

Enhancing Differential-Neural Cryptanalysis

Zhenzhen Bao^{1,2,4} , Jian Guo² , Meicheng Liu³ , Li Ma³ , and Yi Tu² 

¹ Institute for Network Sciences and Cyberspace, BNRIst, Tsinghua University, Beijing, China zzbao@tsinghua.edu.cn

² School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore guojian@ntu.edu.sg, tuyi0002@e.ntu.edu.sg

³ State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China liumeicheng@iie.ac.cn, skloismary@gmail.com

⁴ Zhongguancun Laboratory, Beijing, China

Abstract. In CRYPTO 2019, Gohr shows that well-trained neural networks can perform cryptanalytic distinguishing tasks superior to traditional differential distinguishers. Moreover, applying an unorthodox key guessing strategy, an 11-round key-recovery attack on a modern block cipher SPECK32/64 improves upon the published state-of-the-art result. This calls into the next questions. To what extent is the advantage of machine learning (ML) over traditional methods, and whether the advantage generally exists in the cryptanalysis of modern ciphers? To answer the first question, we devised ML-based key-recovery attacks on more extended round-reduced SPECK32/64. We achieved an improved 12-round and the first practical 13-round attacks. The essential for the new results is enhancing a classical component in the ML-based attacks, that is, the neutral bits. To answer the second question, we produced various neural distinguishers on round-reduced SIMON32/64 and provided comparisons with their pure differential-based counterparts.

Keywords: Differential Cryptanalysis, Machine Learning, SPECK, SIMON, Neural Distinguisher, Key Recovery, Neutral Bits

1 Introduction

Cryptography and machine learning (ML) share many concerns, *e.g.*, distinguishing, classification, decision, searching, and optimization. It has been a long-standing challenge to answer whether computers could “learn to perform cryptanalytic tasks” [27]. These years, ML has made rapid progress in application domains ranging from machine translation, visual recognition, and autonomous vehicles to playing board games at superhuman levels [28]. ML has also been used to construct new types of cryptographic schemes [1] or crack ancient ciphers [16].

However, whether ML models can learn from scratch and then break modern ciphers at a superior level is still unpredictable. Nevertheless, one can still look forward to the prospect that ML approaches become substantial positive

additions to the existing cryptanalysis toolkit, which has already been true in side-channel cryptanalysis [26].

For using ML to assist classical cryptanalysis, there are several questions to explore. That might include the follows:

- Can ML models learn new features with/without prior human cryptanalysis?
- Can ML provide more accurate and efficient measurements of known features?
- Can various ML approaches combined with various cryptanalysis techniques perform cryptanalysis tasks at a superior level to orthodox techniques, then be interpreted, and in turn, help to develop innovative and general cryptanalytic techniques?

In CRYPTO 2019, a remarkable work by Gohr [15] shows that commonly used neural networks could be trained to be superior cryptographic distinguishers. The work shed light on positive answers to the first two questions. It showed that deep neural-network distinguisher could exploit features that strong classical distinguishers fail to capture for SPECK. In [15], neural networks were trained with principles of differential cryptanalysis in mind. They show a remarkable capability in distinguishing attacks. More importantly, combining them with classical differentials and a highly selective key search policy forms a powerful key-recovery attack. Specifically, using the obtained neural distinguishers (\mathcal{ND} s) as the main engines, prepending them with a classical differential (\mathcal{CD}), applying basic reinforcement learning mechanisms, *i.e.*, the Upper Confidence Bounds (UCB) and Bayesian optimization, an 11-round key-recovery attack on SPECK32/64 can achieve an unparalleled speed. However, to attack more rounds, one has to extend either the classical component, *i.e.*, the prepended \mathcal{CD} , or the \mathcal{ND} . Both are facing obstacles that have not been overcome since [15].

In EUROCRYPT 2021, Ghor’s \mathcal{ND} got a deeper interpretation by Benamira *et al.* [7]. They were found to have learned not only the differential distribution on the output pairs but also the differential distribution in penultimate and ante-penultimate rounds. Still, the other enhanced new component, *i.e.*, the UCB and Bayesian optimization-based key-recovery phase in the superior 11-round attack in [15], has not been fully interpreted and theorized, thus still missing necessary guidance on tuning various parameters and sound theoretical models on analyzing data/time complexity and success probability.

Note that one of the main difficulties in evaluating the scope of applicability of ML algorithms is the lack of a formally specified theoretical model. Strong theoretical models for ML-based cryptanalysis are vital for generalizing the techniques. However, in parallel or even before our community achieve sound theoretical models, devising a sufficient number of successful attacks as positive examples in this new setting is essential. Without providing the best attacks as examples, it might be harder to obtain a theoretical model that produces the most powerful attacks. This work provides strong positive examples and extensive experimental data to support the first steps towards a realistic theoretical model for effective ML-based cryptanalysis.

OUR CONTRIBUTION. The contribution of this work includes the following.

- Practical attacks and rules of thumb
 - The first practical 13-round and an improved 12-round \mathcal{ND} -based key-recovery attacks on SPECK32/64 are devised. They have considerable advantages in time complexity over attacks devised using orthodox cryptanalysis. In addition, the first practical 16-round \mathcal{ND} -based key-recovery attack on SIMON32/64 is devised, which has a considerable advantage in data complexity. These results are summarized in Table 1.
 - Substantial illustrations unveil previously hidden details of the unorthodox key-recovery phase. Furthermore, observations derived from the illustrations provide rules of thumb for tuning critical parameters.
- Applications of enhanced cryptanalytic techniques

The improved attacks are achieved by enhancing the classical components in the differential-neural attack scheme in [15], which are the \mathcal{CD} 's neutral bits (NBs). NBs and NB sets were first introduced by Biham and Chen in the cryptanalysis of hash function SHA-0 [9]. Later, many extensions and related concepts were proposed and applied in classical cryptanalysis, including message modification [32], tunnels [22], boomerangs [21], probabilistic NBs [4], and free bits [23]. Flipping an NB of a differential's conforming pair, the resulting pair also conforms to the differential. Thus, NBs can be used to derive a batch of data pairs from a single pair, and they conform or do not conform to the differential simultaneously. Single-bit NBs are employed in ML-based attacks in [15] to boost signals from \mathcal{ND} s. However, NBs of long \mathcal{CD} are too scarce to boost signals from a weak but long \mathcal{ND} , thus inhibiting the ML-based attack from extending more rounds. In this work, we exploit various generalized NBs to make weak \mathcal{ND} usable again. Particularly, we employed conditional simultaneous neutral bit-sets (CSNBS) and switching bit for adjoining differentials (SBfAD), which are essential for achieving efficient 12-round and practical 13-round attacks.
- New observations
 - We note the output difference of the \mathcal{CD} matters to \mathcal{ND} , but not the input difference. Hence, more than one \mathcal{CD} can be prepended to \mathcal{ND} , as long as they share the same output difference. Some neutral bits can be shared by multiple such differentials. Using such differentials might enable data reuse, thus slightly reducing data complexity.
 - We find that there are additional constraints on subkeys for some differential trails used in the presented attacks as well as the previous best attacks on SPECK32/64 [10, 13, 29]. Thus, the attacks only work for a subspace of the keys, *i.e.*, weak keys up to half of the keyspace.
- Various \mathcal{ND} s and DDT-based distinguishers (\mathcal{DD}) for SIMON32/64
 - Besides the Residual Network (ResNet) [18] considered by Gohr in [15], other neural networks that have shown advantages on ResNet in specific tasks, including Dense Network (DenseNet) [20] and the Squeeze-and-Excitation Network (SENet) [19], are investigated. Additionally, various training schemes, including direct training, key-averaging, and staged

training, were attempted. This effort results in various \mathcal{ND} s covering up to 11-round SIMON32/64.

- The full distribution of differences induced by the input difference (0x0000, 0x0040) up to 11 rounds are computed for SIMON32/64, which results in various \mathcal{DD} s. These \mathcal{DD} s provide solid baselines for \mathcal{ND} . We note that r -round \mathcal{ND} should be compared with $(r - 1)$ -round \mathcal{DD} for SIMON (different from SPECK). The results show that r -round \mathcal{ND} s achieve similar but weaker classification accuracy than $(r - 1)$ -round \mathcal{DD} s (see Table 5). We conjecture that r -round \mathcal{ND} s can learn to “decrypt” one un-keyed round and try to learn the distribution of the $(r - 1)$ -round differential, but fails to learn more features beyond the distribution of differences.

The source codes of the new attacks and the new \mathcal{ND} s can be found via https://github.com/differential-neural-cryptanalysis/speck32_simon32.

Table 1: Summary of key-recovery attacks on SPECK32/64 and SIMON32/64

Target	#R	Time (#Enc)	Data (#CP)	Succ. Rate	Weak keys	Configure	Ref.
SPECK32/64	11	2^{46}	2^{14}	-	2^{64}	1+6+4	[13]
		$2^{38}*$	$2^{13.6}$	0.52	2^{64}	1+2+7+1	[15]
	12	2^{51}	2^{19}	-	2^{64}	1+7+4	[13]
		$2^{43.40}*$	$2^{22.97}$	0.40	2^{64}	1+2+8+1	[15]
		$2^{44.89}*$	$2^{22.00}$	0.86	2^{64}	1+2+8+1	Fig. 6 Sect. 4.3
	13	$2^{42.97}*$	$2^{18.58}$	0.83	2^{63}	1+3+7+1	Fig. 7 Sect. 4.3
		2^{57}	2^{25}	-	2^{64}	1+8+4	[13]
		$2^{48.67} + r$	2^{29}	0.82	2^{63}	1+3+8+1	Fig. 5 Sect. 4.2
	14	$2^{62.47}$	$2^{30.47}$	-	2^{64}	1+9+4	[29]
	SIMON32/64	$2^{26.48}$	$2^{29.48}$	0.62	2^{64}	2+12+2	[3]
		$2^{41.81} + r$	2^{21}	0.49	2^{64}	1+3+11+1	Sect. E.4
		$2^{46.00}$	$2^{31.2}$	0.63	2^{64}	1+13+4	[2]
	21	$2^{55.25}$	$2^{31.0}$	-	2^{64}	4+13+4	[31]

- Not available.

* Under the assumption that one second equals the time of 2^{28} executions of SPECK32/64 or SIMON32/64 on a CPU.

$r : \log_2(cpu/gpu)$, where cpu and gpu are the CPU and GPU time running an attack, respectively. In our computing systems, $r = 2.4$ (The worse case execution time of the core of the 12-round attack on SPECK32/64 (without guessing the one key bit of k_0) took 6637 and 1265 seconds on CPU and GPU, respectively).

In the column entitled “Configure”, the numbers colored in blue and red are the numbers of round covered by \mathcal{CD} s and \mathcal{ND} s, respectively.

Please see [5] for the full version of this article.

ORGANIZATION. The rest of the paper is organized as follows. Section 2 gives the preliminary on ML-based differential cryptanalysis and introduces the design of SPECK and SIMON. Section 3 introduces concepts of generalized neutral bits and some new notice on differential trails of SPECK32/64. The framework of the enhanced differential-neural cryptanalysis and its applications to SPECK32/64 and SIMON32/64 are presented in Section 4 and Section E. Section 5 exhibits details of important statistics during the key-recovery phase. Rules of thumb

are provided for tuning various parameters for the attacks. Section 6 presents various of \mathcal{ND} s and \mathcal{DD} s on SIMON32/64 reduced up to 11 rounds.

