, we obtain a set of expressions Q_I by substituting the transitional variables with the original expressions with the help of the dictionary Q. After that, a system of equations $S = \{f = 0 | f \in Q_I\}$ is derived[7], *i.e.*, we will get a HDL distinguisher with a specific bias ε when the equations in S are satisfied. A similar technique has also been used in [LLL21]. The procedure is illustrated in Algorithm 3.

Conditional 2nd order HDL distinguishers. Considering the efficiency, we only search for 2nd order input differences like $\boldsymbol{\Delta} = (\Delta_0, \Delta_1)$ and unit output mask λ in this section. The previous DL attacks [DEMS15,LLL21] have shown that when the input difference is active simultaneously in both the third and

[7] A more detailed discussion of how to handle these conditions can be found in Section D of Supplementary Material.

fourth words, the bias of the output difference tends to be higher. Therefore, we restrict $\varDelta_0$ and $\varDelta_1$ to be active in the third and fourth words of different Sboxes.

With an exhaustive search using Algorithm 3 with $r_0 = 2$ (we choose $r_0 = 2$ for a balance of the high bias and simple conditions) for all the possible positions (i_0, i_1) for input and positions for output bits, there are many combinations of the input difference and output mask $(\varDelta_0, \varDelta_1, \lambda)$ leading to high biases. When $i_0 = 0$, we found four combinations whose biases are 2^{-2}:

1. $\varDelta_0$ is active in $(S^0[3][0], S^0[4][0])$, $\varDelta_1$ is active in $(S^0[3][7], S^0[4][7])$ and λ is active in $S^{4.5}[0][25]$;
2. $\varDelta_0$ is active in $(S^0[3][0], S^0[4][0])$, $\varDelta_1$ is active in $(S^0[3][14], S^0[4][14])$ and λ is active in $S^{4.5}[0][51]$;
3. $\varDelta_0$ is active in $(S^0[3][0], S^0[4][0])$, $\varDelta_1$ is active in $(S^0[3][51], S^0[4][51])$ and λ is active in $S^{4.5}[0][18]$;
4. $\varDelta_0$ is active in $(S^0[3][0], S^0[4][0])$, $\varDelta_1$ is active in $(S^0[3][57], S^0[4][57])$ and λ is active in $S^{4.5}[0][18]$;

We used 2^{26} random data to test each of these biases and the experimental results are respectively $2^{-2.0}$, $2^{-2.0}$, $2^{-2.0}$ and $2^{-2.0}$, which shows the effectiveness and precision of the HATF technique. The concrete conditions and experimental verifications are provided in Section E of Supplementary Material and Tables 6,7,8, and 9. The verification algorithms are also provided in our anonymous git repository.

Recovering key bits from the conditions. We take the first case above as an example to describe our key-recovery attack on 5-round Ascon AEAD. From Algorithm 3 and our experiments, the bias is $2^{-2.0}$ when 18 conditions in Q_I are satisfied. Since some conditions are related to secret key bits, we could observe the bias of the HDL distinguisher to guess some key bits. These conditions can be categorized into three types as introduced in [LM12] (for simplicity, we use $u_0, \ldots, u_{127}$ to represent the 128 bits of nonce $S^0[3][0], \ldots, S^0[4][63]$ and $k_0, \ldots, k_{127}$ to represent the 128 bits of key $S^0[1][0], \ldots, S^0[2][63]$):

- 2 Type-0 conditions involving only nonce bits: $u_0 = u_{64}, u_7 = u_{71}$.
- 12 Type-1 conditions involving bits of nonce and key. We performed some measurements for all 2^{12} cases of the 12 conditions with 2^{22} samples each, and found that 9 conditions seem to be largely redundant: whether they hold or not does not affect the bias significantly. To optimize the data and time complexity, we remove these conditions and retain only the three most significant ones.

$$\begin{aligned}
u_{16} &= u_{19} \oplus u_{49} \oplus u_{80} \oplus u_{83}k_{19} \oplus u_{83} \oplus u_{90}k_{26} \oplus u_{113}k_{49} \oplus u_{113} \oplus k_9 \oplus k_{16} \\
&\quad \oplus k_{19} \oplus k_{49} \oplus k_{73} \oplus k_{80} \oplus k_{90} \\
u_{67} &= u_3k_3 \oplus u_3k_{67} \oplus u_3 \oplus u_{25}k_{25} \oplus u_{25}k_{89} \oplus u_{25} \oplus u_{89} \oplus k_3k_{67} \oplus k_3 \\
&\quad \oplus k_{25}k_{89} \oplus k_{25} \oplus k_{67} \oplus k_{89} \oplus 1 \\
u_{74} &= u_{10}k_{10} \oplus u_{10}k_{74} \oplus u_{10} \oplus u_{32}k_{32} \oplus u_{32}k_{96} \oplus u_{32} \oplus u_{96} \oplus k_{10}k_{74} \oplus k_{10} \\
&\quad \oplus k_{32}k_{96} \oplus k_{32} \oplus k_{74} \oplus k_{96}
\end{aligned}$$

– 4 Type-2 conditions involving only bits of key: $k_0 = 0, k_{64} = 0, k_7 = 0, k_{71} = 0$.

Considering the removal of the 9 conditions, the overall bias changes a bit. By experimenting with 2^{26} data, we observe that when all 9 above conditions are satisfied the bias would be $2^{-3.19}$. When at least one does not hold, the bias is at most $2^{-4.47}$.

The 2 Type-0 conditions can be satisfied for free. However, we cannot control the type-2 conditions. To continue, let's first assume the Type-2 conditions have been satisfied, we then only need to distinguish the right case where all these 3 Type-1 conditions are satisfied from the other 7 wrong cases when (u_{16}, u_{67}, u_{74}) varies over all possible values.

To distinguish the right case from the wrong cases, we perform a statistical test. Suppose we encrypt N samples, the frequency of the parity bit being 0, denoted by T, obeys the binary distribution $\mathcal{B}(N, \frac{1}{2} + \varepsilon)$, where ε is the bias of the parity. According to the law of large numbers, T obeys approximately a normal distribution

$$T \sim \mathcal{N}\left(N(\frac{1}{2} + \varepsilon), N(\frac{1}{4} - \varepsilon^2)\right) \tag{6}$$

Since the right and wrong cases lead to different biases, T of the right and wrong cases follow different normal distributions. Distinguishing the right case from wrong cases is related to distinguishing two different normal distributions. This question has actually been studied extensively in linear-like attacks. We adapt it for the DL or HDL cases, finally, $2^{9.94}$ samples are enough to identify the right case. The theory and concrete calculation are provided in Section J of Supplementary Material.

Recover more key bits. Once we find the right case, we can know more information from the conditions by flipping some nonce bits. We take an example to demonstrate how to get these equations. Note that in the first Type-1 condition, the coefficient of u_{83} is $k_{19} \oplus 1$. Then, if know a set of nonce values that already satisfies all Type-1 conditions, we can flip u_{83} to see whether the conditions are still satisfied by the aforementioned statistical testing. If the conditions still hold, we know $k_{19} \oplus 1 = 0$ since the flipping of u_{83} does not change anything, otherwise $k_{19} \oplus 1 = 1$. With this strategy, we can get several key equations, that is, $k_{19} = c_0, k_{26} = c_1, k_{49} = c_2, k_3 \oplus k_{67} = c_3, k_{25} \oplus k_{89} = c_4, k_{10} \oplus k_{74} = c_5, k_{32} \oplus k_{96} = c_6$, after knowing these key values, the value of the following three key equations are naturally known: $k_9 \oplus k_{16} \oplus k_{19} \oplus k_{49} \oplus k_{73} \oplus k_{80} \oplus k_{90} = c_7, k_3k_{67} \oplus k_3 \oplus k_{25}k_{89} \oplus k_{25} \oplus k_{67} \oplus k_{89} = c_8, k_{10}k_{74} \oplus k_{10} \oplus k_{32}k_{96} \oplus k_{32} \oplus k_{74} \oplus k_{96} = c_9$. Note that the two quadratic equations above can be easily linearized, *e.g.*, since we already know $k_{25} \oplus k_{89} = c_5$, $k_{25}k_{89} = k_{25}(c_5 \oplus k_{89})$. As a result, we can recover 10 key bits from Type-1 conditions and 4 from Type-2 conditions, respectively.

Until now our cryptanalysis assumes that the four Type-2 conditions have been satisfied. Practically, due to the rotation-variance property[8] of the Ascon state, we actually have 64 opportunities to find the satisfied Type-2 conditions. To calculate the number of key bits that might be recovered, we randomly generate 2^{20} keys and find on average 4 opportunities with the Type-2 conditions holding. However, due to some overlap of different positions, the average key bits we can recover from Type-2 conditions are expected as 8 when we try 64 positions. Thus, the expectation is that we can gain $10 \times 4(\text{Type-1}) + 8(\text{Type-2}) = 48$ equations of key bits if we only use the first HDL approximation.

The example we exhibited above requires 2^3 times of statistical test to identify the right case, each statistical needs encrypting $2^{9.94}$ samples. After identifying out the right case, we flip 7 nonce bits to recover the 10-bit key information. Thus the above example costs $4(\text{each sample contains 4 texts}) \times (64(\text{positions}) \times (2^{3+9.94} + 4(\text{opportunities}) \times 7(\text{flips}) \times 2^{9.94}) = 2^{21.0}$ encryptions.

Note that we have another 3 HDL approximations with bias equal to 2^{-2}. Similar analyses for the remaining 3 cases are provided in Section F of Supplementary Material. Together with a strategy to utilize all Type-1 conditions that is provided in Section F.4 of Supplementary Material, we can gain more equations about the key bits without increasing the complexities. Finally, we can gain significantly more than 105 linearly independent equations of key bits. Thus, 4×2^{21} data and encryptions are enough to recover all 128 key bits, which is 8 times faster than the DL attack [LLL21].

HDL cryptanalysis of the 6-round Ascon initialization. With the HATF, we detected two HDL approximations with bias equal to -2^{-30} and two -2^{-37} for 6-round Ascon. In [LLL21], Liu *et al.* remarked that they made a lot of efforts but could not find any DL approximation with bias larger than 2^{-64}, which demonstrate well the advantage of the HDL cryptanalysis. Based on these four HDL approximations, we can mount a key-recovery attack on 6-round Ascon and we provide the details of this attack in Section G of Supplementary Material.

Conditional HD approximation of 130-round Grain v1. In [LLL21], Liu *et al.* found a 125-round conditional differential with bias $2^{-20.77}$ (experimentally $2^{-17.4}$) on Grain v1. To further compare the effects of differential and HD, we used HATF and detected a conditional HD with bias equal to $2^{-30.18}$ for 130-round Grain v1, which is five rounds longer than the conditional differential counterpart. More details are presented in Section H of Supplementary Material.

6 Conclusion

In this paper, we revisited the HD/HDL cryptanalysis from an algebraic perspective: the HD/HDL approximation is equivalent to that of the superpoly of the maxterm in the DSF. Our work provides better insights on the HD and HDL cryptanalysis, which makes this attack proposed at 2015 much easier to

[8] Strictly speaking, Ascon is not rotation-variant because of the IV and p_C, however, our experiments showed that the IV and p_C have little influence to the HDL properties in our cases.

use now. By analyzing the DSF, we provided three methods to detect possible HD/HDL distinguishers. The first one is to estimate an upper bound on the algebraic degree of the DSF. Since the DSF is parameterized by the input value and difference, we can choose some specific values to simplify it and obtain useful HDL distinguishers more easily. The second method is called HATF by which we construct a transitional expression of the DSF and then use it to estimate the bias of the HDL approximation. The third method is based on the cube tester to experimentally obtain some useful practical HDL distinguishers. By these new methods, we greatly improved the best distinguishing attacks on the Ascon permutation and key-recovery attacks on 5 rounds of the Ascon initialization. We also obtained better approximations for some other high-profile primitives such as Grain v1, Xoodoo and ChaCha. We believe that the HDL cryptanalysis has more potential than expected, and deserves more attention from the cryptographic community.

Acknowledgments. We are grateful to the anonymous referees for their comments that improved the quality of this article.

References

ADMS09. Jean-Philippe Aumasson, Itai Dinur, Willi Meier, and Adi Shamir. Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium. In Orr Dunkelman, editor, *Fast Software Encryption - FSE 2009*, volume 5665 of *LNCS*, pages 1–22. Springer, 2009.

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.

BCD11. Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-Order Differential Properties of Keccak and *Luffa*. In Antoine Joux, editor, *Fast Software Encryption - FSE 2011*, volume 6733 of *LNCS*, pages 252–269. Springer, 2011.

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.