2 Preliminary

2.1 Brief Description of SPECK32/64 and SIMON32/64

Notations. Denote by n the word size in bits, $2n$ the state size in bits. Denote by (x_r, y_r) the left and right branches of a state after the encryption of r rounds. Denote by $x[i]$ (resp. $y[i]$) the i -th bit of x (resp. y) counted starting from 0; Denote by $[j]$ the index of the j -th bit of the state, *i.e.*, the concatenation of x and y , where $y[0]$ is the 0-th bit, and $x[0]$ is the 16-th bit. Denote by \oplus the bit-wise XOR, \boxplus the addition modulo 2^n , $\&$ the bit-wise AND, $x^{\ll s}$ the bit-wise left rotation by s positions, $x^{\gg s}$ the bit-wise right rotation by s positions. Denote by F_k (resp. F_k^{-1}) the round function (resp. inverse of the round function) using subkey k of the encryption.

Brief Description of SPECK32/64 and SIMON32/64. SPECK32/64 and SIMON32/64 are small members of the lightweight block cipher families SPECK and SIMON [6] designed by researchers from the National Security Agency (NSA) of the USA. Both SPECK32/64 and SIMON32/64 have a Feistel-like structure⁵, a block size and a key size of 32 resp. 64 bits. The round functions use combinations of rotation, XOR, and addition modulo 2^{16} (SPECK) or bit-wise AND (SIMON). SPECK32/64 has 22 rounds, and SIMON32/64 has 32 rounds. The encryption algorithms of SPECK32/64 and SIMON32/64 are listed in Algorithms 1 and 2. The subkeys of 16-bit for each round are generated from a master key of 64-bit by the non-linear key schedule using the same round function (SPECK32/64) or linear functions of simple rotation and XOR (SIMON32/64).

Algorithm 1: Encryption of SPECK32/64

```

Input:  $P = (x_0, y_0), \{k_0, \dots, k_{21}\}$ 
Output:  $C = (x_{22}, y_{22})$ 
for  $r = 0$  to 21 do
     $x_{r+1} \leftarrow x_r^{\gg 7} \boxplus y_r \oplus k_r$ 
     $y_{r+1} \leftarrow y_r^{\ll 2} \oplus x_{r+1}$ 
end
```

Algorithm 2: Encryption of SIMON32/64

```

Input:  $P = (x_0, y_0), \{k_0, \dots, k_{31}\}$ 
Output:  $C = (x_{32}, y_{32})$ 
for  $r = 0$  to 31 do
     $x_{r+1} \leftarrow$ 
     $(x_r^{\ll 1} \& x_r^{\ll 8}) \oplus x_r^{\ll 2} \oplus y_r \oplus k_r$ 
     $y_{r+1} \leftarrow x_r$ 
end
```

2.2 Differential-based Neural Distinguishers

The work in [15] shows that a neural network could be trained to capture the non-randomness of the distribution of values of output pairs when the input

⁵ SPECK can be represented as a composition of two Feistel maps [6].

pairs to round-reduced SPECK32/64 are of specific difference, and thus play the role of distinguisher in cryptanalysis. This differential-based \mathcal{ND} is the first known machine learning model that successfully performed cryptanalysis tasks on modern ciphers (beyond the applications on side-channel attacks).

In the following, the way of training the differential-based \mathcal{ND} introduced in [15] is briefly recalled.

The Training Data and Input Representation. For a target cipher, the neural network is trained to distinguish between examples of ciphertext pairs corresponding to plaintext pairs with particular difference and those corresponding to random plaintext pairs. Thus, each of the training data is a data pair of the form (C, C') together with a label taking a value 0 or 1, where 0 means the corresponding plaintext pair is generated randomly, and 1 from a particular plaintext difference Δ_I . For SPECK32/64, the Δ_I is chosen to be of a single active bit, *i.e.*, $(0x0040, 0000)$, which is the intermediate difference lying in a known best differential characteristic.

The state of SPECK32/64 has left and right parts; thus, a pair of data is transformed into a quadruple of words (x, y, x', y') where $C = x \| y$ and $C' = x' \| y'$. The word quadruple is then interpreted into a 4×16 -matrix with each word as a row-vector before being fed into the neural network with an input layer consisting of 64 units. Among the training data (and verification data), half are positive and half are negative examples, labeled by 1 and 0, respectively.

Training Schemes. The neural network structure used in [15] is a deep residual network. There are three training schemes proposed in [15]. The first is a basic training scheme that is sufficient for successfully training short-round distinguishers. The second is an improved training scheme for r -round distinguishers that simulate the output of the KEY AVERAGING algorithm used with an $(r-1)$ -round distinguisher. Using the second scheme, the best \mathcal{ND} on 7-round SPECK32/64 was achieved in [15]. The third is a staged training method that turns an already trained $(r-1)$ -round distinguisher into an r -round distinguisher in several stages. Using the third scheme, the longest \mathcal{ND} on SPECK32/64, which is an 8-round one, was achieved.

2.3 Upper Confidence Bounds and Bayesian Optimization

Besides a basic key-recovery attack, an improved attack using specifics of the targeted cipher (*i.e.*, the wrong key randomization hypothesis does not hold when only one round of trial decryption is performed) and elements from reinforcement learning (*i.e.*, automatic exploitation versus exploration trade-off based on upper confidence bounds) was proposed in [15].

The improved key-recovery attack employs an r -round main and an $(r-1)$ -round helper \mathcal{ND} trained with data pairs corresponding to input pairs with difference Δ_I ; a short s -round differential, $\Delta_{I'} \rightarrow \Delta_I$ with probability denoted by 2^{-p} , is prepended on top of the \mathcal{ND} s (refer to Fig. 1 for an illustration of the

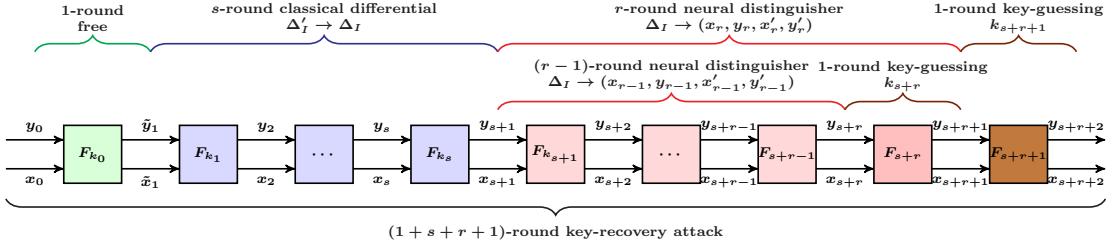


Fig. 1: Components of the key-recovery attacks

components of the key-recovery attack.) About $c \cdot 2^p$ (denoted by n_{cts}) data pairs with difference Δ_I' are randomly generated, where c is a small constant; Neutral bits of the s -round differential are used to expand each data pair to a structure of n_b data pairs. The resulting n_{cts} structures of data pairs are decrypted by one round with 0 as the subkey⁶ to get plaintext structures. All plaintext structures are queried to obtain the corresponding ciphertext structures.

Each ciphertext structure is to be used to generate candidates of the last subkey by the r -round main \mathcal{ND} (and latter of the second to the last subkey by the $(r-1)$ -round helper \mathcal{ND}) with a highly selective key search policy based on a variant of Bayesian optimization.

More specifically, the key search policy depends on an important observation that the expected response of the distinguisher upon wrong-key decryption will depend on the bit-wise difference between the trial key and the real key. This *wrong key response profile*, which can be precomputed, is used to recommend new candidate values for the key from previous candidate values by minimizing the weighted Euclidean distance as the criteria in an `BAYESIANKEYSEARCH` (see Alg. 4.) It recommends a set of subkeys and provides their scores without exhaustively performing trail decryptions.

The use of ciphertext structures is also highly selective using a standard exploration-exploitation technique, namely *Upper Confidence Bounds* (UCB). Each ciphertext structure is assigned a priority according to the scores of the subkeys they recommended and how often they were visited.

An important detail in the `BAYESIANKEYSEARCH` is that the responses $v_{i,k}$ from the \mathcal{ND} on ciphertext pairs in the ciphertext structure (of size n_b) are combined using the Formula 1 and used as the score s_k of the recommended subkey k (refer to Alg. 4). This score is highly decisive for the execution time and success rate of the attack. It will determine whether the recommended subkey will be further treated as its score passes or fails to pass the cutoff and also determine the priority of ciphertext structures to be visited. The number of ciphertext pairs in each structure is decisive when the \mathcal{ND} has low accuracy.

$$s_k := \sum_{i=0}^{n_b-1} \log_2\left(\frac{v_{i,k}}{1 - v_{i,k}}\right) \quad (1)$$

⁶ For SPECK, there is no whitening key and the first subkey is XORed after the first non-linear operation, which makes the first round free in differential attack (see the top of Fig. 3).

3 Deep Exploring of Neutral Bits

3.1 The Motivation of Neutral Bits

Typically, the more rounds a \mathcal{ND} covers, the lower its accuracy. When the accuracy becomes marginally higher than 0.5, it is hard to be used in a practical key-recovery attack. Thus, Gohr in [15] used the combined response (Formula 1) of the \mathcal{ND} over large number of samples of the same distribution as a distinguisher (named as combined-response-distinguisher, \mathcal{CRD}). By doing so, the signal from the \mathcal{ND} is boosted, and the distinguishability is increased. For a \mathcal{CRD} built on top of a weak \mathcal{ND} to reach its most potential with respect to distinguishability, the number of samples of the same distribution should be sufficiently large (see Sect. C for a detailed experimental study on the relation between the distinguishability of \mathcal{CRD} and the number of combined samples).

For the hybrid differential distinguisher used in the key-recovery attack in [15], it is not straightforward to aggregate enough samples of the same distribution fed to the \mathcal{ND} due to the prepended \mathcal{CD} . To overcome this problem, Gohr in [15] used the neutral bits of the \mathcal{CD} , which is a notion first introduced by Biham and Chen for attacking SHA-0 [9]. The neutral bit has many extensions and related concepts, including message modification [32], tunnels [22], boomerangs [21], probabilistic neutral bits [4], and free bits [23]. Changing the values at the neutral bits of an input pair does not change the conformability to the differential. Thus, one can use m neutral bits to derive 2^m data pairs from a single pair such that they conform or do not conform to the differential simultaneously. The more neutral bits there are for the prepended \mathcal{CD} , the more the samples of the same distribution that could be generated for the \mathcal{ND} . However, generally, the longer the \mathcal{CD} , the fewer the neutral bits.

Finding enough neutral bits for prepending a long \mathcal{CD} over a weak \mathcal{ND} becomes a difficult problem for devising a key-recovery to cover more rounds. Thus, the first part of this work focuses on finding various types of neutral bits.

3.2 Neutral Bits and Generalized Neutral Bits

Notations. Let $\delta := \Delta_{in} \rightarrow \Delta_{out}$ be a differential of an r -round encryption F^r . Let (P, P') be the input pair and (C, C') be the output pair, where $P \oplus P' = \Delta_{in}$, $C = F^r(P)$, and $C' = F^r(P')$. If $C \oplus C' = \Delta_{out}$, (P, P') is said to *conform* to the differential δ (conforming pairs, or correct pairs of the differential). The primary notion of neutral bits can be interpreted as follows.

Definition 1 (Neutral bits of a differential, NBs [9]). Let e_0, e_1, \dots, e_{n-1} be the standard basis of \mathbb{F}_2^n . Let i be an index of a bit (starting from 0). The i -th bit is a neutral bit of the differential $\Delta_{in} \rightarrow \Delta_{out}$, if for any conforming pair (P, P') , $(P \oplus e_i, P' \oplus e_i)$ is also a conforming pair.