Ber08a. Daniel J Bernstein. ChaCha, a variant of Salsa20. In *Workshop record of SASC*, volume 8, pages 3–5, 2008.

Ber08b. Daniel J. Bernstein. The Salsa20 Family of Stream Ciphers. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *LNCS*, pages 84–97. Springer, 2008.

BJK$^+$16. Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016*, volume 9815 of *LNCS*, pages 123–153. Springer, 2016.

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.

BLT20. Christof Beierle, Gregor Leander, and Yosuke Todo. Improved Differential-Linear Attacks with Applications to ARX Ciphers. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020*, volume 12172 of *LNCS*, pages 329–358. Springer, 2020.

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.

CM16. Arka Rai Choudhuri and Subhamoy Maitra. Significantly Improved Multi-bit Differentials for Reduced Round Salsa and ChaCha. *IACR Trans. Symmetric Cryptol.*, 2016(2):261–287, 2016.

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.

DHAK18. Joan Daemen, Seth Hoffert, Gilles Van Assche, and Ronny Van Keer. Xoodoo cookbook. *IACR Cryptol. ePrint Arch.*, page 767, 2018.

DR02. Joan Daemen and Vincent Rijmen. AES and the Wide Trail Design Strategy. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 108–109. Springer, 2002.

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.

HJM07. Martin Hell, Thomas Johansson, and Willi Meier. Grain: a stream cipher for constrained environments. *Int. J. Wirel. Mob. Comput.*, 2(1):86–93, 2007.

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.

KW02. Lars R. Knudsen and David A. Wagner. Integral Cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption - FSE 2002*, volume 2365 of *LNCS*, pages 112–127. Springer, 2002.

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

LCM$^+$16. Adam Langley, Wan-Teh Chang, Nikos Mavrogiannopoulos, Joachim Strombergson, and Simon Josefsson. ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS). RFC 7905, June 2016.

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

LG19. Jun-Zhi Li and Jie Guan. Advanced conditional differential attack on Grain-like stream cipher and application on grain v1. *IET Inf. Secur.*, 13(2):141–148, 2019.

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.

LM12. Michael Lehmann and Willi Meier. Conditional Differential Cryptanalysis of Grain-128a. In Josef Pieprzyk, Ahmad-Reza Sadeghi, and Mark Manulis, editors, *Cryptology and Network Security - CANS*, volume 7712, pages 1–11. Springer, 2012.

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.

LZWW17. Yanbin Li, Guoyan Zhang, Wei Wang, and Meiqin Wang. Cryptanalysis of round-reduced ASCON. *Sci. China Inf. Sci.*, 60(3):38102, 2017.

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.

RHSS21. Raghvendra Rohit, Kai Hu, Sumanta Sarkar, and Siwei Sun. Misuse-Free Key-Recovery and Distinguishing Attacks on 7-Round Ascon. *IACR Trans. Symmetric Cryptol.*, 2021(1):130–155, 2021.

RS21. Raghvendra Rohit and Santanu Sarkar. Diving deep into the weak keys of round reduced ascon. *IACR Trans. Symmetric Cryptol.*, 2021(4):74–99, 2021.

SZFW12. Zhenqing Shi, Bin Zhang, Dengguo Feng, and Wenling Wu. Improved Key Recovery Attacks on Reduced-Round Salsa20 and ChaCha. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *Information Security and Cryptology - ICISC 2012*, volume 7839 of *LNCS*, pages 337–351. Springer, 2012.

Tez16. Cihangir Tezcan. Truncated, Impossible, and Improbable Differential Analysis of Ascon. *IACR Cryptol. ePrint Arch.*, page 490, 2016.

Tie16. Tyge Tiessen. Polytopic Cryptanalysis. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016*, volume 9665 of *LNCS*, pages 214–239. Springer, 2016.

Tie17. Tyge Tiessen. From Higher-Order Differentials to Polytopic Cryptanalysis. *IACR Cryptol. ePrint Arch.*, page 266, 2017.

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.

Supplementary Material

A Algorithm for Evaluating the Bias of an ATF

Algorithm 4 Estimation of the Differential-Linear Bias [LLL21]

```
Input: The ATF $\mathcal{A}(f)$ of a Boolean function $f$, and the substitution dictionary $Q$
Output: A bias $\varepsilon$
 1: Calculate $e = D_{\boldsymbol{x}}\mathcal{A}(f)$, set $\varepsilon = \frac{1}{2}$
 2: while $e \neq 0$ do
 3:     Selected the isolated variables in $e$, and sum them to $e_l$
 4:     Compute the bias of $e^\star = e - e_l$ by $\varepsilon^\star = \texttt{Bias}(e^\star)$, and calculate $\varepsilon = 2 \cdot \varepsilon^\star \cdot \varepsilon$
 5:     Substitute the expressions $Q$ into $e_l$, and update $e$ with this new polynomial
            ▷ For some complicated case such as Grain v1, we will substitute only
        one monomial in $e_l$ every time. Then, the final bias of $e_l$ will be estimated by the
        piling lemma with bias of all monomials in $e_l$
 6: return $\varepsilon$
 7: procedure Bias ($f$)
 8:     $(f_1, f_2, \dots, f_{m-1}) \leftarrow \texttt{Separate}(f)$
 9:     $\varepsilon \leftarrow \frac{1}{2}$
10:     for $i$ from 1 to $m$ do
11:         if the number of variables in the expression of $f_i$ is small then
12:             Compute the bias $\varepsilon_i$ of $f_i$ according to its Hamming weight
13:         else
14:             Select a variable $v$ minimizing the maximum cardinality of the variable
        sets of the polynomial in $\texttt{Separate}(f_i|_{v=0})$ and $\texttt{Separate}(f_i|_{v=1})$
15:             Compute the bias of $f_i$ by $\varepsilon_i = \frac{1}{2}\texttt{Bias}(f_i|_{v=0}) + \frac{1}{2}\texttt{Bias}(f_i|_{v=1})$
16:         $\varepsilon \leftarrow 2 \cdot \varepsilon \cdot \varepsilon_i$
17:         if $\varepsilon = 0$ then
18:             break
19:     return $\varepsilon$
20: procedure Separate($f$)
21:     Separate the Boolean polynomial $f$ as a sum of $m$ polynomials $f_i$ whose variable
        sets are mutually disjoint, and sort $f_1, f_2, \dots, f_m$ in ascending order according to
        the number of terms in their ANFs
22:     return $(f_1, f_1, \dots, f_m)$
```

B 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 2. 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 $0 \le i < 5, 0 \le j < 64$. We use $S^{r.5}$ to represent the state after p_S of the r-th round, $r \ge 0$.

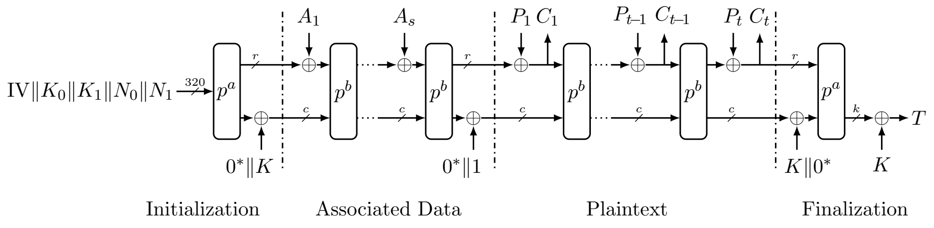


Fig. 2: The encryption algorithm of Ascon

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:

$$\begin{cases} y_0 = x_4x_1 + x_3 + x_2x_1 + x_2 + x_1x_0 + x_1 + x_0 \\ y_1 = x_4 + x_3x_2 + x_3x_1 + x_3 + x_2x_1 + x_2 + x_1 + x_0 \\ y_2 = x_4x_3 + x_4 + x_2 + x_1 + 1 \\ y_3 = x_4x_0 + x_4 + x_3x_0 + x_3 + x_2 + x_1 + x_0 \\ y_4 = x_4x_1 + x_4 + x_3 + x_1x_0 + x_1 \end{cases} \tag{7}$$

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

$$\begin{cases} y_0 = x_4x_3x_2 + x_4x_3x_1 + x_4x_3x_0 + x_3x_2x_0 + x_3x_2 + x_3 + x_2 + x_1x_0 + x_1 + 1 \\ y_1 = x_4x_2x_0 + x_4 + x_3x_2 + x_2x_0 + x_1 + x_0 \\ y_2 = x_4x_3x_1 + x_4x_3 + x_4x_2x_1 + x_4x_2x_0 + x_4x_2 + x_4 + x_3x_2 + x_3x_1x_0 \\ \qquad\qquad +x_3x_1 + x_2x_1x_0 + x_2x_1 + x_2x_0 + x_2 + x_1 + x_0 + 1 \\ y_3 = x_4x_2x_1 + x_4x_2x_0 + x_4x_2 + x_4x_1 + x_4 + x_3 + x_2x_1 + x_2x_0 + x_1 \\ y_4 = x_4x_3x_2 + x_4x_2x_1 + x_4x_2x_0 + x_4x_2 + x_3x_2x_0 + x_3x_2 + x_3 + x_2x_1 + x_2x_0 + x_1x_0 \end{cases} \tag{8}$$

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

$$\begin{cases} y_0 \leftarrow \Sigma_0(x_0) = x_0 + (x_0 \ggg 19) + (x_0 \ggg 28) \\ y_1 \leftarrow \Sigma_1(x_1) = x_1 + (x_1 \ggg 61) + (x_1 \ggg 39) \\ y_2 \leftarrow \Sigma_2(x_2) = x_2 + (x_2 \ggg 1) + (x_2 \ggg 6) \\ y_3 \leftarrow \Sigma_3(x_3) = x_3 + (x_3 \ggg 10) + (x_3 \ggg 17) \\ y_4 \leftarrow \Sigma_4(x_4) = x_4 + (x_4 \ggg 7) + (x_4 \ggg 41) \end{cases} \tag{9}$$

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.1 Proofs for Proposition 2 and 3

C Zero-Sum Distinguishers

We deduce the transitional rules for the inverse operations of the Ascon permutation. Thereby we can give two corollaries of Propositions 2 and 3. According to the ANFs of the inverse Sbox of Ascon as follows,

$$\begin{cases} y_0 = x_4x_3x_2 + x_4x_3x_1 + x_4x_3x_0 + x_3x_2x_0 + x_3x_2 + x_3 + x_2 + x_1x_0 + x_1 + 1 \\ y_1 = x_4x_2x_0 + x_4 + x_3x_2 + x_2x_0 + x_1 + x_0 \\ y_2 = x_4x_3x_1 + x_4x_3 + x_4x_2x_1 + x_4x_2x_0 + x_4x_2 + x_4 + x_3x_2 + x_3x_1x_0 \\ \qquad\qquad +x_3x_1 + x_2x_1x_0 + x_2x_1 + x_2x_0 + x_2 + x_1 + x_0 + 1 \\ y_3 = x_4x_2x_1 + x_4x_2x_0 + x_4x_2 + x_4x_1 + x_4 + x_3 + x_2x_1 + x_2x_0 + x_1 \\ y_4 = x_4x_3x_2 + x_4x_2x_1 + x_4x_2x_0 + x_4x_2 + x_3x_2x_0 + x_3x_2 + x_3 + x_2x_1 + x_2x_0 + x_1x_0 \end{cases} \tag{10}$$

we deduce the following corollary,

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*

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

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* $\mathrm{DM}(p_L^{-1}(S)) = (d''_{i,j}, 0 \leq i < 5, 0 \leq j < 64)$*, where* $d''_{i,j}, 0 \leq i < 5, 0 \leq j < 64$ *are computed as*

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

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 4.2 to the inverse Ascon permutation.

Considering that with $(X, \boldsymbol{\Delta})$ in Equation (5), 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 $p_L^{-1} \circ p_C \circ p_S^{-1} \circ p_L^{-1} \circ p_C(X \oplus \boldsymbol{x}\boldsymbol{\Delta}^T)$, 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 4.

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.

D Handling the Conditions from An Ideal and Similar Ideals

In this section, we discuss more about the conditions we extracted from the coefficients during the computations of the ATFs. The basic ideas behind imposing the conditions are to force the coefficients to be zero to delay the appearance of the maxterm in the ANFs. When we push all related coefficients (excluding constant coefficients) into I, I can be used to calculate the ideal of all the Boolean functions we are dealing with. Thus, heuristically if we force the polynomials in I to zero, we expect that all the coefficients can be reduced to zero and meanwhile all the non-constant monomials will be eliminated. However, since coefficients of some monomials are possible to be constant 1, we cannot eliminate all monomials.

At the same time, given an ideal generated from I, we can change the generators in I by adding a constant 1 to certain generators to derive some *similar ideals*, which is denoted by $\mathcal{S}(I) = \{p + c_p : p \in I, c_p \in \mathbb{F}_2\}$. Considering that the calculations for the ATFs are very complicated, it is possible for the ANFs to be simplified by any ideal in $\mathcal{S}(I)$ and even achieve better performances, *i.e.*, sometimes we gain higher bias when we choose a proper ideal from $\mathcal{S}(I)$ than the original I. We exhibit a small example to show this intuition.

Example 2. Given a composite Boolean function $z = f^1 \circ f^0(x_0, x_1, x_2, x_3, x_4)$ as follows,

$$\begin{aligned} z &= (y_0 \oplus y_1)x_0 \\ y_0 &= x_4x_1 + x_3 + x_2x_1 + x_2 + x_1x_0 + x_1 + x_0 \\ y_1 &= x_4x_0 + x_4 + x_3x_0 + x_3 + x_2 + x_1 + x_0 \end{aligned}$$

We extract the coefficients of x_1 in y_0 and y_1, *i.e.*, $\mathsf{Coe}\,(y_0, x_1) = x_4 + x_2 + x_0 + 1$ and $\mathsf{Coe}\,(y_1, x_1) = 1$, respectively. Since $\mathsf{Coe}\,(y_1, x_1)$ is constant 1, $\mathsf{Coe}\,(y_0, x_1)$ is the only ideal generator, we push it to I. Then we reduce y_0 and y_1 by the ideal of I, we will get

$$\begin{aligned} y_0 \quad \bmod I &= x_3 \oplus x_2 \oplus x_0 \\ y_1 \quad \bmod I &= x_1 + x_2x_3 + x_2x_4 + x_3x_4 + 1 \end{aligned}$$

Thus, when we set the only generator as zero, *i.e.*, $x_4+x_2+x_0+1=0$, y_0 and y_1 will be reduced actually to

$$\begin{aligned}y_0|_{x_4=x_2+x_0+1} &= x_3 \oplus x_2 \oplus x_0\\ y_1|_{x_4=x_2+x_0+1} &= x_0x_2+x_0x_3+x_1+x_3+1\end{aligned}$$

So, $\mathsf{Coe}\left(z,x_1\right)=x_0$. However, if we use its similar ideal, *i.e.*, $x_4+x_2+x_0$, we will obtain

$$\begin{aligned}y_0|_{x_4=x_2+x_0} &= x_0 \oplus x_1 \oplus x_2 \oplus x_3\\ y_1|_{x_4=x_2+x_0} &= x_0x_2 \oplus x_0x_3 \oplus x_0 \oplus x_1 \oplus x_3\end{aligned}$$

Thus, $\mathsf{Coe}\left(z,x_1\right)=0$. The bias of $\mathsf{Coe}\left(z,x_1\right)$ increases.

In our application, it is not easy to test the similar ideals in calculations of ATFs. However, when we obtain an ideal from the computations, we will test all possible similar ideals when we experimentally verify the results. This will be clearer in the next subsection.

E Conditions and Verifications for 5-Round Ascon in Section 5

In Section 5.3, we provided four highly biased HDL approximations as follows,

1. $\varDelta_0$ is active in $(S^0[3][0], S^0[4][0])$, $\varDelta_1$ is active in $(S^0[3][7], S^0[4][7])$ and λ is active in $S^{4.5}[0][25]$, the bias is equal to 2^{-2};
2. $\varDelta_0$ is active in $(S^0[3][0], S^0[4][0])$, $\varDelta_1$ is active in $(S^0[3][14], S^0[4][14])$ and λ is active in $S^{4.5}[0][51]$, the bias is equal to 2^{-2};
3. $\varDelta_0$ is active in $(S^0[3][0], S^0[4][0])$, $\varDelta_1$ is active in $(S^0[3][51], S^0[4][51])$ and λ is active in $S^{4.5}[0][18]$, the bias is equal to 2^{-2};
4. $\varDelta_0$ is active in $(S^0[3][0], S^0[4][0])$, $\varDelta_1$ is active in $(S^0[3][57], S^0[4][57])$ and λ is active in $S^{4.5}[0][18]$, the bias is equal to 2^{-2}.

In the remaining part of this section, we provide their conditions and experimental verification results.

E.1 HDL Approximation 1

Conditions. The Type-0, Type-1 and Type-2 conditions for Case 1 are as follows,

- 2 Type-0 conditions involving only nonce bits: $u_0=u_{64}, u_7=u_{71}$
- 12 Type-1 conditions involving bits of nonce and key.

$$\begin{aligned}u_{95} = {} & u_{28}k_{28} \oplus u_{28}k_{92} \oplus u_{28} \oplus u_{31}k_{31} \oplus u_{31}k_{95} \oplus u_{31} \oplus u_{53}k_{53} \oplus u_{53}k_{117}\\ & \oplus u_{53} \oplus u_{92} \oplus u_{117} \oplus k_{28}k_{92} \oplus k_{28} \oplus k_{31}k_{95} \oplus k_{31} \oplus k_{53}k_{117} \oplus k_{53}\\ & \oplus k_{92} \oplus k_{95} \oplus k_{117}\\ u_{16} = {} & u_{19} \oplus u_{49} \oplus u_{80} \oplus u_{83}k_{19} \oplus u_{83} \oplus u_{90}k_{26} \oplus u_{113}k_{49} \oplus u_{113} \oplus k_9 \oplus k_{16}\end{aligned}$$

$$
\begin{aligned}
&\oplus k_{19} \oplus k_{49} \oplus k_{73} \oplus k_{80} \oplus k_{90} \\
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_{67} &= u_{3}k_{3} \oplus u_{3}k_{67} \oplus u_{3} \oplus u_{25}k_{25} \oplus u_{25}k_{89} \oplus u_{25} \oplus u_{89} \oplus k_{3}k_{67} \oplus k_{3} \\
&\oplus k_{25}k_{89} \oplus k_{25} \oplus k_{67} \oplus k_{89} \oplus 1 \\
u_{74} &= u_{10}k_{10} \oplus u_{10}k_{74} \oplus u_{10} \oplus u_{32}k_{32} \oplus u_{32}k_{96} \oplus u_{32} \oplus u_{96} \oplus k_{10}k_{74} \oplus k_{10} \\
&\oplus k_{32}k_{96} \oplus k_{32} \oplus k_{74} \oplus k_{96} \\
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} \oplus k_{42} \oplus k_{66} \oplus k_{73} \oplus k_{83} \\
u_{93} &= u_{26}k_{26} \oplus u_{26}k_{90} \oplus u_{26} \oplus u_{29}k_{29} \oplus u_{29}k_{93} \oplus u_{29} \oplus u_{51}k_{51} \oplus u_{51}k_{115} \\
&\oplus u_{51} \oplus u_{90} \oplus u_{115} \oplus k_{26}k_{90} \oplus k_{26} \oplus k_{29}k_{93} \oplus k_{29} \oplus k_{51}k_{115} \oplus k_{51} \\
&\oplus k_{90} \oplus k_{93} \oplus k_{115} \oplus 1 \\
u_{102} &= u_{35}k_{35} \oplus u_{35}k_{99} \oplus u_{35} \oplus u_{38}k_{38} \oplus u_{38}k_{102} \oplus u_{38} \oplus u_{60}k_{60} \oplus u_{60}k_{124} \\
&\oplus u_{60} \oplus u_{99} \oplus u_{124} \oplus k_{35}k_{99} \oplus k_{35} \oplus k_{38}k_{102} \oplus k_{38} \oplus k_{60}k_{124} \oplus k_{60} \\
&\oplus k_{99} \oplus k_{102} \oplus k_{124} \\
u_{58} &= u_{18} \oplus u_{25} \oplus u_{28} \oplus u_{82} \oplus u_{89} \oplus u_{92}k_{28} \oplus u_{92} \oplus u_{99}k_{35} \oplus u_{122}k_{58} \\
&\oplus u_{122} \oplus k_{18} \oplus k_{25} \oplus k_{28} \oplus k_{58} \oplus k_{82} \oplus k_{89} \oplus k_{99} \oplus 1 \\
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} \\
u_{30} &= u_{54} \oplus u_{61} \oplus u_{94}k_{30} \oplus u_{94} \oplus u_{118} \oplus u_{125} \oplus k_{54} \oplus k_{61} \oplus k_{118} \oplus k_{125} \oplus 1 \\
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}
$$

– 4 Type-2 conditions involving only bits of key: $k_0 = 0, k_{64} = 0, k_7 = 0, k_{71} = 0$.

Our HATF technique shows that when the above 18 conditions are satisfied, the bias of the HDL approximation will be 2^{-2}. We use 2^{26} randomly chosen data to test the bias and find the actual bias is $2^{2.0}$, which is very precise. What's more, we use the HATF to compute the biases of all the 64 output bits. The theoretical and experimental results are provided in Table 6. From the values in this table, we can conclude that for most cases, the HDL technique is reliable and useful to detect the highly biased approximations. However, for the 36th output bit, the theoretical bias is 2^{-6} while the experimental bias is smaller than 2^{-13}. Thus, the theoretical results from the HATF technique are not always reliable, when it is possible, we should use experiments to verify them.

E.2 HDL Approximation 2

Conditions. The Type-0, Type-1 and Type-2 conditions for Case 1 are as follows,

Table 6: Theoretical and experimental biases with input difference $\varDelta_0, \varDelta_1$ which are active in $(S^0[3][0], S^0[4][0])$ and $(S^0[3][7], S^0[4][7])$. Note since our experiments are done with 2^{26} samples, only those experimental biases that are significantly greater than 2^{-13} are reliable. Those gray values in this table are unreliable.

Bit	0	1	2	3	4	5	6	7	8	9
Theory	$2^{-7.0}$	$2^{-7.0}$	0	$2^{-6.0}$	0	$2^{-4.0}$	0	0	$2^{-6.0}$	0
Expr.	$2^{-7.0}$	$2^{-6.4}$	$2^{-14.1}$	$2^{-5.0}$	$2^{-14.1}$	$2^{-4.0}$	$2^{-8.9}$	$2^{-8.0}$	$2^{-5.0}$	$2^{-13.2}$
Bit	10	11	12	13	14	15	16	17	18	19
Theory	$2^{-11.0}$	$2^{-8.0}$	0	$2^{-7.0}$	0	0	$2^{-8.0}$	$2^{-11.0}$	$2^{-4.0}$	$2^{-6.0}$
Expr.	$2^{-10.0}$	$2^{-8.0}$	$2^{-13.8}$	$2^{-7.0}$	$2^{-6.4}$	$2^{-12.9}$	$2^{-8.0}$	$2^{-9.0}$	$2^{-4.0}$	$2^{-15.2}$
Bit	20	21	22	23	24	25	26	27	28	29
Theory	0	$2^{-7.0}$	0	$2^{-7.0}$	$2^{-7.0}$	$2^{-2.0}$	0	$2^{-7.0}$	$2^{-5.0}$	0
Expr.	$2^{-13.6}$	$2^{-5.4}$	$2^{-7.0}$	$2^{-5.4}$	$2^{-6.0}$	$2^{-2.0}$	$2^{-13.4}$	$2^{-5.0}$	$2^{-4.4}$	$2^{-13.4}$
Bit	30	31	32	33	34	35	36	37	38	39
Theory	0	0	$2^{-7.0}$	$2^{-9.0}$	$2^{-4.0}$	0	$2^{-6.0}$	0	0	0
Expr.	$2^{-14.7}$	$2^{-14.6}$	$2^{-7.0}$	$2^{-8.0}$	$2^{-3.0}$	$2^{-7.0}$	$2^{-14.5}$	$2^{-14.0}$	$2^{-13.5}$	$2^{-13.4}$
Bit	40	41	42	43	44	45	46	47	48	49
Theory	0	$2^{-6.0}$	0	$2^{-6.0}$	$2^{-5.0}$	0	0	$2^{-4.0}$	$2^{-8.0}$	$2^{-8.0}$
Expr.	$2^{-15.3}$	$2^{-6.0}$	$2^{-9.0}$	$2^{-5.0}$	$2^{-5.0}$	$2^{-15.5}$	$2^{-15.7}$	$2^{-4.0}$	$2^{-7.0}$	$2^{-7.0}$
Bit	50	51	52	53	54	55	56	57	58	59
Theory	0	$2^{-4.0}$	0	$2^{-7.0}$	$2^{-4.0}$	$2^{-8.0}$	0	$2^{-4.0}$	$2^{-4.0}$	0
Expr.	$2^{-16.2}$	$2^{-4.0}$	$2^{-13.9}$	$2^{-5.4}$	$2^{-4.0}$	$2^{-7.7}$	$2^{-13.9}$	$2^{-4.0}$	$2^{-2.4}$	$2^{-13.9}$
Bit	60	61	62	63						
Theory	0	$2^{-5.0}$	$2^{-6.0}$	$2^{-9.0}$						
Expr.	$2^{-6.0}$	$2^{-4.4}$	$2^{-6.0}$	$2^{-9.0}$						