Let $\{i_1, i_2, \dots, i_n\}$ be the set of NBs of a differential $\Delta_{in} \rightarrow \Delta_{out}$. Denote the subspace of \mathbb{F}_2^n with basis $\{e_{i_1}, e_{i_2}, \dots, e_{i_n}\}$ by \mathcal{S} . Then, from one input pair

(P, P') where $P \oplus P' = \Delta_{in}$, one can generate a set $\{(P_i, P'_i) \mid P_i \in P \oplus \mathcal{S}, P'_i = P_i \oplus \Delta_{in}\}$ that forms a structure with the same conformability to the differential.

For a differential $\Delta_{in} \rightarrow \Delta_{out}$ of F^r , in the view of a system of equations defined on the derivative function of F^r , i.e., $D_{\Delta_{in}} F^r(P) = \Delta_{out}$, a set of neutral bits \mathcal{NB} partitions the solution space of $D_{\Delta_{in}} F^r(x) = \Delta_{out}$ into equivalence classes. It can be seen that the more neutral bits for a differential, the more structured the solution space.

Generalization of Neutral Bits. In general, neutral bits of non-trivial differentials are scarce. In [15], because of the lack of neutral bits for the 2-round differential of SPECK32/64, probabilistic neutral bits (PNBs for short) are exploited. This notion of PNB has already been introduced by Aumasson *et al.* in previous differential cryptanalysis of stream ciphers Salsa20 and Chacha, and compression function Rumba [4]. Formally, it can be defined as follows.

Definition 2 (Probabilistic neutral bits, PNBs [4]). *Let i be an index of a bit. The i -th bit is a p -probabilistic neutral bit of the differential $\Delta_{in} \rightarrow \Delta_{out}$, if the event that when (P, P') conforms to the differential then $(P \oplus e_i, P' \oplus e_i)$ also conforms to the differential under the same key, has a probability p (over the choice of P and the key).*

In the sequel attacks, the higher the probability p is, the higher the neutrality quality, and the more useful the neutral bit becomes. For convenience, when $p = 1$, the neutral bits are said to be *deterministic neutral bits*.

In this work, two types of generalized neutral bits are considered beyond the (probabilistic) neutral bits considered in [15]. The first type, named simultaneous-neutral bit-set (SNBSs for short), has already been introduced together with the notion of neutral bit in [9]. That is, for a differential, given a conforming pair, complementing individual bits, the conformability might be changed, but simultaneously complementing a set of bits does not change the conformability of the resulted pair. Formally, it can be defined as follows.

Definition 3 (Simultaneous-neutral bit-sets, SNBSs [9]). *Let $I_s = \{i_1, i_2, \dots, i_s\}$ be a set of bit indices. Denote $f_{I_s} = \bigoplus_{i \in I_s} e_i$. The bit-set I_s is a simultaneous-neutral bit-set of the differential $\Delta_{in} \rightarrow \Delta_{out}$, if for any conforming pair (P, P') , $(P \oplus f_{I_s}, P' \oplus f_{I_s})$ is also a conforming pair, while for any subsets of I_s , the conformability of the resulted pair does not always hold.*

If we view that finding neutral bits is to form a subspace of \mathbb{F}_2^n in which the corresponding data have the same conformability to the differential, the essence of generalizing to SNBS is that, instead of only considering the standard basis corresponding to single-bit NBs, we now consider arbitrary bases.

The second type, which is a natural generalization, is named in this work as conditional (simultaneous-) neutral bit(-set)s (CSNBSs for short), that is, the bits or bit-sets are neutral for input pairs fulfilling specific conditions. Formally, it can be defined as follows.

Definition 4 (Conditional (simultaneous-) neutral bit(-set)s, CSNBSSs). let $I_s = \{i_1, i_2, \dots, i_s\}$ be a set of bit indices. Denote $f_{I_s} = \bigoplus_{i \in I_s} e_i$. Let \mathcal{C} be a set of constraints on the value of an input P , and \mathcal{P}_C be the set of inputs that fulfill the constraints \mathcal{C} . The bit-set I_s is a conditional simultaneous-neutral bit-set of the differential $\Delta_{in} \rightarrow \Delta_{out}$, if for any conforming pair (P, P') where $P \in \mathcal{P}_C$, $(P \oplus f_{I_s}, P' \oplus f_{I_s})$ is also a conforming pair.

The most straightforward constraints can be that some bit values of P are fixed. However, the constraints on the values of input P can be a more involved system of linear or non-linear equations.

Remark 1. Interestingly, various ‘tunnels’ have been used in [22] to speed up the search of MD5 collisions. They are essentially (generalized) neutral bits, including PNBs and CNBs. For consistency, in this paper, we use the extended names of the more well-known concept of ‘neutral bits’ instead of ‘tunnels’.

Remark 2. The neutrality of CSNBSSs depends on the values of some particular bits. The selected data is at an intermediate round in our attacks in this work, although the difference does not depend on the round-key, the values do. Thus, using CSNBSSs, the attack requires guessing some key bits of the first round.

3.3 Automatic Procedure to Search for CSNBSSs

To find CSNBSSs, we use an automatic procedure to experimentally evaluate the conditional neutral probability of candidate SNBSs. Concretely, we investigate how the neutrality of each candidate SNBS is influenced by values of bits in some involved and controllable variables (for SPECK32/64, such variables are supposed to be the variables involved in the first modular addition, particularly, they are x_1 , y_1 , and $x_1 \ggg^7 \oplus y_1$ ⁷), and search CSNBSSs conditioned on bits of these variables with the procedure in Algorithm 3.

3.4 Switching Bits for Adjoining Differentials

One knows that for a differential $\delta_1 = \Delta_{in_1} \rightarrow \Delta_{out}$, flipping a *non-neutral* bit of a conforming pair might make the resulting pair not conform to the differential. However, it is interesting that the resulted pair might turn into a conforming pair of another differential $\delta_2 = \Delta_{in_2} \rightarrow \Delta_{out}$ (after adjusting the input difference). If flipping this bit turns all conforming pairs of δ_1 into all conforming pairs of δ_2 , then δ_2 has the same probability as δ_1 . Since δ_1 and δ_2 have the same probability and share the same output difference, which will be the connecting difference in a hybrid distinguisher, the two differentials are equally useful. In this case, that non-neutral bit can play the same role as neutral bits for generating structures of pairs simultaneously satisfying the connecting difference in a hybrid distinguisher. Formally, we define such bits that relate two differentials as follows.

⁷ In these considered variables, $(x_1, y_1) = (\tilde{x}_1 \oplus k_0, \tilde{y}_1 \oplus k_0)$ is the real input to the \mathcal{CD} (see Fig. 3), where $(\tilde{x}_1, \tilde{y}_1)$ is the chosen data in the key-recovery attack (since, in the key-recovery attack, the \mathcal{CD} will be freely extended one round backward).

Algorithm 3: An automatic procedure to search for CSNBSs

1. Generate N random conforming pairs of the differential, each with a different random key.
 2. For each candidate SNBS, denoted by I , for each bit b of a variable x possibly influencing the neutrality, and for $c \in \{0, 1\}$, do the following.
 3. Experimentally evaluate the following probabilities over the conforming pairs.
 - $\Pr[I \text{ is neutral}]$, i.e., the neutral probability of I ;
 - $\Pr[b = c]$, i.e., the probability of b taking value c ;
 - $\Pr[I \text{ is neutral and } b = c]$, i.e., the probability of I is neutral and b taking value c for a random conforming pair;
 4. Compute $\Pr[I \text{ is neutral} \mid b = c]$ as $\frac{\Pr[I \text{ is neutral and } b = c]}{\Pr[b = c]}$ (when $\Pr[b = c] = 0$, set $\Pr[I \text{ is neutral} \mid b = c]$ as 0).
 5. If $\Pr[I \text{ is neutral} \mid b = c] - \Pr[I \text{ is neutral}] > \tau$ and $\Pr[I \text{ is neutral} \mid b = c] > \xi$, take I as a useful CSNBS and $b = c$ as its condition, and store in a set \mathcal{CNB} .
 - In our experiments, N takes 1000. Statistics using 10, 100, 5000 conforming pairs were also made as preliminary tests. The statistical results do not have obvious divergence when using more than 100 conforming pairs, thus 1000 should be sufficient.
 - In this procedure, τ and ξ are thresholds which can be adjusted to make trade-offs between the cost of imposing the condition, the number of CNBs, and the quality of CNBs. Typically, set τ be 0.2 and ξ be 0.8 will work well.
-

Definition 5 (Switching bits for adjoining differentials, SBfADs). Let i be an index of a bit. The i -th bit is an switching bit of two differentials $\delta_1 = \Delta_{in_1} \rightarrow \Delta_{out}$ and $\delta_2 = \Delta_{in_2} \rightarrow \Delta_{out}$, if for any conforming pair $(P, P \oplus \Delta_{in_1})$ of δ_1 , flipping the i -th bit and adjusting the input difference, the resulted pair $(P \oplus e_i, P \oplus e_i \oplus \Delta_{in_2})$ conforms to δ_2 under the same key. We call δ_1 and δ_2 adjoining differentials.

Conceivably, such adjoining differentials and switching bit should be rare. However, they do exist. Currently, we found one type for XOR (\oplus) differential of addition modulo 2^n (\boxplus), and the details can be found in Sect. A.1. A concrete example can be found in Sect. 4.1.

3.5 Paired Differentials Sharing the Same Neutral Bits

From the connecting difference between the \mathcal{CD} and the \mathcal{ND} propagating upward, there might be multiple differentials similar to \mathcal{CD} and have equally good probability. These similar differentials are likely to share many neutral bits. When a shared neutral bit happens to be exactly the difference between input differences of two differentials, one can re-group ciphertext pairs within each ciphertext structure corresponding to one differential, and obtain ciphertext structures corresponding to the other differential without additional queries, i.e., doubling the number of ciphertext structures for free. Formally, one has the following.

Definition 6 (Paired Differentials). let $\delta_1 = \Delta_{in_1} \rightarrow \Delta_{out}$ and $\delta_2 = \Delta_{in_2} \rightarrow \Delta_{out}$ be two differentials with the same output difference and with input differences satisfying $\Delta_{in_1} \oplus \Delta_{in_2} = \Delta_{nb_i}$. Suppose nb_i is a NB/SNBS for both δ_1 and δ_2 . Then, once a pair of input pair $\{(P, P \oplus \Delta_{in_1}), (P \oplus \Delta_{nb_i}, P \oplus \Delta_{in_1} \oplus \Delta_{nb_i})\}$

is generated for differential δ_1 , one can re-pair the inputs as $\{(P, P \oplus \Delta_{in_1} \oplus \Delta_{nb_i}), (P \oplus \Delta_{nb_i}, P \oplus \Delta_{in_1})\}$ and obtain a pair of input pair for differential δ_2 . Thus, by re-pairing the corresponding ciphertext pairs, the number of ciphertext structures is doubled. Such two differentials are said to be paired differentials.

Exploiting paired differentials can reduce the data complexity by half, but is only of interest when the two differentials are with almost equally good probability and share enough neutral bits to be used in key-recovery attacks. An example can be found in Sect. 4.1.

Remark 3. Noticeably, the example of paired differentials is exactly the example of adjoining differentials in Sect. 4.1. However, this same example plays different roles when employed as paired differentials or as adjoining differentials. Employing as the former is to reuse data by re-pairing, employing as the latter is to achieve the effect of neutral bits. If a single pair of differentials acts as paired differentials and adjoining differentials simultaneously, the generated data *pairs* will be all different. Thus, two differentials can play both roles at the same time.

Remark 4. Reusing data to form different pairs adds dependencies between the chosen data pairs. However, the influence of such dependencies should not matter. We performed the attacks with and without reusing the data. The results show that as long as the total number of ciphertext structures and their size are the same, the success rates are roughly the same.

Remark 5. There is an implicit relation between neutral bits of a differential and high-order differential. An SNBS I_s of a differential $\Delta_{in} \rightarrow \Delta_{out}$ defines a special high-order differential $\Delta_{a_1, a_2} \rightarrow 0$, where $a_1 = \Delta_{in}$ and $a_2 = \bigoplus_{i \in I_s} e_i$.

Besides, there is an interesting relation between neutral bits and the mixture-differential distinguisher of AES [17]. Some neutral bits found for SPECK32/64 and SIMON32/64 in this work can result in some bit-level mixture quadruples.

4 Key Recovery Attack on Round-Reduced SPECK32/64

This section shows that the neural distinguishers have not reached their full potential in the key-recovery attacks in [15]. They could be harnessed to cooperate with classical cryptanalytic tools and perform key-recovery attacks competitive to the attacks devised by orthodox cryptanalysis. In the following, we present key-recovery attacks employing the same neural distinguishers used in the 11-round and 12-round attacks on SPECK32/64 in [15]. The first neural distinguishers based 13-round attack and an improved 12-round attack were obtained.

The improved attacks follow the framework of the improved key-recovery attacks in [15]. An r -round main and an $(r - 1)$ -round helper \mathcal{ND} s are employed, and an s -round CD is prepended. The key guessing procedure applies a simple reinforcement learning procedure. The last subkey and the second to last subkey are to be recovered without exhaustively using all candidate values to do one-round decryption. Instead, a Bayesian key search employing the wrong key response profile is to be used.

The prepended \mathcal{CD} s to be used in the improved attacks include the same 2-round differential used in the attack in [15] and four new 3-round differentials. The preliminary is to find enough NBs of these differentials to obtain enough samples of the same distribution so that we can use the combined response from the \mathcal{ND} s. In the following, the SNBSs, CNBs, and SBfADs introduced in Sect. 3 are to be found and exploited.

4.1 Finding CSNBSSs for SPECK32/64

For finding NBs of the differential of round-reduced SPECK32/64, we used an exhaustive search for empirical results because of the complexity brought by the carry of modular addition.

Finding SNBSs for 2-round Differential. For the prepended 2-round \mathcal{CD} on top of the \mathcal{ND} s, one can experimentally obtain three deterministic NBs and two SNBSs (simultaneously complementing up to 4 bits) using an exhaustive search. Besides, bits and bit-sets that are (simultaneous-)neutral with high probabilities are also detected. Concretely, for the 2-round differential $(0x0211, 0xa04) \rightarrow (0x0040, 0x0000)$, bits and bit-sets that are (probabilistic) (simultaneous-)neutral are summarized in Table 2.

Table 2: (Probabilistic) SNBSs for 2-round differential $(0x0211, 0xa04) \rightarrow (0x0040, 0x0000)$ of SPECK32/64. The statistics were performed on 1000 correct pairs, each with a different random key. For comparison, one can find the NBs used by attacks in [15] in Table 9.

NBs	Pr.		NBs	Pr.		NBs	Pr.		NBs	Pr.		NBs	Pr.						
[20]	1		[21]	1		[22]	1		[9, 16]	1		[2, 11, 25]	1						
[6, 29]	0.91		[23]	0.812		[30]	0.809		[7]	0.806		[0]	0.754		[14]	0.965		[15]	0.938
															[11, 27]	0.736		[8]	0.664

Finding SNBSs for 3-round Differential. The 2-round differential $(0x0211, 0xa04) \rightarrow (0x0040, 0x0000)$ can be extended to two optimal (prob. $\approx 2^{-11}$) 3-round differentials, *i.e.*, $(0xa20, 0x4205) \rightarrow (0x0040, 0x0000)$, $(0xa60, 0x4205) \rightarrow (0x0040, 0x0000)$.

However, the NBs/SNBSs of these two optimal differentials are very scarce. There are four sub-optimal 3-round differentials (prob. $\approx 2^{-12}$ when being estimated following Markov model, but are actually 2^{-11} for 2^{63} keys and 0 for another 2^{63} keys, see Sect. D for more details), *i.e.*,

$$(0x8020, 0x4101) \rightarrow (0x0040, 0x0000), (0x8060, 0x4101) \rightarrow (0x0040, 0x0000), \\ (0x8021, 0x4101) \rightarrow (0x0040, 0x0000), (0x8061, 0x4101) \rightarrow (0x0040, 0x0000).$$

For these 3-round differentials, the hamming weights of the input differences are low, and they have more NBs/SNBSs. Still, the numbers of NBs/SNBSs are not

enough for appending a weak neural network distinguisher. Thus, conditional ones were searched using the procedure in Algorithm 3. For $\xi = 0.7$, the obtained CSNBSs and their conditions are summarized together with unconditional NBs/SNBSs in Table 4. In the table, the columns titled ‘Post.’ are finally verified neutral probabilities of the (C)SNBSs when all four conditions are fulfilled.

For each of the four differentials, there are three linear conditions (xy-type) that are necessary for a pair $((x, y), (x', y'))$ to conform to it, which are listed in Table 3 (without coloring in gray). For each linear condition, once it is fulfilled, the probability of the differential increases by a factor of 2^1 . In the following key-recovery attacks, the linear conditions can be fulfilled by chosen data once the corresponding bits of k_0 are guessed.

Table 3: Necessary conditions to conform to the 3-round differentials (or the dominant trail in the differential, intermediate differences in the dominant trail are colored in gray, the condition for the trail instead of the differential is colored in gray, where $c = (x \gg 7 \oplus y) \oplus (x \gg 7 \oplus y)$). Each column corresponds to one differential (trail).)

(0x8020, 0x4101)	(0x8060, 0x4101)	(0x8021, 0x4101)	(0x8061, 0x4101)
(0x0201, 0x0604)	(0x0201, 0x0604)	(0x0201, 0x0604)	(0x0201, 0x0604)
(0x1800, 0x0010)	(0x1800, 0x0010)	(0x1800, 0x0010)	(0x1800, 0x0010)
(0x0040, 0x0000)	(0x0040, 0x0000)	(0x0040, 0x0000)	(0x0040, 0x0000)
$\begin{cases} x[7] = 0, \\ x[5] \oplus y[14] = 1, \\ x[15] \oplus y[8] = 0, \\ x[0] \oplus y[9] = 0. \end{cases}$	$\begin{cases} x[7] = 0, \\ x[5] \oplus y[14] = 0, \\ x[15] \oplus y[8] = 0, \\ x[0] \oplus y[9] = 0. \end{cases}$	$\begin{cases} x[7] = 0, \\ x[5] \oplus y[14] = 1, \\ x[15] \oplus y[8] = 1, \\ y[9] \oplus c[9] = 0. \end{cases}$	$\begin{cases} x[7] = 0, \\ x[5] \oplus y[14] = 0, \\ x[15] \oplus y[8] = 1, \\ y[9] \oplus c[9] = 0. \end{cases}$

Table 4: (C)SNBSs for 3-round differential $(0x8020, 0x4101) \rightarrow (0x0040, 0x0000)$, $(0x8060, 0x4101) \rightarrow (0x0040, 0x0000)$, $(0x8021, 0x4101) \rightarrow (0x0040, 0x0000)$, and $(0x8061, 0x4101) \rightarrow (0x0040, 0x0000)$ of SPECK32/64.

Bit-set	(8020, 4101)		(8060, 4101)		(8021, 4101)		(8061, 4101)		Condition
	Pre.	Post.	Pre.	Post.	Pre.	Post.	Pre.	Post.	
[22]	0.995	1.000	0.995	1.000	0.996	1.000	0.997	1.000	–
[20]	0.986	1.000	0.997	1.000	0.996	1.000	0.995	1.000	–
[13]	0.986	1.000	0.989	1.000	0.988	1.000	0.992	1.000	–
[12, 19]	0.986	1.000	0.995	1.000	0.993	1.000	0.986	1.000	–
[14, 21]	0.855	0.860	0.874	0.871	0.881	0.873	0.881	0.876	–
[6, 29]	0.901	0.902	0.898	0.893	0.721	0.706	0.721	0.723	–
[30]	0.803	0.818	0.818	0.860	0.442	0.442	0.412	0.407	–
[0, 8, 31]	0.855	0.859	0.858	0.881	0.000	0.000	0.000	0.000	–
[5, 28]	0.495	1.000	0.495	1.000	0.481	1.000	0.469	1.000	$x[12] \oplus y[5] = 1$
[15, 24]	0.482	1.000	0.542	1.000	0.498	1.000	0.496	1.000	$y[1] = 0$
[4, 27, 29]	0.672	0.916	0.648	0.905	0.535	0.736	0.536	0.718	$x[11] \oplus y[4] = 1$
[6, 11, 12, 18]	0.445	0.903	0.456	0.906	0.333	0.701	0.382	0.726	$x[2] \oplus y[11] = 0$

A condition at the end of a row is specific to the bit-set at the same row. ‘–’ means that there is no condition for the corresponding bit-set.

Pre.: probability obtained using 1000 correct pairs without imposing the conditions.

Post.: probability obtained using 1000 correct pairs and imposing all conditions in the last column.

□: Neutral bit(-set)s used in the 13-round attack $\mathcal{A}_{\text{SPECK13R}}$ on SPECK32/64.

□: Neutral bit(-set)s used in the 12-round attack $\mathcal{A}_{\text{SPECK12R}}$ on SPECK32/64.

Exploiting SBfADs. Among the four 3-round differentials, $(0x8020, 0x4101) \rightarrow (0x0040, 0x0000)$ and $(0x8060, 0x4101) \rightarrow (0x0040, 0x0000)$ are adjoining differentials, and $(0x8021, 0x4101) \rightarrow (0x0040, 0x0000)$ and $(0x8061, 0x4101) \rightarrow (0x0040, 0x0000)$ are adjoining differentials (refer to Sect. 3.4). The bit 5 of x (the bit 21 of $x||y$) is the SBfAD of both pairs. An SBfAD plays the same role as a deterministic unconditional NB, thus is better to be used than probabilistic and conditional NBs. Specifically, employing SBfAD saves one guessed key bit and reduces both time and data complexity by half compared to employing the CSNBS. In the presented 13-round (resp. 12-round) attacks, this SBfAD is employed, and one CSNBS (resp. PNBS) in Table 4 can be dismissed.

The reasoning on why one can switch between these differentials by bit 5 of x can be found in Sect. A.2. Experiments were performed and have verified that this SBfAD plays a better role than a CSNBS or a PNBS.

Exploiting paired differentials. The four 3-round differentials share most of the high-probabilistic NBs and the conditions on the NBs. Besides, the neutral bit [22] makes $(0x8020, 0x4101) \rightarrow (0x0040, 0x0000)$ and $(0x8060, 0x4101) \rightarrow (0x0040, 0x0000)$ (resp. $(0x8021, 0x4101) \rightarrow (0x0040, 0x0000)$ and $(0x8061, 0x4101) \rightarrow (0x0040, 0x0000)$) be paired differentials as introduced in Sect 3.5.

Specifically, take the first two differentials for example. They share the neutral bit [22] and all other useful NB. Since $(0x8020, 0x4101) \oplus (0x8060, 0x4101) = (0x0040, 0000)$, while bit [22] corresponds to difference $\Delta_{22} = (0x0040, 0000)$, ciphertext structures for $(0x8060, 0x4101) \rightarrow (0x0040, 0x0000)$ can be directly obtained from that of $(0x8020, 0x4101) \rightarrow (0x0040, 0x0000)$ (refer to Sect. 3.5). Thus, using a paired differentials (as in the following attack $\mathcal{A}^{\text{SPECK}_{13R}}$ on the 13-round SPECK32/64), one can generate half of the required data pairs for free. Accordingly, the data complexity to get one pair of ciphertexts is one instead of two.