– 2 Type-0 conditions involving only nonce bits: $x_0 = x_{64}, x_{14} = x_{78}$
– 12 Type-1 conditions involving bits of nonce and key.

$$\begin{aligned}
u_{81} = {} & u_{17}k_{17} \oplus u_{17}k_{81} \oplus u_{17} \oplus u_{39}k_{39} \oplus u_{39}k_{103} \oplus u_{39} \oplus u_{103} \oplus k_{17}k_{81} \\
& \oplus k_{17} \oplus k_{39}k_{103} \oplus k_{39} \oplus k_{81} \oplus k_{103} \\
u_{100} = {} & u_{33}k_{33} \oplus u_{33}k_{97} \oplus u_{33} \oplus u_{36}k_{36} \oplus u_{36}k_{100} \oplus u_{36} \oplus u_{58}k_{58} \oplus u_{58}k_{122} \\
& \oplus u_{97} \oplus u_{122} \oplus k_{33}k_{97} \oplus k_{33} \oplus k_{36}k_{100} \oplus k_{36} \oplus k_{58}k_{122} \oplus k_{97} \oplus k_{100} \\
& \oplus k_{122} \oplus 1 \\
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_{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_{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_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} \oplus k_{42} \oplus k_{66} \oplus k_{73} \oplus k_{83} \\
u_4 = {} & u_7 \oplus u_{37} \oplus u_{61} \oplus u_{68} \oplus u_{71}k_7 \oplus u_{71} \oplus u_{101}k_{37} \oplus u_{101} \oplus u_{125} \oplus k_4 \\
& \oplus k_7 \oplus k_{37} \oplus k_{61} \oplus k_{68} \oplus k_{125} \oplus 1 \\
u_1 = {} & u_{25} \oplus u_{32} \oplus u_{35} \oplus u_{65}k_1 \oplus u_{65} \oplus u_{89} \oplus u_{96} \oplus u_{99}k_{35} \oplus u_{99} \oplus u_{106}k_{42} \\
& \oplus k_1 \oplus k_{25} \oplus k_{32} \oplus k_{35} \oplus k_{89} \oplus k_{96} \oplus k_{106} \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_{16} = {} & u_{23} \oplus u_{26} \oplus u_{56} \oplus u_{80} \oplus u_{87} \oplus u_{90}k_{26} \oplus u_{90} \oplus u_{97}k_{33} \oplus u_{120}k_{56} \\
& \oplus u_{120} \oplus k_{16} \oplus k_{23} \oplus k_{26} \oplus k_{56} \oplus k_{80} \oplus k_{87} \oplus k_{97} \oplus 1 \\
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} \\
u_{109} = {} & u_3k_3 \oplus u_3k_{67} \oplus u_3 \oplus u_{42}k_{42} \oplus u_{42}k_{106} \oplus u_{42} \oplus u_{45}k_{45} \oplus u_{45}k_{109} \\
& \oplus u_{45} \oplus u_{67} \oplus u_{106} \oplus k_3k_{67} \oplus k_3 \oplus k_{42}k_{106} \oplus k_{42} \oplus k_{45}k_{109} \oplus k_{45} \\
& \oplus k_{67} \oplus k_{106} \oplus k_{109} \oplus 1
\end{aligned}$$

– 4 Type-2 conditions involving only bits of key: $k_0 = 0, k_{64} = 0, k_{14} = 0, k_{78} = 0$.

Our HATF technique shows that when the above 18 conditions are satisfied, the bias of the HDL approximation will be 2^{-2}. We use 2^{26} randomly chosen data to test the bias and find the actual bias is also $2^{2.0}$. What's more, we use the HATF to compute the biases of all the 64 output bits. We also provide theoretical and experimental results in Table 7.

Table 7: Theoretical and experimental biases with input difference Δ_0, Δ_1 which are active in $(S^0[3][0], S^0[4][0])$ and $(S^0[3][14], S^0[4][14])$. Note since our experiments are done with 2^{26} samples, only those experimental biases that are significantly greater than 2^{-13} are reliable. Those gray values in this table are unreliable.

Bit	0	1	2	3	4	5	6	7	8	9
Theory	0	$2^{-6.0}$	0	0	0	$2^{-6.0}$	0	0	$2^{-6.0}$	$2^{-5.0}$
Expr.	$2^{-14.1}$	$2^{-5.0}$	$2^{-15.0}$	$2^{-13.9}$	$2^{-14.5}$	$2^{-5.0}$	$2^{-16.0}$	$2^{-14.2}$	$2^{-5.0}$	$2^{-5.0}$
Bit	10	11	12	13	14	15	16	17	18	19
Theory	$2^{-8.0}$	0	$2^{-7.0}$	0	0	0	0	$2^{-8.0}$	$2^{-4.0}$	0
Expr.	$2^{-7.0}$	$2^{-15.9}$	$2^{-14.1}$	$2^{-13.6}$	$2^{-14.0}$	$2^{-13.2}$	$2^{-14.8}$	$2^{-7.0}$	$2^{-3.0}$	$2^{-14.1}$
Bit	20	21	22	23	24	25	26	27	28	29
Theory	$2^{-8.0}$	0	$2^{-5.0}$	0	0	0	0	$2^{-5.0}$	0	0
Expr.	$2^{-7.0}$	$2^{-19.7}$	$2^{-6.0}$	$2^{-15.0}$	$2^{-13.4}$	$2^{-13.0}$	$2^{-13.3}$	$2^{-4.0}$	$2^{-14.7}$	$2^{-13.9}$
Bit	30	31	32	33	34	35	36	37	38	39
Theory	0	0	0	0	$2^{-6.0}$	0	0	0	0	0
Expr.	$2^{-14.3}$	$2^{-22.8}$	$2^{-7.0}$	$2^{-16.0}$	$2^{-6.0}$	$2^{-14.3}$	$2^{-14.6}$	$2^{-5.0}$	$2^{-14.0}$	$2^{-7.0}$
Bit	40	41	42	43	44	45	46	47	48	49
Theory	0	$2^{-5.0}$	0	0	0	0	0	0	0	0
Expr.	$2^{-13.8}$	$2^{-3.8}$	$2^{-13.9}$	$2^{-15.0}$	$2^{-16.0}$	$2^{-13.0}$	$2^{-14.8}$	$2^{-17.2}$	$2^{-12.7}$	$2^{-18.2}$
Bit	50	51	52	53	54	55	56	57	58	59
Theory	$2^{-6.0}$	$2^{-2.0}$	0	0	0	0	$2^{-8.0}$	0	$2^{-7.0}$	0
Expr.	$2^{-6.0}$	$2^{-2.0}$	$2^{-13.4}$	$2^{-16.3}$	$2^{-13.8}$	$2^{-15.2}$	$2^{-8.4}$	$2^{-18.3}$	$2^{-6.4}$	$2^{-13.9}$
Bit	60	61	62	63						
Theory	0	0	$2^{-7.0}$	0						
Expr.	$2^{-15.1}$	$2^{-13.4}$	$2^{-7.0}$	$2^{-16.5}$						

E.3 HDL Approximation 3

Conditions. The Type-0, Type-1 and Type-2 conditions for Case 1 are as follows,

- 2 Type-0 conditions involving only nonce bits: $u_0 = u_{64}, u_{51} = u_{115}$
- 12 Type-1 conditions involving bits of nonce and key.

$$\begin{aligned}
u_{76} &= u_{12}k_{12} \oplus u_{12}k_{76} \oplus u_{12} \oplus u_{54}k_{54} \oplus u_{54}k_{118} \oplus u_{54} \oplus u_{118} \oplus k_{12}k_{76} \\
&\quad \oplus k_{12} \oplus k_{54}k_{118} \oplus k_{54} \oplus k_{76} \oplus k_{118} \oplus 1 \\
u_{41} &= u_{48} \oplus u_{58} \oplus u_{105} \oplus u_{112} \oplus u_{122} \oplus k_{41} \oplus k_{48} \oplus k_{58} \oplus k_{105} \oplus k_{112} \\
&\quad \oplus k_{122} \\
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} \\
&\quad \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} \\
&\quad \oplus k_{92} \oplus k_{95} \oplus k_{117} \\
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} \\
&\quad \oplus k_{12} \oplus k_{42} \oplus k_{66} \oplus k_{73} \oplus k_{83} \\
u_{67} &= u_{3}k_{3} \oplus u_{3}k_{67} \oplus u_{3} \oplus u_{25}k_{25} \oplus u_{25}k_{89} \oplus u_{25} \oplus u_{89} \oplus k_{3}k_{67} \oplus k_{3} \\
&\quad \oplus k_{25}k_{89} \oplus k_{25} \oplus k_{67} \oplus k_{89} \oplus 1 \\
u_{34} &= u_{41} \oplus u_{98} \oplus u_{105} \oplus k_{34} \oplus k_{41} \oplus k_{98} \oplus k_{105}\spadesuit \\
u_{30} &= u_{39} \oplus u_{41} \oplus u_{48} \oplus u_{94}k_{30} \oplus u_{103}k_{39} \oplus u_{105} \oplus u_{112} \oplus u_{122}k_{58} \oplus u_{122} \\
&\quad \oplus k_{30}k_{94} \oplus k_{39}k_{103} \oplus k_{39} \oplus k_{41} \oplus k_{48} \oplus k_{58}k_{122} \oplus k_{58} \oplus k_{94} \oplus k_{103} \\
&\quad \oplus k_{105} \oplus k_{112} \oplus 1 \\
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_{32} &= u_{23} \oplus u_{30} \oplus u_{34} \oplus u_{39} \oplus u_{48} \oplus u_{51} \oplus u_{87}k_{23} \oplus u_{94}k_{30} \oplus u_{96}k_{32} \oplus u_{98} \\
&\quad \oplus u_{103}k_{39} \oplus u_{112} \oplus u_{115} \oplus u_{122}k_{58} \oplus u_{122} \oplus k_{23}k_{87} \oplus k_{23} \oplus k_{30}k_{94} \\
&\quad \oplus k_{32}k_{96} \oplus k_{32} \oplus k_{34} \oplus k_{39}k_{103} \oplus k_{39} \oplus k_{48} \oplus k_{58}k_{122} \oplus k_{58} \oplus k_{87} \\
&\quad \oplus k_{94} \oplus k_{96} \oplus k_{98} \oplus k_{103} \oplus k_{112} \oplus 1\spadesuit \\
u_{125} &= u_{19}k_{19} \oplus u_{19}k_{83} \oplus u_{19} \oplus u_{58}k_{58} \oplus u_{58}k_{122} \oplus u_{61}k_{61} \oplus u_{61}k_{125} \oplus u_{61} \\
&\quad \oplus u_{83} \oplus u_{122} \oplus k_{19}k_{83} \oplus k_{19} \oplus k_{58}k_{122} \oplus k_{61}k_{125} \oplus k_{61} \oplus k_{83} \oplus k_{122} \\
&\quad \oplus k_{125} \\
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} \\
&\quad \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} \\
&\quad \oplus k_{83} \oplus k_{86} \oplus k_{108} \\
u_{11} &= u_{18} \oplus u_{21} \oplus u_{75} \oplus u_{82} \oplus u_{85}k_{21} \oplus u_{85} \oplus u_{92}k_{28} \oplus u_{115} \oplus k_{11} \oplus k_{18} \\
&\quad \oplus k_{75} \oplus k_{82} \oplus k_{92} \oplus x_{1}
\end{aligned}$$

- 4 Type-2 conditions involving only bits of key: $k_0 = 0, k_{64} = 0, k_{51} = 1, k_{115} = 1$.

Our HATF technique shows that when the above 18 conditions are satisfied, the bias of the HDL approximation will be 2^{-2}. We use 2^{26} randomly chosen data to test the bias and find the actual bias is also $2^{2.0}$. What's more, when we use a *similar ideal* to do the experiments, we find the bias would increase to 2^{-1}, *i.e.*, a deterministic approximation. The similar ideal we choose is to add a constant 1 to the two Type-1 conditions labeled by ♠.

Additionally, we use the HATF to compute the biases of all the 64 output bits. We provide theoretical and experimental results in Table 8.

Table 8: Theoretical and experimental biases with input difference Δ_0, Δ_1 which are active in $(S^0[3][0], S^0[4][0])$ and $(S^0[3][51], S^0[4][51])$. Note since our experiments are done with 2^{26} samples, only those experimental biases that are significantly greater than 2^{-13} are reliable. Those gray values in this table are unreliable.