Further, the data complexity can be slightly reduced by using both paired differentials when the attack requires no more than six NBs (the number of shared unconditional NBs). For the ease of notation, let us denote $(0x8020, 0x4101)$ as example difference Δ_E^1 , and $(0x8021, 0x4101)$ as Δ_E^2 . Six queries of a plain-text structure consisting of $(P, P \oplus \Delta_E^1, P \oplus \Delta_{22}, P \oplus \Delta_E^1 \oplus \Delta_{22}, P \oplus \Delta_E^2, P \oplus \Delta_E^2 \oplus \Delta_{22})$ result in eight pairs to be used in the upcoming attack $\mathcal{A}^{\text{SPECK}_{12R}}$ on the 12-round SPECK32/64. The eight pairs are two pairs $(P, P \oplus \Delta_E^1)$ and $(P \oplus \Delta_{22}, P \oplus \Delta_E^1 \oplus \Delta_{22})$ following input difference Δ_E^1 , two pairs $(P, P \oplus \Delta_E^1 \oplus \Delta_{22})$, $(P \oplus \Delta_{22}, P \oplus \Delta_E^1)$ following input difference $\Delta_E^1 \oplus \Delta_{22}$, two pairs $(P, P \oplus \Delta_E^2)$, $(P \oplus \Delta_{22}, P \oplus \Delta_E^2 \oplus \Delta_{22})$ following input difference Δ_E^2 , and two pairs $(P, P \oplus \Delta_E^2 \oplus \Delta_{22})$, $(P \oplus \Delta_{22}, P \oplus \Delta_E^2)$ following input difference $\Delta_E^2 \oplus \Delta_{22}$. In such a way, the average data complexity to get one pair of ciphertexts reduces from 2 to 3/4.

4.2 Key Recovery Attack on 13-round SPECK32/64

Employing two classical differentials that can simultaneously act as adjoining differentials and paired differentials, and combining them with neural distinguishers, we examine how far a practical attack can go on reduced-round SPECK32/64. A 13-round attack, denoted by $\mathcal{A}^{\text{SPECK}_{13R}}$, is devised as follows.

The preliminary components that capture characteristics of SPECK32/64 for devising the attack $\mathcal{A}^{\text{SPECK}_{13R}}$ are as follows.

1. Two 3-round \mathcal{CD} s $(0x8020, 0x4101) \rightarrow (0x0040, 0x0000)$, $(0x8060, 0x4101) \rightarrow (0x0040, 0x0000)$ (refer to the rounds colored in blue in Fig. 3), which act as both adjoining differentials and paired differentials (refer to Remark 3), 11 common NBs (including single-bit NBs, SNBSs, CSNBSSs), i.e., $\mathcal{NB}: \{[22], [13], [20], [5, 28], [15, 24], [12, 19], [6, 29], [4, 27, 29], [14, 21], [0, 8, 31], [30]\}$ (refer to the columns framed by blue lines in Table 4), and a SBfAD [21];
2. An 8-round \mathcal{ND} , named $\mathcal{ND}^{\text{SPECK}_{8R}}$, trained with difference $(0x0040, 0x0000)$ and its wrong key response profiles $\mathcal{ND}^{\text{SPECK}_{8R}}.\mu$ and $\mathcal{ND}^{\text{SPECK}_{8R}}.\sigma$;
3. A 7-round \mathcal{ND} , named $\mathcal{ND}^{\text{SPECK}_{7R}}$, trained with difference $(0x0040, 0x0000)$ and its wrong key response profiles $\mathcal{ND}^{\text{SPECK}_{7R}}.\mu$ and $\mathcal{ND}^{\text{SPECK}_{7R}}.\sigma$.

The parameters for recovering the last two subkeys are denoted as follows.

1. n_{kg} : the number of possible values for the few guessed bits of k_0 .
2. n_{cts} : the number of ciphertext structures.
3. n_b : the number of ciphertext pairs in each ciphertext structure, i.e., $2^{|\mathcal{NB}|+1}$.
4. n_{it} : the total number of iterations on the ciphertext structures.
5. c_1 and c_2 : the cutoffs with respect to the scores of the recommended last subkey and second to last subkey, respectively.
6. n_{byit1}, n_{cand1} and n_{byit2}, n_{cand2} : the number of iterations and number of key candidates within each iteration in the BAYESIANKEYSEARCH procedures (refer to Alg. 4) for guessing each of the last and second to last subkeys.

The attack procedure is as follows (refer to Figures 2 and 3).

1. Initialize variables $G_{best_key} \leftarrow (\text{None}, \text{None})$, $G_{best_score} \leftarrow -\infty$.
2. For each of the n_{kg} values of the 5 key bits $k_0[7], k_0[15] \oplus k_0[8], k_0[12] \oplus k_0[5], k_0[1], k_0[11] \oplus k_0[4]$ ⁸,
 - (a) Generate $n_{cts}/2$ random data pairs, i.e., $(\tilde{x}_1 || \tilde{y}_1, \tilde{x}'_1 || \tilde{y}'_1)$'s, with difference $(0x8020, 0x4101)$, and satisfying the conditions for conforming pairs, i.e., $\begin{cases} \tilde{x}_1[7] = k_0[7], \\ \tilde{x}_1[15] \oplus \tilde{y}_1[8] = k_0[15] \oplus k_0[8], \end{cases}$ and the conditions for three CSNBSSs $\begin{cases} \tilde{x}_1[12] \oplus \tilde{y}_1[5] \oplus 1 = k_0[12] \oplus k_0[5], \\ \tilde{y}_1[1] = k_0[1], \\ \tilde{x}_1[11] \oplus \tilde{y}_1[4] \oplus 1 = k_0[11] \oplus k_0[4], \end{cases}$ (refer to Tables 3 and 4).

⁸ Since the first two 3-round \mathcal{CD} s are used as paired differentials, the key bit $k_0[5] \oplus k_0[14]$ does not need to be guessed. Besides, since the CSNBS [6, 11, 12, 18] in Table 4 is not used in the attack, the key bit $k_0[2] \oplus k_0[11]$ does not need to be guessed. In total only 5 bits of k_0 are guessed

- (b) From the $n_{cts}/2$ random data pairs, generate $n_{cts}/2$ structures using the NBs in \mathcal{NB} , marking the correspondence between old pairs and new pairs that are generated using the NB [22].
- (c) Use the SBfAD [21] to double the number of pairs in each of the $n_{cts}/2$ structures. The new pairs are generated by flipping the [21] bit in the original pairs and adjusting the difference to be (0x8060, 0x4101).
- (d) Decrypt one round using zero as the subkey for all data in the structures obtained above and obtain $n_{cts}/2$ plaintext structures;
- (e) Query for the ciphertexts under 13-round SPECK32/64 of the $n_{cts}/2 \times n_b \times 2$ plaintexts, obtaining $n_{cts}/2$ ciphertext structures.
- (f) For each couple of ciphertext pairs, denoted by (c_1, c'_1) and (c_2, c'_2) , whose corresponding couple of data pairs are related by flipping the neutral bit [22], that is the couple $(\tilde{x}_1 \parallel \tilde{y}_1, \tilde{x}_1 \parallel \tilde{y}_1 \oplus (0x8020, 0x4101))$ and $(\tilde{x}_1 \parallel \tilde{y}_1 \oplus (0x0040, 0000), \tilde{x}_1 \parallel \tilde{y}_1 \oplus (0x8020, 0x4101) \oplus (0x0040, 0000))$, obtain a new couple of ciphertext pairs, that is (c_1, c'_2) and (c_2, c'_1) . As a result, the new couples generated in this way correspond to couples of plaintext pairs for the second differential (0x8060, 0x4101) and its neutral bit [22]. Thus, additional $n_{cts}/2$ ciphertext structures can be obtained without new queries. In total, n_{cts} ciphertext structures, denoted by $\{\mathcal{C}_1, \dots, \mathcal{C}_{n_{cts}}\}$, are obtained.
- (g) Initialize an array w_{\max} and an array n_{visit} to record the highest scores and the numbers of visits obtained by ciphertext structures.
- (h) Initialize variables $best_{\text{score}} \leftarrow -\infty$, $best_{\text{key}} \leftarrow (\text{None}, \text{None})$, $best_{\text{pos}} \leftarrow \text{None}$ to record the best score, the corresponding best-recommended values for the two subkeys obtained among all ciphertext structures and the index of this ciphertext structure.
- (i) For j from 1 to n_{it} :
 - i. Compute the priority of each of the ciphertext structures as follows:
 $s_i = w_{\max,i} + \alpha \cdot \sqrt{\log_2(j)/n_{\text{visit},i}}$, for $i \in \{1, \dots, n_{cts}\}$, and $\alpha = \sqrt{n_{cts}}$; This formula of priority is designed according to a general method in reinforcement learning for achieving automatic exploitation versus exploration trade-off based on Upper Confidence Bounds. It is motivated to focus the key search on the most promising ciphertext structures [15].
 - ii. Pick the ciphertext structure with the highest priority score for further processing in this j -th iteration, denote it by \mathcal{C} , and its index by idx , $n_{\text{visit},idx} \leftarrow n_{\text{visit},idx} + 1$.
 - iii. Run BAYESIANKEYSEARCH with \mathcal{C} , the neural distinguisher $\mathcal{ND}^{\text{SPECK}_{8R}}$ and its wrong key response profile $\mathcal{ND}^{\text{SPECK}_{8R}}.\mu$ and $\mathcal{ND}^{\text{SPECK}_{8R}}.\sigma$, n_{cand1} , and n_{byit1} as input parameters; obtain the output, that is a list L_1 of $n_{byit1} \times n_{cand1}$ candidate values for the last subkey and their scores, i.e., $L_1 = \{(g_{1i}, v_{1i}) : i \in \{1, \dots, n_{byit1} \times n_{cand1}\}\}$.
 - iv. Find the maximum $v_{1\max}$ among v_{1i} in L_1 , if $v_{1\max} > w_{\max,idx}$, $w_{\max,idx} \leftarrow v_{1\max}$.
 - v. For each recommended last subkey $g_{1i} \in L_1$, if the score $v_{1i} > c_1$,

- A. Decrypt the ciphertexts in \mathcal{C} using the g_{1i} by one round and obtain the ciphertext structure \mathcal{C}' of 12-round SPECK32/64.
- B. Run BAYESIANKEYSEARCH with \mathcal{C}' , $\mathcal{ND}^{\text{SPECK7R}}$ and its wrong key response profile $\mathcal{ND}^{\text{SPECK7R}}.\mu$ and $\mathcal{ND}^{\text{SPECK7R}}.\sigma$, n_{cand2} , and n_{byit2} as input parameters; obtain the output, that is a list L_2 of $n_{byit2} \times n_{cand2}$ candidate values for the second to last subkey and their scores, i.e., $L_2 = \{(g_{2i}, v_{2i}) : i \in \{1, \dots, n_{byit2} \times n_{cand2}\}\}$.
- C. Find the maximum among v_{2i} and the corresponding g_{2i} in L_2 , and denote them by $v_{2\max}$ and $g_{2\max}$.
- D. If $v_{2\max} > best_{\text{score}}$, update
 $best_{\text{score}} \leftarrow v_{2\max}$, $best_{\text{key}} \leftarrow (g_{1i}, g_{2\max})$, $best_{\text{pos}} \leftarrow idx$.
- vi. If $best_{\text{score}} > c_2$, go to Step 2j.
- (j) Make a final improvement using VERIFIERSEARCH [14] on the value of $best_{\text{key}}$ by examining whether the scores of a set of keys obtained by changing at most 2 bits on top of the incrementally updated $best_{\text{key}}$ could be improved recursively until no improvement is obtained, update $best_{\text{score}}$ to the best score in the final improvement; If $best_{\text{score}} > Gbest_{\text{score}}$, update $Gbest_{\text{score}} \leftarrow best_{\text{score}}$, $Gbest_{\text{key}} \leftarrow best_{\text{key}}$.
- 3. Return $Gbest_{\text{key}}$, $Gbest_{\text{score}}$.

Remark 6. In Gohr's implementations of the attack [14], two bits of g_1 are randomly assigned instead of being recommended by minimizing the weighted euclidean distance. This is based on observation of the symmetry of the wrong key response profiles, which indicates that values of the last two bits of the last subkey have almost the same influence on the response, thus hard to be correctly guessed. In our implementations, guessing these two bits in the last subkey is integrated into guessing the second to the last subkey, which is done using the stronger helper \mathcal{ND} . The wrong key response profile with respect to the helper \mathcal{ND} are thus on 18 key bits. In doing so, these two key bits can be correctly recommended with a higher probability.

In the experimental verification of the attack $\mathcal{A}^{\text{SPECK13R}}$, the 8-round and 7-round neural distinguishers provided in [14] were used. The accuracy of $\mathcal{ND}^{\text{SPECK8R}}$ (resp. $\mathcal{ND}^{\text{SPECK7R}}$) is about 0.514 (resp. 0.616). Concrete parameters and the complexity of $\mathcal{A}^{\text{SPECK13R}}$ are as follows (see Figure 5).

$$\begin{aligned} n_{kg} &= 2^5, & n_b &= 2^{11+1}, & n_{cts} &= 2^{12}, & n_{it} &= 4 \times n_{cts} \\ c_1 &= 18, & c_2 &= -500, & n_{byit1} &= n_{byit2} = 5, & n_{cand1} &= n_{cand2} = 32 \end{aligned}$$

The data complexity is $n_{kg} \times n_b \times n_{cts}$, that is, $2^{5+11+1+12}$, i.e., 2^{29} plaintexts (because of the use of two matched differentials, data complexity for getting each ciphertext pair is 1 instead of 2.)

To make the experimental verification economic, we tested the core of the attack with the five conditions being fulfilled only. That is, tested whether a particular one of $2^{n_{kg}}$ loops in Step 2 can successfully recover the last two subkeys. In that particular loop, the trialed value of the 5 bits of k_0 is correct. In the other loops, the trialed values deviate from the correct value by at least one bit. The

other loops can be expected to obtain worse scores and wrong key guesses than that particular loop. Besides, since the prepended classical differentials are valid to keys fulfilling $k_2[12] \neq k_2[11]$, we tested for these valid keys only, and the presented attack works for 2^{63} keys (refer to Sect. D).

The core of the attack was examined in 40 trials. We count a key guess as successful if the returned last two subkeys and the real two subkeys have a Hamming distance at most two in total. Among the 40 trials, there are 33 successful trials. Thus, the success rate is 33/40, which is 0.8250.

The trials were executed using a server with 8 GPUs⁹. The maximum execution time (worst-case run time) among the 40 runs is 14.5 hours (which runs all the n_{it} , *i.e.*, 2^{14} iterations). For 2^5 loops in Step 2, the worst situation is that within each loop, all n_{it} iterations are executed. Accordingly, the full attack requires about $2^5 \times 14.5$, *i.e.*, 464 GPU hours, which is equivalent to $2^{48.67+r}$ executions of SPECK32/64¹⁰.

Remark 7. For invalid guesses of the few bits of k_0 , worse scores and wrong key guesses for the last two subkeys will be obtained. Invalidated guesses of bits of k_0 directly cause all or most ciphertext pairs in all ciphertext structures to be nonconforming pairs (wrong ciphertext structures). For wrong ciphertext structures, the scores of the recommended last and the second to last subkeys will be very low such that fewer last subkeys will pass cutoff c_1 , and almost no second to the last subkey will pass cutoff c_2 . Therefore, under invalid key guesses of k_0 , all n_{it} iterations will be used. Using n_{kg} times the worst-case run-time (which is taken by a failed trial using all the n_{it} iterations) of an attack core provides a conservative estimation of the time complexity of a full attack.

4.3 Key Recovery Attack on 12-round Speck32/64

To devise key-recovery attack on 12-round SPECK32/64, Gohr in [15] used the 2-round classical differential $(0x0211, 0x0a04) \rightarrow (0x0040, 0x0000)$ combined with the 8-round and 7-round \mathcal{ND} s. For amplifying the weak signal from the 8-round neural distinguisher, 13 single-bit NBs of the prepended 2-round \mathcal{CD} were exploited. However, many of the 13 NBs are neutral with probabilities that are not high (refer to Table 9). Besides, 500 ciphertext structures and 2000 iterations were used to achieve a success rate of 0.40. Thus, the data complexity is $500 \times 2^{13} \times 2$, *i.e.*, $2^{22.97}$ plaintexts. The attack takes roughly 12 hours on a quad-core PC (as listed in Table 1).

From Table 2, one can see that there are many SNBSSs being deterministically neutral or neutral with relatively high probability. Using 13 SNBSSs, cutting the required data by nearly half, and using the following parameters, our experiments

⁹ Tesla V100-SXM2-32GB, computeCapability: 7.0; coreClock: 1.53GHz; coreCount: 80; deviceMemorySize: 31.72GB; deviceMemoryBandwidth: 836.37GB/s)

¹⁰ Under the assumption that one second equals the time of 2^{28} executions of SPECK32/64 on a CPU, and $r = \log_2(cpu/gpu)$, where cpu is the CPU time and gpu is the GPU time running an attack. In our computing systems, $r = 2.4$

show that the success rate of the resulting attack can be increased to 0.86 using fewer data (see Fig. 6).

$$\begin{aligned} n_{kg} &= 0, & n_b &= 2^{13}, & n_{cts} &= 2^8, & n_{it} &= 2^{10} \\ c_1 &= 15, & c_2 &= 500, & n_{byit1} &= n_{byit2} = 5, & n_{cand1} &= n_{cand2} = 32 \end{aligned}$$

However, the data complexity is still bounded by the weakness of the 8-round \mathcal{ND} . To further reduce the data requirement, we employ the 3-round \mathcal{CDs} and combine them with the stronger 7-round (and 6-round) \mathcal{ND} . In this case, unconditional SNSBs are enough for the 7-round \mathcal{ND} . Thus, those conditional ones can be dismissed in such a 12-round attack. Besides, since bit [21] is an SBfAD which switches the first two and the last two differentials, it can be used to replace a probabilistic NB. The four 3-round differentials share enough NBs, thus, all can be employed, which makes it possible to obtain one plaintext pair with 3/4 instead of 2 queries (as discussed in Sect. 4.1).

Concretely, the components of the 12-round key-recovery attack, denoted by $\mathcal{A}^{\text{SPECK}_{12R}}$, are as follows.

1. Four 3-round \mathcal{CDs} $(0x8020, 0x4101) \rightarrow (0x0040, 0x0000), (0x8060, 0x4101) \rightarrow (0x0040, 0x0000), (0x8021, 0x4101) \rightarrow (0x0040, 0x0000), (0x8061, 0x4101) \rightarrow (0x0040, 0x0000)$, five neutral bit(-set)s \mathcal{NB} : $\{[22], [13], [20], [12, 19], [14, 21]\}$ (refer to the rows framed by green lines in Table 4), one SBfAD [21];
2. A 7-round \mathcal{ND} , named $\mathcal{ND}^{\text{SPECK}_{7R}}$, trained with difference $(0x0040, 0x0000)$ and its wrong key response profiles $\mathcal{ND}^{\text{SPECK}_{7R}}.\mu$ and $\mathcal{ND}^{\text{SPECK}_{7R}}.\sigma$;
3. A 6-round \mathcal{ND} , named $\mathcal{ND}^{\text{SPECK}_{6R}}$ trained with difference $(0x0040, 0x0000)$ and its wrong key response profiles $\mathcal{ND}^{\text{SPECK}_{6R}}.\mu$ and $\mathcal{ND}^{\text{SPECK}_{6R}}.\sigma$.

The framework of the 12-round attack $\mathcal{A}^{\text{SPECK}_{12R}}$ follows that of $\mathcal{A}^{\text{SPECK}_{13R}}$. The difference is that, in the beginning, we only guess one key bit of k_0 , that is $k_0[7]$, because for all four 3-round differentials, there is only one common condition for conforming pairs, *i.e.*, $x_1[7] = 0$ (refer to Table 3). Thus, n_{kg} is 2^1 , and there are only 2 outermost loops.

The concrete parameters and complexity of $\mathcal{A}^{\text{SPECK}_{12R}}$ are as follows (see Fig. 7 for details). The accuracy of $\mathcal{ND}^{\text{SPECK}_{7R}}$ (resp. $\mathcal{ND}^{\text{SPECK}_{6R}}$) is about 0.616 (resp. 0.788).

$$\begin{aligned} n_{kg} &= 2^1, & n_b &= 2^{5+1}, & n_{cts} &= 2^{12}, & n_{it} &= 2^{13} \\ c_1 &= 8, & c_2 &= 10, & n_{byit1} &= n_{byit2} = 5, & n_{cand1} &= 2 \times n_{cand2} = 64 \end{aligned}$$

The data complexity is $n_{kg} \times n_{cts} \times n_b \times 3/4$, that is, $2^{18.58}$ plaintexts. To compare with previous attacks, the experiments were done using CPUs. Concretely, 128 trials were done with 32 threads in a CPU server¹¹. Within the 128 trials, 2 trials have no correct ciphertext structures. In the remaining 126 trials, there are 107 successful trials (the returned last two subkeys have a Hamming distance to the real subkeys at most two). The success rate is 107/128, *i.e.*, 0.8359.

¹¹ Equipped with a 32-core Intel Cascade-Lake Xeon(R) Platinum 9221 2.30 GHz, and with 384GB RAM, on CentOS 7.6.

The maximum execution time among the trials is 4.4 hours (which runs all the n_{it} , *i.e.*, 8192 iterations). Repeating 2^1 times, the maximum run time should be about 8.8 CPU hours, which is equivalent to $2^{42.97}$ executions of SPECK32/64.

Trade-off. If accepting a success rate of 0.6016, the data complexity can be further reduced to $2^{17.58}$ (by setting $n_{cts} = 2^{11}, c_1 = 7, n_{cand1} = 32$) (see Fig. 8 for details).

Comparison. Compared to classical attacks on SPECK32/64 (refer to Table 1), these new attacks commonly employ longer distinguishers consisting of short CDs and NDs, and their key-guessing phase covers fewer rounds. As for complexity, their advantage is considerable in terms of time. Compared to previous ML-based results in [15], for attacking 12-round, the success rate improves considerably using fewer data; most importantly, one more round is covered.

5 Tuning Parameters for the Key Recovery Attacks

The key-recovery attack with UCB and BAYESIANKEYSEARCH has shown its effectiveness in guessing keys in [15] and this work. However, the tuning of the parameters, especially the cutoffs, which determine the execution time and the success rate, is still missing theoretical guidance up to the time of this work. Thus, in this section, we provide detailed experimental data and derived observations to bring some light on tuning important parameters and making better trade-offs.