Bit	0	1	2	3	4	5	6	7	8	9
Theory	$2^{-10.0}$	$2^{-9.0}$	$2^{-10.0}$	0	0	$2^{-6.0}$	$2^{-4.0}$	$2^{-9.0}$	$2^{-7.0}$	$2^{-8.0}$
Expr.	$2^{-19.8}$	$2^{-9.6}$	$2^{-13.9}$	$2^{-16.6}$	$2^{-16.1}$	$2^{-13.0}$	$2^{-4.7}$	$2^{-13.6}$	$2^{-10.9}$	$2^{-14.8}$
Bit	10	11	12	13	14	15	16	17	18	19
Theory	0	0	0	0	$2^{-7.0}$	$2^{-4.0}$	$2^{-7.0}$	0	$2^{-2.0}$	0
Expr.	$2^{-13.7}$	$2^{-8.4}$	$2^{-13.6}$	$2^{-16.5}$	$2^{-6.4}$	$2^{-14.0}$	$2^{-14.1}$	$2^{-14.1}$	$2^{-2.0}$	$2^{-13.9}$
Bit	20	21	22	23	24	25	26	27	28	29
Theory	0	0	$2^{-8.0}$	0	0	0	0	$2^{-5.0}$	0	$2^{-8.0}$
Expr.	$2^{-15.2}$	$2^{-15.5}$	$2^{-14.1}$	$2^{-15.2}$	$2^{-7.0}$	$2^{-15.3}$	$2^{-14.0}$	$2^{-14.9}$	$2^{-16.7}$	$2^{-12.9}$
Bit	30	31	32	33	34	35	36	37	38	39
Theory	0	$2^{-5.0}$	0	0	$2^{-5.0}$	$2^{-13.0}$	$2^{-6.0}$	0	0	$2^{-8.0}$
Expr.	$2^{-12.6}$	$2^{-7.0}$	$2^{-13.4}$	$2^{-15.6}$	$2^{-13.9}$	$2^{-15.5}$	$2^{-6.0}$	$2^{-14.7}$	$2^{-16.2}$	$2^{-14.2}$
Bit	40	41	42	43	44	45	46	47	48	49
Theory	$2^{-7.0}$	$2^{-6.0}$	0	$2^{-6.0}$	0	0	$2^{-5.0}$	$2^{-5.0}$	$2^{-8.0}$	$2^{-7.0}$
Expr.	$2^{-7.0}$	$2^{-14.2}$	$2^{-13.4}$	$2^{-15.3}$	$2^{-15.5}$	$2^{-14.0}$	$2^{-13.2}$	$2^{-5.0}$	$2^{-13.8}$	$2^{-15.2}$
Bit	50	51	52	53	54	55	56	57	58	59
Theory	0	0	$2^{-8.0}$	$2^{-5.0}$	0	$2^{-10.0}$	$2^{-8.0}$	$2^{-8.0}$	0	$2^{-4.0}$
Expr.	$2^{-18.3}$	$2^{-13.8}$	$2^{-17.9}$	$2^{-14.5}$	$2^{-12.9}$	$2^{-14.2}$	$2^{-14.5}$	$2^{-13.7}$	$2^{-13.3}$	$2^{-17.5}$
Bit	60	61	62	63						
Theory	0	$2^{-9.0}$	$2^{-7.0}$	$2^{-7.0}$						
Expr.	$2^{-5.0}$	$2^{-13.5}$	$2^{-16.6}$	$2^{-14.6}$						

E.4 HDL Approximation 4

Conditions. The Type-0, Type-1 and Type-2 conditions for Case 1 are as follows,

– 2 Type-0 conditions involving only nonce bits: $u_0 = u_{64}, u_{57} = u_{121}$

– 12 Type-1 conditions involving bits of nonce and key.

$$\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} \\
&\quad \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} \\
&\quad \oplus k_{92} \oplus k_{95} \oplus k_{117} \\
u_{23} &= u_{47} \oplus u_{54} \oplus u_{87}k_{23} \oplus u_{87} \oplus u_{111} \oplus u_{118} \oplus k_{23} \oplus k_{47} \oplus k_{54} \oplus k_{111} \\
&\quad \oplus k_{118} \\
u_{51} &= u_{11} \oplus u_{18} \oplus u_{21} \oplus u_{75} \oplus u_{82} \oplus u_{85}k_{21} \oplus u_{85} \oplus u_{92}k_{28} \oplus u_{115}k_{51} \\
&\quad \oplus u_{115} \oplus k_{11} \oplus k_{18} \oplus k_{51} \oplus k_{75} \oplus k_{82} \oplus k_{92} \oplus 1 \\
u_{67} &= u_{3}k_{3} \oplus u_{3}k_{67} \oplus u_{3} \oplus u_{25}k_{25} \oplus u_{25}k_{89} \oplus u_{25} \oplus u_{89} \oplus k_{3}k_{67} \oplus k_{3} \\
&\quad \oplus k_{25}k_{89} \oplus k_{25} \oplus k_{67} \oplus k_{89} \oplus 1 \\
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_{9} \\
&\quad \oplus k_{12} \oplus k_{42} \oplus k_{66} \oplus k_{73} \oplus k_{83} \\
u_{88} &= u_{21}k_{21} \oplus u_{21}k_{85} \oplus u_{21} \oplus u_{24}k_{24} \oplus u_{24}k_{88} \oplus u_{24} \oplus u_{46}k_{46} \oplus u_{46}k_{110} \\
&\quad \oplus u_{46} \oplus u_{85} \oplus u_{110} \oplus k_{21}k_{85} \oplus k_{21} \oplus k_{24}k_{88} \oplus k_{24} \oplus k_{46}k_{110} \oplus k_{46} \\
&\quad \oplus k_{85} \oplus k_{88} \oplus k_{110} \oplus 1 \\
u_{4} &= u_{11} \oplus u_{14} \oplus u_{21} \oplus u_{44} \oplus u_{68} \oplus u_{75} \oplus u_{78}k_{14} \oplus u_{78} \oplus u_{85}k_{21} \oplus u_{85} \\
&\quad \oplus u_{108}k_{44} \oplus u_{108} \oplus k_{4} \oplus k_{11} \oplus k_{14} \oplus k_{21} \oplus k_{44} \oplus k_{68} \oplus k_{75} \oplus k_{85} \\
u_{16} &= u_{40} \oplus u_{47} \oplus u_{50} \oplus u_{80}k_{16} \oplus u_{80} \oplus u_{104} \oplus u_{111} \oplus u_{114}k_{50} \oplus u_{114} \oplus k_{16} \\
&\quad \oplus k_{40} \oplus k_{47} \oplus k_{50} \oplus k_{104} \oplus k_{111} \oplus 1 \\
u_{124} &= u_{18}k_{18} \oplus u_{18}k_{82} \oplus u_{18} \oplus u_{60}k_{60} \oplus u_{60}k_{124} \oplus u_{60} \oplus u_{82} \oplus k_{18}k_{82} \oplus k_{18} \\
&\quad \oplus k_{60}k_{124} \oplus k_{60} \oplus k_{82} \oplus k_{124} \\
u_{5} &= u_{2} \oplus u_{35} \oplus u_{59} \oplus u_{66} \oplus u_{69}k_{5} \oplus u_{69} \oplus u_{76}k_{12} \oplus u_{99}k_{35} \oplus u_{99} \oplus u_{123} \\
&\quad \oplus k_{2} \oplus k_{5} \oplus k_{35} \oplus k_{59} \oplus k_{66} \oplus k_{76} \oplus k_{123} \\
u_{79} &= u_{12}k_{12} \oplus u_{12}k_{76} \oplus u_{12} \oplus u_{15}k_{15} \oplus u_{15}k_{79} \oplus u_{15} \oplus u_{37}k_{37} \oplus u_{37}k_{101} \\
&\quad \oplus u_{37} \oplus u_{76} \oplus u_{101} \oplus k_{12}k_{76} \oplus k_{12} \oplus k_{15}k_{79} \oplus k_{15} \oplus k_{37}k_{101} \oplus k_{37} \\
&\quad \oplus k_{76} \oplus k_{79} \oplus k_{101} \\
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} \\
&\quad \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} \\
&\quad \oplus k_{83} \oplus k_{86} \oplus k_{108}
\end{aligned}$$

– 4 Type-2 conditions involving only bits of key: $k_0 = 0, k_{64} = 0, k_{57} = 0, k_{121} = 1$.

Our HATF technique shows that when the above 18 conditions are satisfied, the bias of the HDL approximation will be 2^{-2}. We use 2^{26} randomly chosen data to test the bias and find the actual bias is also $2^{2.0}$. Additionally, we use the HATF to compute the biases of all the 64 output bits. We provide theoretical and experimental results in Table 9.

Table 9: Theoretical and experimental biases with input difference Δ_0, Δ_1 which are active in $(S^0[3][0], S^0[4][0])$ and $(S^0[3][57], S^0[4][57])$. Note since our experiments are done with 2^{26} samples, only those experimental biases that are significantly greater than 2^{-13} are reliable. Those gray values in this table are unreliable.

Bit	0	1	2	3	4	5	6	7	8	9
Theory	0	$2^{-6.0}$	0	$2^{-11.0}$	$2^{-8.0}$	0	$2^{-7.0}$	0	0	$2^{-8.0}$
Expr.	$2^{-8.0}$	$2^{-5.0}$	$2^{-13.1}$	$2^{-10.0}$	$2^{-8.0}$	$2^{-18.2}$	$2^{-7.0}$	$2^{-6.4}$	$2^{-18.4}$	$2^{-8.0}$
Bit	10	11	12	13	14	15	16	17	18	19
Theory	$2^{-11.0}$	$2^{-4.0}$	$2^{-6.0}$	0	$2^{-7.0}$	0	$2^{-7.0}$	$2^{-7.0}$	$2^{-2.0}$	0
Expr.	$2^{-9.0}$	$2^{-4.0}$	$2^{-17.2}$	$2^{-13.4}$	$2^{-5.4}$	$2^{-7.0}$	$2^{-5.4}$	$2^{-6.0}$	$2^{-2.0}$	$2^{-13.7}$
Bit	20	21	22	23	24	25	26	27	28	29
Theory	$2^{-7.0}$	$2^{-5.0}$	0	0	0	$2^{-7.0}$	$2^{-9.0}$	$2^{-4.0}$	0	$2^{-6.0}$
Expr.	$2^{-5.0}$	$2^{-4.4}$	$2^{-15.2}$	$2^{-13.1}$	$2^{-13.8}$	$2^{-7.0}$	$2^{-8.0}$	$2^{-3.0}$	$2^{-7.0}$	$2^{-16.1}$
Bit	30	31	32	33	34	35	36	37	38	39
Theory	0	0	0	0	$2^{-6.0}$	0	$2^{-6.0}$	$2^{-5.0}$	0	0
Expr.	$2^{-13.2}$	$2^{-13.3}$	$2^{-14.4}$	$2^{-14.2}$	$2^{-6.0}$	$2^{-9.1}$	$2^{-5.0}$	$2^{-5.0}$	$2^{-15.1}$	$2^{-14.6}$
Bit	40	41	42	43	44	45	46	47	48	49
Theory	$2^{-4.0}$	$2^{-8.0}$	$2^{-8.0}$	0	$2^{-4.0}$	0	$2^{-7.0}$	$2^{-4.0}$	$2^{-8.0}$	0
Expr.	$2^{-4.0}$	$2^{-7.0}$	$2^{-7.0}$	$2^{-13.9}$	$2^{-4.0}$	$2^{-14.6}$	$2^{-5.4}$	$2^{-4.0}$	$2^{-7.7}$	$2^{-14.0}$
Bit	50	51	52	53	54	55	56	57	58	59
Theory	$2^{-4.0}$	$2^{-4.0}$	0	0	$2^{-5.0}$	$2^{-6.0}$	$2^{-9.0}$	$2^{-7.0}$	$2^{-7.0}$	0
Expr.	$2^{-4.0}$	$2^{-2.4}$	$2^{-16.9}$	$2^{-6.0}$	$2^{-4.4}$	$2^{-6.0}$	$2^{-9.1}$	$2^{-7.0}$	$2^{-6.4}$	$2^{-12.9}$
Bit	60	61	62	63						
Theory	$2^{-6.0}$	0	$2^{-4.0}$	0						
Expr.	$2^{-5.0}$	$2^{-15.4}$	$2^{-4.0}$	$2^{-9.1}$						

F Conditional HDL Key Recovery from Another HDL Approximations

The key recovery phases based on the second, the third and the fourth HDL approximations in Section 5.3 are very similar to the first one. In this section, we provide these details.

F.1 Conditional HDL Key Recovery From HDL Approximation 2

With experiments, we choose 3 Type-1 conditions that affect the HDL bias most significantly as follows,

- 2 Type-0 conditions involving only nonce bits: $u_0 = u_{64}, u_{14} = u_{78}$
- 12 Type-1 conditions involving bits of nonce and key.

$$\begin{aligned}
u_{100} &= u_{33}k_{33} \oplus u_{33}k_{97} \oplus u_{33} \oplus u_{36}k_{36} \oplus u_{36}k_{100} \oplus u_{36} \oplus u_{58}k_{58} \oplus u_{58}k_{122} \\
&\oplus u_{97} \oplus u_{122} \oplus k_{33}k_{97} \oplus k_{33} \oplus k_{36}k_{100} \oplus k_{36} \oplus k_{58}k_{122} \oplus k_{97} \oplus k_{100} \\
&\oplus k_{122} \oplus 1 \\
u_{95} &= u_{28}k_{28} \oplus u_{28}k_{92} \oplus u_{28} \oplus u_{31}k_{31} \oplus u_{31}k_{95} \oplus u_{31} \oplus u_{53}k_{53} \oplus u_{53}k_{117} \\
&\oplus u_{53} \oplus u_{92} \oplus u_{117} \oplus k_{28}k_{92} \oplus k_{28} \oplus k_{31}k_{95} \oplus k_{31} \oplus k_{53}k_{117} \oplus k_{53} \\
&\oplus k_{92} \oplus k_{95} \oplus k_{117} \\
u_{47} &= u_{23} \oplus u_{54} \oplus u_{57} \oplus u_{87}k_{23} \oplus u_{87} \oplus u_{111} \oplus u_{118} \oplus u_{121}k_{57} \oplus u_{121} \oplus k_{23} \\
&\oplus k_{47} \oplus k_{54} \oplus k_{57} \oplus k_{111} \oplus k_{118}
\end{aligned}$$

- 4 Type-2 conditions involving only bits of key: $k_0 = 0, k_{64} = 0, k_{14} = 0, k_{78} = 0$.

With 2^{26} random samples, we find when all the above conditions hold, the bias is approximately $2^{-3.19}$. While when all but the fourth Type-2 conditions hold (*i.e.*, $k_{78} = 1$), the bias would be $2^{-3.79}$. The biases for all the remaining cases are at most $2^{-4.74}$. Thus the statistical test requires about $2^{9.61}$ samples according to Equation (12) and the threshold is 455, if we give up to recover the exact value of k_{78} (Or we can keep the two possibilities of k_{78}, and determine its exact value by exhaustive search after we recovered all remaining key bits).

From these 3 Type-1 conditions, we can know the following values about keys: $k_{33} \oplus k_{97} = c_0, k_{36} \oplus k_{100} = c_1, k_{58} \oplus k_{122} = c_2, k_{33}k_{97} \oplus k_{33} \oplus k_{36}k_{100} \oplus k_{36} \oplus k_{58}k_{122} \oplus k_{97} \oplus k_{100} \oplus k_{122} \oplus 1 = c_3, k_{28} \oplus k_{92} = c_4, k_{31} \oplus k_{95} = c_5, k_{53} \oplus k_{117} = c_6, k_{28}k_{92} \oplus k_{28} \oplus k_{31}k_{95} \oplus k_{31} \oplus k_{53}k_{117} \oplus k_{53} \oplus k_{92} \oplus k_{95} \oplus k_{117} = c_7, k_{23} = c_8, k_{57} = c_9, k_{23} \oplus k_{47} \oplus k_{54} \oplus k_{57} \oplus k_{111} \oplus k_{118} = c_{10}$. Together with the four Type-2 conditions, we can recover another 3 bits of key information (we give up recovering the exact value of k_{78}). Totally, we can recover 14 bits of key information. Since we have 64 opportunities to use this HDL approximation to do the key recovery, and averagely we have 4 opportunities, the key bits we can recover are about 60. The complexity is then $4 \times (64 \times 2^{3+9.61} + 4 \times 8 \times 2^{9.61}) \approx 2^{20.61}$.

F.2 Conditional HDL Key Recovery From HDL Approximation 3

With experiments, we choose 3 Type-1 conditions that affect the HDL bias most significantly as follows,

- 2 Type-0 conditions involving only nonce bits: $u_0 = u_{64}, u_{51} = u_{115}$
- 12 Type-1 conditions involving bits of nonce and key.

$$\begin{aligned} u_{67} &= u_3k_3 \oplus u_3k_{67} \oplus u_3 \oplus u_{25}k_{25} \oplus u_{25}k_{89} \oplus u_{25} \oplus u_{89} \oplus k_3k_{67} \oplus k_3 \\ &\quad \oplus k_{25}k_{89} \oplus k_{25} \oplus k_{67} \oplus k_{89} \oplus 1 \\ u_{76} &= u_{12}k_{12} \oplus u_{12}k_{76} \oplus u_{12} \oplus u_{54}k_{54} \oplus u_{54}k_{118} \oplus u_{54} \oplus u_{118} \oplus k_{12}k_{76} \oplus k_{12} \\ &\quad \oplus k_{54}k_{118} \oplus k_{54} \oplus k_{76} \oplus k_{118} \\ u_2 &= u_{12} \oplus u_{42} \oplus u_{66} \oplus u_{76}k_{12} \oplus u_{76} \oplus u_{83}k_{19} \oplus u_{106}k_{42} \oplus u_{106} \oplus k_2 \oplus k_9 \\ &\quad \oplus k_{12} \oplus k_{42} \oplus k_{66} \oplus k_{73} \oplus k_{83} \end{aligned}$$

- 4 Type-2 conditions involving only bits of key: $k_0 = 0, k_{51} = 0, k_{64} = 0, k_{115} = 0$.

Note the above conditions are not the same with those we provide in Section E.3 of Supplementary Material because we use a similar ideal to achieve better performance. With 2^{26} random samples, we find when all the above conditions hold, the bias is approximately $2^{-3.16}$. If only the second Type-1 condition, the second Type-2 condition and the fourth Type-2 condition do not hold simultaneously, the bias would be $2^{-3.43}$. The biases for all the remaining cases are at most $2^{-5.21}$. Thus the statistical test requires about $2^{9.14}$ samples according to Equation (12) and the threshold is 327, if we give up to recover the exact value of k_{78} (Or we can keep the two possibilities of k_{78}, and determine its exact value by exhaustive search after we recovered all remaining key bits).

From these 3 Type-1 conditions, we can know the following values about keys: $k_3 \oplus k_{67} = c_0, k_{25} \oplus k_{89} = c_1, k_{58} \oplus k_{122} = c_2, k_3k_{67} \oplus k_3 \oplus k_{25}k_{89} \oplus k_{25} \oplus k_{67} \oplus k_{89} = c_3, k_{12} \oplus k_{76} = c_4, k_{54} \oplus k_{118} = c_5, k_{12}k_{76} \oplus k_{12} \oplus k_{54}k_{118} \oplus k_{54} \oplus k_{76} \oplus k_{118} = c_6, k_{12} = c_7, k_{19} = c_8, k_{42} = c_9, k_2 \oplus k_9 \oplus k_{12} \oplus k_{42} \oplus k_{66} \oplus k_{73} \oplus k_{83} = c_{10}$. Together with the four Type-2 conditions, we can recover another 4 bits of key information (we give up recovering the exact value of $(k_{51}, k_{115}, k_{12}k_{76} \oplus k_{12} \oplus k_{54}k_{118} \oplus k_{54} \oplus k_{76} \oplus k_{118})$). Totally, we can recover 15 bits of key information. Since we have 64 opportunities to use this HDL approximation to do the key recovery, and averagely we have 4 opportunities, the key bits we can recover are about 60. The complexity is then $4 \times (64 \times 2^{3+9.14} + 4 \times 8 \times 2^{9.14}) \approx 2^{20.14}$.

F.3 Conditional HDL Key Recovery From HDL Approximation 4

With experiments, we choose 3 Type-1 conditions that affect the HDL bias most significantly as follows,

- 2 Type-0 conditions involving only nonce bits: $u_0 = u_{64}, u_{57} = u_{121}$

– 12 Type-1 conditions involving bits of nonce and key.

$$\begin{aligned}
u_{67} &= u_3k_3 \oplus u_3k_{67} \oplus u_3 \oplus u_{25}k_{25} \oplus u_{25}k_{89} \oplus u_{25} \oplus u_{89} \oplus k_3k_{67} \oplus k_3 \\
&\quad \oplus k_{25}k_{89} \oplus k_{25} \oplus k_{67} \oplus k_{89} \oplus 1 \\
u_{42} &= u_2 \oplus u_{12} \oplus u_{66} \oplus u_{76}k_{12} \oplus u_{76} \oplus u_{83}k_{19} \oplus u_{106}k_{42} \oplus u_{106} \oplus k_2 \oplus k_9 \\
&\quad \oplus k_{12} \oplus k_{42} \oplus k_{66} \oplus k_{73} \oplus k_{83} \\
u_{124} &= u_{18}k_{18} \oplus u_{18}k_{82} \oplus u_{18} \oplus u_{60}k_{60} \oplus u_{60}k_{124} \oplus u_{60} \oplus u_{82} \oplus k_{18}k_{82} \oplus k_{18} \\
&\quad \oplus k_{60}k_{124} \oplus k_{60} \oplus k_{82} \oplus k_{124}
\end{aligned}$$

– 4 Type-2 conditions involving only bits of key: $k_0 = 0, k_{64} = 0, k_{57} = 0, k_{121} = 0$.

With 2^{26} random samples, we find when all the above conditions hold, the bias is approximately $2^{-3.19}$. Otherwise, the bias will be at most $2^{-4.46}$. Thus the statistical test requires about $2^{9.94}$ samples according to Equation (12) and the threshold is 327.

From these 3 Type-1 conditions, we can know the following values about keys: $k_3 \oplus k_{67} = c_0, k_{25} \oplus k_{89} = c_1, k_{58} \oplus k_{122} = c_2, k_3k_{67} \oplus k_3 \oplus k_{25}k_{89} \oplus k_{25} \oplus k_{67} \oplus k_{89} = c_3 k_{12} = c_4, k_{19} = c_5, k_{42} = c_6, k_2 \oplus k_9 \oplus k_{12} \oplus k_{42} \oplus k_{66} \oplus k_{73} \oplus k_{83} = c_7, k_{18} \oplus k_{82} = c_8, k_{60} \oplus k_{124} = c_9, k_{18}k_{82} \oplus k_{18} \oplus k_{60}k_{124} \oplus k_{60} \oplus k_{82} \oplus k_{124} = c_{10}$ Together with the four Type-2 conditions, we can recover another 4 bits of key information, Totally, we can recover 15 bits of key information. Since we have 64 opportunities to use this HDL approximation to do the key recovery, and averagely we have 4 opportunities, the key bits we can recover are about 60. The complexity is then $4 \times (64 \times 2^{3+9.14} + 4 \times 8 \times 2^{9.94}) \approx 2^{21.00}$.

F.4 Strategy to Utilize All Equations

In order to reduce the complexities, we choose only Type-1 conditions to do the key recovery. However, once we succeed in detecting the positions that satisfy the four Type-2 conditions, we can continue to take the remaining conditions into consideration. To do it, based on the 7 Type-1 and Type-2 conditions that are already satisfied, we choose another one Type-1 condition among the remaining ones. Whether this new condition holds will lead to different bias, we can use this property to recover key information based this new condition. Since we do not need to exhaust all 64 positions to find the correct positions to do it, the complexities for these additional steps are neglected. Finally, we can take advantage of all conditions without increasing the complexities.

G Application to 6-Round Ascon Initialization

We perform here a 2nd order HDL cryptanalysis of 6-round of the initialization of Ascon. With an exhaustive search using Algorithm 3 for all the possible positions $(0, i_1 > 0)$ for input and positions for output, there are four combinations of the input difference and output mask $(\Delta_0, \Delta_1, \lambda)$ leading to a significantly high bias:

1. Δ_0 is active in $(S^0[3][0], S^0[4][0])$ and Δ_1 is active in $(S^0[3][5], S^0[4][5])$, λ is active in $S^{5.5}[0][37]$. The bias is -2^{-30}.
2. Δ_0 is active in $(S^0[3][0], S^0[4][0])$ and Δ_1 is active in $(S^0[3][59], S^0[4][59])$, λ is active in $S^{5.5}[0][32]$. The bias is -2^{-30}.
3. Δ_0 is active in $(S^0[3][0], S^0[4][0])$ and Δ_1 is active in $(S^0[3][28], S^0[4][28])$, λ is active in $S^{5.5}[0][4]$. The bias is -2^{-37}.
4. Δ_0 is active in $(S^0[3][0], S^0[4][0])$ and Δ_1 is active in $(S^0[3][51], S^0[4][51])$, λ is active in $S^{5.5}[0][55]$. The bias is -2^{-37}.