5.1 Exhibitions of important statistics in various attacks

It is noticed that $v_{1\max}$ (*i.e.*, $\max(\{v_{1i} \mid v_{1i} \in L_1\})$) in the key-recovery phase is an important variable determining the priority of each ciphertext structure and indicates whether promising sub-keys are discovered in each run of BAYESIANKEYSEARCH. Investigating the distributions of this variable corresponding to correct ciphertext structures (denoted by $\mathcal{D}_r^{v_{1\max}}$) and wrong ciphertext structures (denoted by $\mathcal{D}_w^{v_{1\max}}$) is helpful. These distributions can be used to learn how to tune cutoff c_1 to make trade-offs between time complexity and success rate. Investigating the distributions of $v_{2\max}$ (*i.e.*, $\max(\{v_{2i} \mid v_{2i} \in L_2\})$) could be used to learn how to tune cutoff c_2 (denoted by $\mathcal{D}_r^{v_{2\max}}$ and $\mathcal{D}_w^{v_{2\max}}$ for correct ciphertext structures and wrong structures, respectively). Thus, together with the information on attack configurations, attack complexity, and success rate, histograms are given to show $\mathcal{D}_r^{v_{1\max}}, \mathcal{D}_w^{v_{1\max}}, \mathcal{D}_r^{v_{2\max}}, \mathcal{D}_w^{v_{2\max}}$ for each presented attack ($\mathcal{A}^{\text{SPECK}_{13R}}$ and $\mathcal{A}^{\text{SPECK}_{12R}}$). Concretely, for each attack, details of the following statistics are illustrated in its corresponding figure (*e.g.*, Figures 5 to 8).

- $\mathcal{D}_w^{v_{1\max}}, \mathcal{D}_r^{v_{1\max}}, \mathcal{D}_s^{v_{1\max}}$: indicated using `rand`, `real`, and `succ` in the histograms, respectively; $\mathcal{D}_s^{v_{1\max}}$ is the distribution of $v_{1\max}$ corresponding to the successfully recovered subkeys.
- qct_w, qct_r : percentage of $v_{1\max}$'s corresponding to wrong (resp. correct) ciphertext structures passing cutoff c_1 ;

- percentage of passing samples if different cutoffs are set, including both the quantile plot with the samples and the plot with the best fitting generalized logistic distribution on the samples;
- similar statistics for $v_{2\max}$ (including $\mathcal{D}_r^{v_{2\max}}$, $\mathcal{D}_w^{v_{2\max}}$, $\mathcal{D}_s^{v_{2\max}}$)¹²;
- distribution of Hamming distances between returned and the real subkeys;
- distribution of the used number of iterations in successful attacks.

5.2 Some rules of thumb

Apart from substantial illustrations of previously hidden details of the key-recovery phase, the following observations are made to provide some rules of thumb for deciding the number of data required and the cutoff c_1 . Before that, we note that compared to c_1 , cutoff c_2 is much easier to decide because a successful attack requires the value of c_2 to be ‘at the top rank’ (compared with a ‘threshold’ sense of cutoff c_1). Thus, it is safe to select a value for c_2 that is just large enough to be uncovered by $\mathcal{D}_w^{v_{2\max}}$.

Observation 1 Suppose in the above attack framework, the probability of the prepended differential is p , the number of ciphertext structures is n_{cts} . Denote the attack success probability by P_s .

Note that $P_s \leq 1 - (1 - p \cdot q)^{n_{cts}}$, where q is the probability for the response $v_{1\max}$ from a correct ciphertext structure pass the cutoff c_1 , i.e., $q = \Pr_{\mathcal{C}_r}[v_{1\max} \geq c_1]$, where \mathcal{C}_r is space of correct ciphertext structures.

Thus, the following relation should be fulfilled:

$$n_{cts} \geq \frac{\log_2(1 - P_s)}{\log_2(1 - p \cdot q)}.$$

For given n_{cts} , p , and P_s , the cutoff c_1 should be chosen such that

$$c_1 \leq Q\left(1 - \frac{1 - (1 - P_s)^{\frac{1}{n_{cts}}}}{p}\right),$$

where $Q(\cdot)$ is the quantile function of the distribution of $v_{1\max}$ corresponding to correct ciphertext structures, i.e., $\mathcal{D}_r^{v_{1\max}}$.

For example, in the attack configuration in Fig. 5, after correctly guessing the key bits in k_0 , the probability p of the prepended differential is 2^{-9} ; suppose c_1 is selected as 18 so that q is 0.31; then, to have a success probability of 0.82, the required number of ciphertext structures, i.e., n_{cts} should satisfy $n_{cts} \geq \log 2(1 - 0.82) / \log 2(1 - 2^{-9} \cdot 0.31) \approx 2831.33 \approx 2^{11.4673}$. On the other hand, suppose one selects n_{cts} to be 2^{12} , and aims P_s to be 0.82; since p is 2^{-9} , this requires $c_1 \leq Q\left(1 - (1 - (1 - 0.82)^{2^{-12}})/2^{-9}\right) = Q(1 - 0.2143) \approx 20.5$.

Note that Observation 1 provides an upper bound on the value of the cutoff c_1 . As for a lower bound on c_1 , we provide the following observations.

¹² Some $v_{2\max}$ ’s corresponding to success cases are lower than cutoff c_2 ; that is due to the final improvement.

The cutoff c_1 seems to be the smaller, the better for having a high success probability. However, a smaller cutoff c_1 is not a better choice for having a good time complexity than a larger one. On the one hand, even using the correct ciphertext structures, if a recommended subkey gets a small score v_1 , then, typically, it also has a large Hamming distance towards the real subkeys; thus, it is hard to produce good recommendations for the second to last subkeys. On the other hand, too small cutoff c_1 results in a high percentage of v_1 from the wrong ciphertext structures passing it. As a consequence, a lot of running time will be wasted on the wrong ciphertext structures. Thus, the cutoff c_1 is better to be large enough such that a low percentage of v_1 of bad recommendations of last subkeys (*e.g.*, with more than Hamming distance 3 to the real subkey) from both correct and wrong ciphertext structures is passing it.

The preliminary to use these observations as guidance to tune the parameters is to have a good knowledge of the distribution $\mathcal{D}_r^{v_1 \max}$ and $\mathcal{D}_w^{v_1 \max}$. Experimental investigations on $\mathcal{D}_r^{v_1 \max}$ and $\mathcal{D}_w^{v_1 \max}$ can be found in Sect. B.3.

6 Neural Distinguishers on Round-Reduced SIMON32/64

This section presents the neural distinguishers on SIMON32/64 obtained in this work, using which a key-recovery attack covering 16 rounds is devised and presented in Sect. E. Besides, DDT-based \mathcal{DD} s are computed and provide baselines for \mathcal{ND} s. Comparisons between \mathcal{DD} s and \mathcal{ND} s are made accordingly.

6.1 The Choice of the Network Architecture

Considering that several state-of-the-art neural network structures have been developed, a preliminary search for a better network other than the Residual Network (ResNet) [18] used in [15] was conducted. Specifically, Dense Network (DenseNet) [20] shows advantages in parameter efficiency, implicit deep supervision, and feature reuse. Squeeze-and-Excitation Network (SENet) [19] won the first place in the *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC 2017) for classification tasks. SENet can also be combined with existing deep architectures to boost performance at a minimal additional computational cost. One example is the SE-ResNeXt that employs squeeze-and-excitation blocks and uses the ResNeXt as backbone. Thus, these two networks, together with ResNet, were investigated. The results on the performance of distinguishers that cover 7 to 9 rounds SIMON32/64 under the three different network structures are presented in Table 5. From the comparison, for longer rounds, SENet yields distinguishers that are superior to that of the other two. In the following, we only report essential details of the distinguishers trained using the SENet.

6.2 The Training of Neural Distinguishers

The training schemes follow that in [15]. All three schemes are attempted. For short rounds, the basic training scheme already works well. For longer rounds,

the KeyAverageing and Staged schemes are necessary to achieve distinguishers with non-marginal advantage. Due to the specific round structure of SIMON, distinguishers fed with partial values combined with partial differences between ciphertext pairs, instead of full values of ciphertext pairs, should be more useful than their counterparts for carrying out key-recovery attacks. Thus, we trained distinguishers accepting data composed partial values and partial differences.

The input difference is $(0x0000, 0x0040)$. This choice takes into account both the \mathcal{ND} and the prepended \mathcal{CD} , whose output difference is this input difference of the \mathcal{ND} . The goal is to obtain the best hybrid distinguisher to make the longest key-recovery attack. Therefore, the firstly examined were the intermediate differences in the best 13-round differential trail [10]. All intermediate differences were examined by training \mathcal{ND} s. This difference, $(0x0000, 0x0040)$, yielded the best 7, 8, and 9-round \mathcal{ND} s and, at the same time, allows prepending a good \mathcal{CD} , thus resulting in the best hybrid distinguisher. Note that since the differentials of SIMON32/64 has a rotational equivalent property along with the 16-bit word, all r -round \mathcal{ND} s with input difference $(0, e_i)$ and $(r - 1)$ -round \mathcal{ND} s with input difference $(e_i, 0)$ were found to have similar accuracy, for $0 \leq i < 16$.

Training using the basic scheme. Using the basic training scheme and adopting SENet (more precisely, the adopted is the SE-ResNeXt variant), neural distinguishers to recognize output pairs of 7-, 8-, 9-round SIMON32/64 with the input difference $(0x0000, 0x0040)$ are obtained. That is, given an output pair (x, y) and (x', y') and represented in the form of (x, y, x', y') , they can predict whether the data corresponds to input pairs with difference $(0x0000, 0x0040)$ of the 7-, 8-, 9-round SIMON32/64. To make a distinction from their counterparts accepting transformed data, *i.e.*, $(x, x', y \oplus y')$, the 7-, 8-, 9-round neural distinguishers presented here are named as $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_7R}$, $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_8R}$, and $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_9R}$, respectively. The 7-round $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_7R}$ achieves an accuracy as high as 0.9825, which drops by 0.17 per round to 0.8151 and 0.6325 for $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_8R}$ and $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_9R}$, respectively.

Training to simulate KeyAverageing algorithm. Successful training of the 10-round distinguisher is achieved by adopting the training scheme of simulating a KeyAverageing Algorithm [15] used with the 9-round $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_9R}$. Concretely, a size 2^{20} sample set \mathcal{S} of ciphertext pairs for 10-round SIMON32/64 is generated, one half corresponds to plaintext pairs with difference $(0x0000, 0x0040)$ and the other half corresponds to random plaintext pairs. The labels of these samples are not assigned directly but using the KeyAverageing Algorithm calling the 9-round $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_9R}$. That is, each ciphertext pair c_i in the set \mathcal{S} is decrypted by one-round using all possible values of the 10-th round subkey; thus 2^{16} intermediate values $c'_{i,j}$'s for $j \in \{0, 1\}^{16}$ are generated; grading the $c'_{i,j}$'s using the 9-round $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_9R}$, and combining the 2^{16} scores into a score for the ciphertext pair c_i by transforming the scores into real-vs-random likelihood ratios and averaging. This combined score is then taken as the label of c_i in \mathcal{S} . Using the sample set \mathcal{S} with the labels so obtained, a training, which follows the training of the best

7-round neural distinguisher in [15], is performed from a randomly initialized network state. This training procedure results in a 10-round distinguisher, named $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{10R}}$, with an accuracy of 0.5551.

Training using the Staged Training Method. The best 10-round and 11-round distinguisher are trained using the staged training method, which was the same method used to train the 8-round distinguisher of SPECK32/64 in [15]. Concretely, for training an 11-round \mathcal{ND} , in the first stage, the best 9-round distinguisher $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{9R}}$ is retained to recognize 8-round SIMON32/64 with the input difference (0x0440, 0x0100). Note that the most likely difference to appear three rounds after the input difference (0x0000, 0x0040) is (0x0440, 0x0100), and the probability is about 2^{-4} . In this first stage, the number of examples for training and for testing are 2^{28} and 2^{26} , respectively. The number of epochs is 10 and the learning rate is 10^{-4} . In the second stage, the resulted network of the first stage is retained to recognize 11-round SIMON32/64 with the input difference (0x0000, 0x0040). For this training, 2^{30} examples are freshly generated and fed, and 2^{28} examples are for verification. One epoch with a learning rate of 10^{-4} is done. In the last stage, the resulting network of the second stage is retained in two epochs with 2^{30} freshly generated data for training and 2^{28} data for verification. The learning rate is 10^{-5} . The resulting distinguisher $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{11R}}$ achieves an accuracy of 0.5174.