We take the first two 2nd order HDL approximations to mount the key recovery attacks. Since we obtain conditions from the first two rounds, which is the same as the 5-round case, we expect a similar situation except that we will need more samples for the statistical testing to distinguish the right case. Suppose that there are also three Type-1 conditions after removing some redundant conditions, the sample amount is approximately calculated as $N \approx 2^{62.06}$. After exhausting all 64 positions, we have 2 opportunities to do the key recovery attacks, thus the approximate data and time complexities are

$$4 \times 2 \times 64 \times 2^{62.06+3} + 4 \times 2 \times 2 \times 8 \times 2^{62.06} \approx 2^{74.10}.$$

Note that because of the extremely small bias, we cannot perform a more detailed analysis based on experiments as we did for the 5 rounds, thus the above complexities are rough estimations. However, there is no doubt that the HDL attacks on 6-round Ascon are possible since the biases are much greater than 2^{-64}. We emphasize that in [LLL21] Liu *et al.* remarked that they made a lot of efforts but could not find any DL approximation with bias larger than 2^{-64}. This demonstrates well the advantage of the HDL cryptanalysis. We note that this is the second type of attack applicable to 6 rounds of Ascon initialization besides the cube-like attacks [LDW17,DEMS15,RHSS21,LZWW17].

H HD Cryptanalysis of Grain v1

Grain v1 is a stream cipher proposed by Hell *et al.* [HJM07] which has been selected in the eSTREAM hardware profile. Grain v1 uses an 80-bit secret key $K = (k_0, k_1, \ldots, k_{79})$ and a 64-bit initial value $V = (v_0, v_1, \ldots, v_{63})$. It consists of three main building blocks: an 80-bit linear feedback shift register (LFSR), an 80-bit non-linear feedback shift register (NFSR) and a non-linear output function. In [LLL21], Liu *et al.* proposed the conditional differential attacks (for stream ciphers whose output is one bit, the DL and differential attacks are identical) on the 125-round Grain v1 initialization with a theoretical bias $2^{-20.77}$.

To give a comparison between the effects of the 2nd and 1st order differential attacks, we tested the 2nd order differential on Grain v1 based on HATF to see whether we could reach more rounds. In [LLL21], the authors used an input difference Δ_0 which is active in the 21th and 46th bits of the IV. We set it as one component of our 2nd order difference, say Δ_0, and use another component Δ_1 which is active in the 19th and 44th bits of the IV. Therefore, our 2nd order difference is $\boldsymbol{\Delta} = (\Delta_0, \Delta_1)$. We apply Algorithm 3 with $\boldsymbol{\Delta}$ as input to 130-round Grain v1, and set r_1 as 50 (conditions are imposed for the first 50 rounds, the same setting as [LLL21]). The HATF of the first output bit is calculated and the bias is estimated by Algorithm 4. Since the difference expression is very complicated, to make the computation feasible, we substitute only one linearly isolated term using Q every time, and finally use the piling lemma to estimate the overall bias with all the biases we get from all terms in e_l (see Line 5 in Algorithm 4). Finally, our experiment shows that the output 2nd order difference of 130-round Grain v1 has a bias approximately equal to $2^{-30.18}$. This conditional HD is 5 rounds longer than the previous best conditional differential distinguisher. Unfortunately, the bias is too small to verify it experimentally, as we would need about $2^{60.36}$ data. Considering that some freedom degrees of the IV bits are used to meet the conditions that we impose in the first 50 rounds, this HD approximation cannot be used in key-recovery attacks. Therefore, this approximation shows some non-randomness property of Grain v1 and we present it for a comparison with its 1st order conditional differential counterpart.

I Practical HDL Distinguishers Based on Cube Testers

In this section, we show how to construct practical HDL distinguishers based on cube testers. The cube tester technique was originally proposed at FSE 2009 as a general method to test the non-randomness of the superpoly for stream ciphers [ADMS09]. We have seen that the HDL attack on a Boolean function is equivalent to the cube attack on its DSF, so we can also apply cube testers to its DSF function then convert them back to HDL attacks on the original Boolean function. Actually, the experimental methods which have been extensively used in previous DL works, *e.g.*, [DEMS15,AFK+08], can also be viewed as cube testers. When applied to the DSF, an advantage is that we can take different X and $\boldsymbol{\Delta}$ to simplify the $\mathrm{DSF}_{X,\boldsymbol{\Delta},f}$. In this section, we use the cube tester to construct practical HDL distinguishers for the reduced-round Ascon permutation, Xoodoo [DHAK18] and ChaCha [Ber08a].

I.1 5-Round HDL Distinguishers for Ascon

In [DEMS15], the designers experimentally found a DL distinguisher with bias equal to 2^{-9} for 5 rounds of the Ascon initialization. In this subsection, we provide more HDL distinguishers for this variant. From the previous study, when elements in an ℓ-th order difference $\boldsymbol{\Delta} = (\Delta_0, \Delta_1, \ldots, \Delta_{\ell-1})$ are active both in the third and fourth words and the output bit is active in a single bit, the bias could be higher. Naturally, in our experiments, we always let the difference to be active in these two words and consider only one bit of output. Therefore, we need to choose l positions from $0, 1, \ldots, 63$ to incorporate the differences. Recall that in the key-recovery attack on 5-round Ascon, we could impose some conditions to enhance the bias. For simplicity, we only consider the Type-0 and Type-2

Table 10: Some HDL approximations obtained using experiments for 5-round Ascon initialization. Type-0 means we impose Type-0 conditions into the cube while Type-0/2 means both Type-0 and Type-2 conditions are imposed. Since Type-2 conditions are related to the key, the corresponding HDL are considered as conditional HDL.

Order	**Input Diff. / Output Mask**	**Bias**$(-\log)$ **Type-0**	**Bias**$(-\log)$ **Type-0/2**
3	`(0,24,33)/51`	6.52	3.56
4	`(0,9,15,41)/27`	6.44	2.14
5	`(0,9,24,51,55)/18`	5.31	2.02
6	`(1,12,18,22,21,52)/49`	4.88	1.89
7	`(10,13,21,31,49,55,61)/28`	4.03	1
8	`(0,3,10,11,26,28,31,55)/60`	2.46	1
9	`(8,13,14,16,21,25,39,42,46)/12`	1.76	1
10	`(4,14,23,27,35,39,41,49,51,55)/0`	1.09	1
11	`(19,24,33,35,36,48,54,57,59,62,63)/27`	1.04	1

conditions. Note if some type-2 conditions are imposed, the corresponding HDL approximation is conditional on the key, which we cannot access. When ℓ is not large, *e.g.*, $\ell \leq 4$, we can exhaust all combinations of input differences and output masks in the aforementioned form. When ℓ is large, *e.g.*, $\ell \geq 5$, it is costly to exhaust all possibilities of the ℓ-th order differences. Thus, we choose randomly the positions of ℓ-th order differences and the output bit. For each combination of the differences and masks, we compute their bias with 2^{15} samples. After we detect some biases that are significantly larger than 2^{-7}, we use 2^{26} samples to confirm these biases. Some ℓ-th order HDL approximations are shown in Table 10. If we take the 8th order HDL with bias equal to $2^{-2.46}$ (with Type-0 conditions being imposed) to distinguish 5-round Ascon initialization, we need about $2^{4.92}$ samples, *i.e.*, $2^{4.92+8} = 2^{12.92}$ data/time complexity. The previous best distinguisher for 5 rounds is the integral distinguisher proposed in [RHSS21] requiring 2^{16} data/time complexity.

I.2 4-Round Deterministic HDL Distinguisher for Xoodoo

Xoodoo [DHAK18] is an efficient 384-bit permutation designed by the Keccak Team[9]. The state of Xoodoo is arranged into a $4 \times 3 \times 32$ cube and a state bit is denoted by $S[x][y][z]$. One round of Xoodoo consists of the following

[9] `https://keccak.team`

operations.

$$
\begin{aligned}
&S[x][y][z] = S[x][y][z] \oplus \bigoplus_y S[x-1][y][z-5] \oplus \bigoplus_y S[x-1][y][z-14] \\
&S[x][1][z] = S[x-1][1][z], S[x][2][z] = S[x][2][z-11] \\
&S[0][0] = S[0][0] \oplus RC_i \\
&S[x][y][z] = S[x][y][z] \oplus ((S[x][y+1][z] \oplus 1) \cdot S[x][y+2][z]) \\
&S[x][1][z] = S[x][1][z-1], S[x][2][z] = S[x-2][2][z-8]
\end{aligned}
$$

The total number of rounds in Xoodoo is 12, but in some modes the core permutation calls a 6-round Xoodoo permutation. More details of Xoodoo including the constants RC_i can be found in its specification [DHAK18].

In our cryptanalysis of Xoodoo, we do not consider the linear layers before and after the nonlinear operations in the first and last round. Since Xoodoo is a permutation, such an assumption is reasonable. Before applying the cube tester to Xoodoo, we first analyze its nonlinear operation $S[x][y][z] = S[x][y][z] \oplus ((S[x][y+1][z] \oplus 1) \cdot S[x][y+2][z])$. Intuitively, if we let $S[x][y+1][z] = 0$ and $S[x][y][z] = a[x][y+2][z]$ and set the difference active in both $S[x][y][z]$ and $S[x][y+2][z]$, the nonlinear operation will be simplified. This way, the difference in $S[x][y][z]$ after the nonlinear operation will be canceled. We apply this setting to the 96 bits represented by $S[0]$ as follows:

1. Let $S[0][0][z] = S[0][2][z]$ and $S[0][1][z] = 0$,
2. Exhaust all 2nd order differences $\boldsymbol{\Delta} = (\Delta_0, \Delta_1)$ where Δ_0 and Δ_1 are both active in $S[0][0][z]$ and $S[0][2][z]$ but $\Delta_0 \neq \Delta_1$.

We observe that with these settings many output bits after 4 rounds are highly biased. For example, if Δ_0 is active in $S[0][0][0]$ and $S[0][2][0]$, Δ_1 is active in $S[0][0][20]$ and $S[0][2][20]$, the bias of $S[0][0][0]$ after 4 rounds would be $\frac{1}{2}$, *i.e.*, we found a deterministic 2nd DL distinguisher for 4-round Xoodoo. We note that before our work there was another deterministic rotational-differential-linear distinguisher found for 4-round Xoodoo [LSL21]. However, no DL distinguisher has been reported for 4-round Xoodoo until now. Unfortunately, with the same method, we did not find useful 2nd order HDL for 5 rounds of Xoodoo.

I.3 3-, 4- and 4.5-Round HDL Distinguisher for ChaCha

ChaCha is a variant of Salsa which are both designed by Bernstein [Ber08a,Ber08b]. Because of its high software efficiency, ChaCha has been adopted by the TLS protocol [LCM+16].

The state of ChaCha is of size 64 bytes or 512 bits, which is divided into 16 words, each of 32 bits. These words are framed of a 4×4 matrix. In the initial matrix denoted by X^0, the 1st row consists of 4 constants $c_0 =$ `0x61707865`, $c_1 =$ `0x3320646e`, $c_2 =$ `0x79622d32` and $c_3 =$ `0x6b206574`. The second and third row consist of 8 key words $k_0, k_1, \ldots, k_7$ and the fourth row consists of the two 32-bit nonces v_0, v_1 and two 32-bit counters t_0, t_1. The nonces and counters are usually considered as IVs, which we can control. The state X^0 is illustrated as follows,

$$X^0 = \begin{bmatrix} X_0^0 & X_1^0 & X_2^0 & X_3^0 \\ X_4^0 & X_5^0 & X_6^0 & X_7^0 \\ X_8^0 & X_9^0 & X_{10}^0 & X_{11}^0 \\ X_{12}^0 & X_{13}^0 & X_{14}^0 & X_{15}^0 \end{bmatrix} = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ t_0 & t_1 & v_0 & v_1 \end{bmatrix}.$$

The round function of ChaCha is based on an operation called quarter-round ($\mathcal{QR}$) which operates on a 4 tuple (a, b, c, d) and updates it as follows,

$$\begin{aligned} a' &= a + b, \quad d' = (d \oplus a') \lll 16, \ c' = c + d', \quad b' = (b \oplus c') \lll 12, \\ a'' &= a' + b', d'' = (d' \oplus a'') \lll 8, c'' = \ c' + d'', b'' = (b' \oplus c'') \lll 7. \end{aligned}$$

i.e., $\mathcal{QR}(a, b, c, d) \xrightarrow{(a',b',c',d')} (a'', b'', c'', d'')$.

$\mathcal{QR}$ is applied on the 4 words of each column in the odd rounds and each diagonal in the even rounds. The state after r rounds is denoted by

$$X^0 = \begin{bmatrix} X_0^r & X_1^r & X_2^r & X_3^r \\ X_4^r & X_5^r & X_6^r & X_7^r \\ X_8^r & X_9^r & X_{10}^r & X_{11}^r \\ X_{12}^r & X_{13}^r & X_{14}^r & X_{15}^r \end{bmatrix}$$

The output key-stream block Z is executed as $Z = X^0 + X^R$ for ChaCha/R. A half-round represents the update of (a, b, c, d) to (a', b', c', d') in the $\mathcal{QR}$ operation. Thus, ChaCha/R.5 means a R full and a half round function. We continue to use $X[i]$ to represent the i-th bit of the word X, but only within this subsection, $X[0]$ stands for the least significant bit. This is for consistency with previous work related to ChaCha.

Currently, the most efficient methods for analyzing ChaCha have been differential-linear cryptanalysis [AFK$^+$08,SZFW12,CM16,BLT20]. Interestingly, we notice that the HDL idea has been partially used in the previous cryptanalysis on ChaCha but the terminology *higher-order differential-linear attack* was not used. In [SZFW12], Shi *et al.* proposed some higher biased truncated 2nd order differentials whose outputs are one bit for ChaCha/3, which had been better than the 1st order truncated differentials. In the appendix of [CM16], Choudhuri *et al.* gave several truncated 2nd order differentials whose outputs are multiple bits for 4-round ChaCha by appending one-round linear approximations to the distinguishers from [SZFW12]. However, in general the HDL cryptanalysis has not considered extensively for ChaCha. In this subsection, we give the best distinguishers based on HDL for ChaCha/3.5, ChaCha/4 and ChaCha/4.5, which shows that the HDL cryptanalysis has a larger potential than expected and probably deserves more attention from the cryptography community.

To establish these distinguishers, we first use experiments to find high biased 2nd order HDL whose output is active in one bit, and secondly append it with a 1.5-round deterministic linear approximation. Since the round functions of ChaCha are different for odd and even rounds, these 1.5-round linear approximations we use are also different. Yet, all of them are similar to the 1.5-round linear approximation proposed in [CM16, Section 3.1.3] and can be built similarly.

3.5-round 2nd order HDL with bias close to $\frac{1}{2}$. The input is the 2nd order difference $(\varDelta_0, \varDelta_1)$ where $\varDelta_0$ is active in $X^0_{12}[0]$ and $\varDelta_1$ is active in $X^0_{14}[0]$, the output is the difference active in $X^2_8[0]$. The bias of this 2nd order HDL is approximately $\frac{1}{2}$ since among 2^{30} samples, only 131 led to a nonzero difference. The 1.5-round linear approximation with bias $\frac{1}{2}$ is

$$X^2_8[0] = X^{3.5}_0[0] \oplus X^{3.5}_0[8] \oplus X^{3.5}_3[0] \oplus X^{3.5}_4[12] \oplus X^{3.5}_9[0] \oplus X^{3.5}_{11}[0] \oplus X^{3.5}_{12}[0] \oplus X^{3.5}_{15}[16] \oplus X^{3.5}_{15}[24].$$

We connect the first 2-round 2nd order HDL approximation with the 1.5-round linear approximation to get a 3.5-round 2nd order HDL distinguisher whose bias is almost $\frac{1}{2}$.

4-round 2nd order HDL with bias approximately $2^{-1.19}$. The input is the 2nd order difference $(\varDelta_0, \varDelta_1)$ where $\varDelta_0$ is active in $X^0_{13}[16]$ and $\varDelta_1$ is active in $X^0_{14}[0]$, the output is active in $X^2_8[0]$. The bias of this 2nd order HDL is approximately $0.4386 \approx 2^{-1.19}$, which is close to a deterministic distinguisher. The 1.5-round linear approximation with bias $\frac{1}{2}$ is

$$X^{2.5}_8[0] = X^4_1[0] \oplus X^4_1[16] \oplus X^4_2[0] \oplus X^4_6[7] \oplus X^4_8[0] \oplus X^4_{11}[0] \oplus X^4_{12}[24] \oplus X^4_{13}[0] \oplus X^4_{13}[8].$$

We connect the first 2.5-round 2nd order HDL approximation with the 1.5-round linear approximation to get a 4-round 2nd order HDL distinguisher whose bias is about $2^{-1.19}$.

4.5-round 2nd order HDL with bias approximately $2^{-4.81}$. The input is the 2nd order difference $(\varDelta_0, \varDelta_1)$ where $\varDelta_0$ is active in $X^0_{14}[12]$ and $\varDelta_1$ is active in $X^0_{15}[15]$, the output is active in $X^2_8[0]$. The bias of this 2nd order HDL is approximately $0.0357 \approx 2^{-4.81}$. The 1.5-round linear approximation with bias $\frac{1}{2}$ is

$$X^3_8[0] = X^{4.5}_0[0] \oplus X^{4.5}_0[8] \oplus X^{4.5}_1[0] \oplus X^{4.5}_5[12] \oplus X^{4.5}_9[0] \oplus X^{4.5}_{11}[0] \oplus X^{4.5}_{12}[16] \oplus X^{4.5}_{12}[24] \oplus X^{4.5}_{15}[0].$$

We connect the first 3-round 2nd order HDL approximation with the 1.5-round linear approximation to get the 4-round 2nd order HDL distinguisher whose bias is about $2^{-4.81}$.

The biases of these three 2nd order HDL distinguishers are significantly higher than all previous DL distinguishers, a detailed comparison has been given in Table 1. With these higher biased approximations, the distinguishing attacks on ChaCha/3.5, ChaCha/4 and ChaCha/4.5 can be improved. With a conventional method where we need ε^{-2} samples to distinguish the cipher from a random permutation, we need about 2^2, $2^{2.38} \approx 11$ and $2^{9.62} \approx 787$ samples for the three variants of ChaCha. Considering that each sample contains 4 texts, the complexity is then $2^4 = 16$, $2^{4.38} \approx 44$ and $2^{11.61} \approx 3184$ respectively. On the same scale, the previous best DL distinguishers for 4 and 4.5 rounds required $2 \times 2^{6.66} \approx 202$ and $2 \times 2^{12.28} \approx 9947$ chosen texts. The HDL achieves a better performance.

We also tried to construct 2nd order HDL for ChaCha/5. However, we did not find advantageous approximations compared to the existing DL approximations. Firstly, no 2nd order HDL for the first 3.5-round ChaCha was found in our experiments, so we have to construct a 5-round approximation with a 3-round 2nd order HDL and a 2-round linear approximation. We reuse the 2nd order HDL which has been introduced for ChaCha/4.5, *i.e.*, the input is the 2nd order difference $(\varDelta_0, \varDelta_1)$ where $\varDelta_0$ is active in $X_{14}^0[12]$ and $\varDelta_1$ is active in $X_{15}^0[15]$, the output is active in $X_8^2[0]$. The probability of this 2nd order HDL is approximately $2^{-4.81}$. In this case, the bias for the optimal linear approximation with the input mask being active in $X^2[8][0]$ is 2^{-2}, one of such approximations can also be constructed by the method in [CM16] as follows,

$$\begin{aligned} X_8^3[0] =& X_0^{4.5}[0] \oplus X_0^{4.5}[8] \oplus X_1^{4.5}[0] \oplus X_5^{4.5}[12] \oplus X_9^{4.5}[0] \oplus X_{11}^{4.5}[0] \oplus X_{12}^{4.5}[16] \\ & \oplus X_{12}^{4.5}[24] \oplus X_{15}^{4.5}[0] \end{aligned}$$

Thus, the overall bias of the approximation for ChaCha/5 is $2^{-1} \times 2^{-3.81} \times 2^{-4} \approx 2^{-8.81}$. While the previous best DL has a bias equal to $2^{-8.2}$. Therefore, for ChaCha/5 we found that 2nd order HDL is not better than DL. It implies that when we need to append linear approximations to a higher-order HDL to extend the rounds, the overall bias of the approximations would decrease faster than its DL counterpart.

J Distinguishing Two Normal Distributions with Statistical Testing

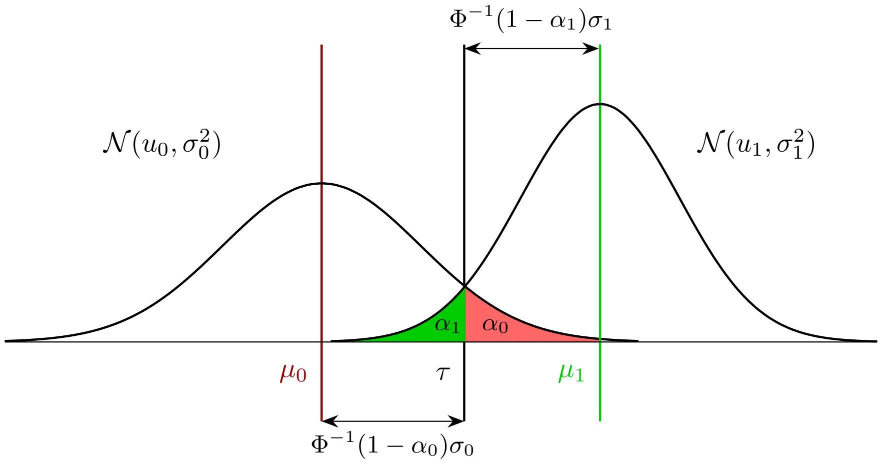
Suppose that we have known a statistics T obeys either $\mathcal{N}(\mu_0, \sigma_0^2)$ or $\mathcal{N}(\mu_1, \sigma_1^2)$ and *w.l.o.g.* $u_0 < u_1$, we want to judge which one T really follows. The method is to find a threshold $\mu_0 < \tau < u_1$ such that when $T < \tau$ we judge $T \sim \mathcal{N}(\mu_0, \sigma_0^2)$, 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 3. Let $\varPhi$ be the cumulative distribution functions of the standard normal distribution. According to Figure 3 we have

$$\begin{cases} \tau = \mu_0 + \varPhi^{-1}(1-\alpha_0)\sigma_0 = \mu_1 - \varPhi^{-1}(1-\alpha_1)\sigma_1 \\ \mu_1 - \mu_0 = \varPhi^{-1}(1-\alpha_0)\sigma_0 + \varPhi^{-1}(1-\alpha_1)\sigma_1 \end{cases} \tag{11}$$

In Equation (6), the number of samples is the only parameter influencing the mean μ and variance σ^2 of the normal distribution (with known bias). Let ε_0 and ε_1 represent the bias of wrong and right cases, respectively. Substituting the $\mu_0, \mu_1, \sigma_0^2, \sigma_1^2$ with $N \times (\frac{1}{2} + \varepsilon_0), N \times (\frac{1}{2} + \varepsilon_1), N \times (\frac{1}{4} - \varepsilon_0^2), N \times (\frac{1}{4} - \varepsilon_1^2)$,



$\Phi^{-1}(1-\alpha_1)\sigma_1$
$\mathcal{N}(u_0,\sigma_0^2)$
$\mathcal{N}(u_1,\sigma_1^2)$
α_1
α_0
μ_0
τ
μ_1
$\Phi^{-1}(1-\alpha_0)\sigma_0$

Fig. 3: The relationship among u_0, u_1, τ, α_0 and α_1

respectively, we can get the formula to compute the necessary simple amount with predefined probabilities of errors α_0 and α_1 as

$$N = \left(\frac{\sqrt{\frac{1}{4} - \varepsilon_0^2}\ \Phi^{-1}(1-\alpha_0) + \sqrt{\frac{1}{4} - \varepsilon_1^2}\ \Phi^{-1}(1-\alpha_1)}{\varepsilon_1 - \varepsilon_0} \right)^2 . \qquad (12)$$

We set $\alpha_1 = 0.05$, that is the right case would be judged as wrong cases with probability 0.05 at worst. We set α_0 to make sure that the probability that at least one wrong case among m wrong cases is identified as the right case is no larger than 0.05, therefore from $1-(1-\alpha_0)^m = 0.05$, we derive $\alpha_0 = 1 - 2^{\frac{\log(1-0.05)}{m}}$. In terms of our case, we have 7 wrong cases, so $m = 7$. The bias of the right case is $\varepsilon_1 = 2^{-3.19}$ while the bias for wrong cases is at most $\varepsilon_0 = 2^{-4.47}$. Thus, $2^{9.94}$ samples are enough to identify the right case according to Equation (12) and the threshold is $\tau = 572$ according to Equation (11).