Training using Data of Form $(x, x', y \oplus y')$. Notice that once the output of the r -th round (x_r, x'_r, y_r, y'_r) is known, one can directly compute $(x_{r-1}, x'_{r-1}, y_{r-1} \oplus y'_{r-1})$ without knowing the $(r-1)$ -th subkey. Thus, an $(r-1)$ -round distinguisher accepting data of the form $(x, x', y \oplus y')$ can be used as an r -round distinguisher in the key-recovery attack. With this consideration, $(r-1)$ -round distinguishers accepting data of the form $(x, x', y \oplus y')$ are trained to see whether they are superior to r -round distinguishers accepting data of the form (x, x', y, y') . To make a distinction, let us denote the former by $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{(r-1)R}}$ and the latter by $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_r R}$.

The results show that $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{(r-1)R}}$ could achieve slightly better accuracy than $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_r R}$. Besides, the wrong key response profiles of $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_8 R}$ and that of $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_9 R}$ share observable pattern and symmetry. For key values that have little different from the real value, responses from $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_8 R}$ are higher than responses from $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_9 R}$. Similar observations can be derived from a comparison between that of $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_9 R}$ and that of $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{10R}}$.

Summaries on various distinguishers are presented in Table 5 for detailed accuracy and in Fig. 19 for their wrong key response profiles.

6.3 Computing DDs and Further Interpretations

To provide baselines for \mathcal{ND} s, we calculate the full distribution of differences for SIMON32/64 induced by the input difference 0x0000/0040 up to 11 rounds

Table 5: Summary of neural distinguishers on SIMON32/64

#R	Name	Network	Accuracy	True Positive Rate	True Negative Rate
6	$\mathcal{DD}_{\text{DD}}^{\text{SIMON}_6R}$	DDT	0.9918	0.9995	0.9841
7	$\mathcal{ND}_{\text{VV}}^{\text{SIMON}_7R}$	ResNet	$0.9823 \pm 1.2 \times 10^{-4}$	$0.9996 \pm 2.7 \times 10^{-5}$	$0.9650 \pm 2.3 \times 10^{-4}$
		SENet [†]	$0.9802 \pm 1.3 \times 10^{-4}$	$0.9987 \pm 4.2 \times 10^{-5}$	$0.9617 \pm 2.4 \times 10^{-4}$
		DenseNet	$0.9244 \pm 2.7 \times 10^{-4}$	$0.9670 \pm 2.2 \times 10^{-4}$	$0.8818 \pm 4.5 \times 10^{-4}$
7	$\mathcal{DD}_{\text{DD}}^{\text{SIMON}_7R}$	DDT	0.8465	0.8641	0.8288
8	$\mathcal{ND}_{\text{VV}}^{\text{SIMON}_8R}$	SENet [†]	$0.8150 \pm 4.2 \times 10^{-4}$	$0.8418 \pm 5.5 \times 10^{-4}$	$0.7882 \pm 5.1 \times 10^{-4}$
		ResNet	$0.7912 \pm 4.2 \times 10^{-4}$	$0.8041 \pm 5.5 \times 10^{-4}$	$0.7783 \pm 6.2 \times 10^{-4}$
		DenseNet	$0.7789 \pm 4.4 \times 10^{-4}$	$0.7709 \pm 6.8 \times 10^{-4}$	$0.7868 \pm 5.6 \times 10^{-4}$
8	$\mathcal{DD}_{\text{DD}}^{\text{SIMON}_8R}$	DDT	0.6628	0.5781	0.7476
8	$\mathcal{ND}_{\text{VD}}^{\text{SIMON}_8R}$	SENet [†]	$0.6587 \pm 4.8 \times 10^{-4}$	$0.5586 \pm 7.4 \times 10^{-4}$	$0.7588 \pm 5.6 \times 10^{-4}$
9	$\mathcal{ND}_{\text{VV}}^{\text{SIMON}_9R}$	SENet [†]	$0.6515 \pm 5.3 \times 10^{-4}$	$0.5334 \pm 7.0 \times 10^{-4}$	$0.7695 \pm 5.7 \times 10^{-4}$
		ResNet	$0.6296 \pm 4.5 \times 10^{-4}$	$0.5164 \pm 6.3 \times 10^{-4}$	$0.7429 \pm 5.5 \times 10^{-4}$
		DenseNet	$0.6443 \pm 4.1 \times 10^{-4}$	$0.5337 \pm 6.1 \times 10^{-4}$	$0.7550 \pm 5.0 \times 10^{-4}$
9	$\mathcal{DD}_{\text{DD}}^{\text{SIMON}_9R}$	DDT	0.5683	0.4691	0.6674
9	$\mathcal{ND}_{\text{VD}}^{\text{SIMON}_9R}$	SENet [†]	$0.5657 \pm 4.9 \times 10^{-4}$	$0.4748 \pm 7.1 \times 10^{-4}$	$0.6565 \pm 6.6 \times 10^{-4}$
10	$\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{10R}}$ + $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{10R}}$ *	SENet [†]	$0.5610 \pm 4.5 \times 10^{-4}$	$0.4761 \pm 6.0 \times 10^{-4}$	$0.6460 \pm 7.2 \times 10^{-4}$
		SENet [†]	$0.5549 \pm 4.6 \times 10^{-4}$	$0.4605 \pm 6.5 \times 10^{-4}$	$0.6493 \pm 7.7 \times 10^{-4}$
10	$\mathcal{DD}_{\text{DD}}^{\text{SIMON}_{10R}}$	DDT	0.5203	0.5002	0.5404
11	$\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{11R}}$	SENet [†]	$0.5174 \pm 5.3 \times 10^{-4}$	$0.5041 \pm 7.1 \times 10^{-4}$	$0.5307 \pm 7.9 \times 10^{-4}$
11	$\mathcal{DD}_{\text{DD}}^{\text{SIMON}_{11R}}$	DDT	0.5044	0.4852	0.5236

[†] More precisely, the adopted is the SE-ResNeXt variant.

- The network structure and parameters for the ResNet follow exactly that used in [14] for training the \mathcal{ND} s on SPECK32/64 except for the learning rate. Using a smaller learning rate (*i.e.*, cyclic_lr(10,0.001,0.00001)) instead of the original learning rate (*i.e.*, cyclic_lr(10,0.002,0.0001)) results in a better accuracy (*e.g.*, 0.6296 vs 0.6110 for 9-round) for \mathcal{ND} s on SIMON32/64.

* This neural distinguisher is trained using the KEYAVERAGING algorithm.

+ This neural distinguisher is trained using the staged training method.

(see Table 5). This is done using the framework of Gohr’s implementation for SPECK32/64 and integrating the algorithm for computing one-round differential probability for SIMON offered by Kölbl *et al.* in [24]. Note that, the fed data to r -round ND are values of ciphertexts, from which, for SIMON, one can directly compute the differences on $(r-1)$ -round outputs without knowing the subkey. Thus, $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_rR}$ or $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{(r-1)}R}$ should be compared with $\mathcal{ND}_{\text{DD}}^{\text{SIMON}_{(r-1)}R}$.

The results show that $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_rR}$ and $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{(r-1)}R}$ achieve similar but weaker classification accuracy than $\mathcal{ND}_{\text{DD}}^{\text{SIMON}_{(r-1)}R}$. To further evaluate the gaps between the advantage of \mathcal{DD} over \mathcal{ND} , we devised a key ranking task, as done by Gohr for comparing \mathcal{ND} s and \mathcal{DD} s on SPECK32/64 in [15]. Specifically, a simple key ranking procedure to recover the last subkey on 11-round SIMON32/64 can be performed both by $\mathcal{DD}_{\text{DD}}^{\text{SIMON}_8R}$ or $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_8R}$ in a configuration of 1+8+2. Table 6 shows the performance of $\mathcal{DD}_{\text{DD}}^{\text{SIMON}_8R}$ and $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_8R}$ in the ranking for real subkeys among 2^{16} candidate subkeys. It can be seen that they both work well in this task; the data requirement is 64 chosen plaintexts to achieve a success

Table 6: Comparing \mathcal{ND} and \mathcal{DD} on SIMON32/64 using statistics in a simple key recovery attack on 11-round SIMON32/64. The configuration is 1+8+1+1, *i.e.*, a free prepended invert round, an 8-round distinguisher, a free inverting round, and a key-guessing (last) round. All data are based on 1000 trials of the respective attacks, all measurements of these statistics follow that in [15]: The rank of the real subkey is in the range $[0, 2^{16}]$; it is defined as the number of subkeys ranked higher, *i.e.*, rank 0 corresponds to successful key recovery. When several keys were ranked equally, the right key was assumed to be in a random position among the equally ranked keys. The reported error bars around the mean are for a 2σ confidence interval, where σ is calculated based on the observed standard deviation of the key rank. #D indicates the number of chosen plaintexts.

#D	Distinguisher	Mean of key rank	Median key rank	Success rate
32 × 2	$\mathcal{DD}_{\text{DD}}^{\text{SIMON}8R}$	11.8 ± 3.1	1.0	0.238
	$\mathcal{ND}_{\text{VD}}^{\text{SIMON}8R}$	43.9 ± 21.4	2.0	0.188
64 × 2	$\mathcal{DD}_{\text{DD}}^{\text{SIMON}8R}$	0.9 ± 0.2	1.0	0.415
	$\mathcal{ND}_{\text{VD}}^{\text{SIMON}8R}$	1.3 ± 0.2	1.0	0.335

rate of around 20%. However, $\mathcal{ND}_{\text{VD}}^{\text{SIMON}8R}$ is slightly inferior to $\mathcal{DD}_{\text{DD}}^{\text{SIMON}8R}$. To achieve the same success rate, $\mathcal{ND}_{\text{VD}}^{\text{SIMON}8R}$ requires more data than $\mathcal{DD}_{\text{DD}}^{\text{SIMON}8R}$, but the difference is less than twice.

These comparisons suggest that r -round $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_r R}$ can “decrypt” one unkeyed round to obtain the $(r - 1)$ -round difference and learn the differential distribution, which confirms the interpretation in [7], but fails to learn more features beyond the distribution of differences.

Remark 8. This fact for SIMON is different from the corresponding conclusion for SPECK. For SPECK, knowing values of ciphertexts, without knowing the subkey, one can only compute half but not full of the differences on $(r - 1)$ -round outputs. Thus, the counterpart of r -round \mathcal{ND} is r -round \mathcal{DD} . From [15], r -round \mathcal{ND} learns additional features beyond differences and has better classification accuracy than r -round \mathcal{DD} . We conjecture that the main reason is that, for SPECK, pure XOR-difference \mathcal{DD} s cannot provide the best baselines for \mathcal{ND} s. On the one hand, they are not accurate because of being computed following the Markov assumption. On the other hand, features related to generalized XOR-difference through modular addition and multi-bit constraints [12, 25] might be useful to capture the additional features in outputs of SPECK32/64. For examples, Tables 7 and 8 present generalized constraints beyond XOR-differences on some differential trails, considering which the probability of the trails could be refined. In contrast, for SIMON32/64, the XOR-differences distribution table computed using the Markov model might already be an accurate approximation for the actual differential distribution.

We note that the \mathcal{ND} s on SPECK32/64 might also “decrypt” half of the “unkeyed” last round to retrieve the input values on the right branch y_{r-1} . This interesting fact that the \mathcal{ND} s can “learn to decrypt up to the values not messed up by outer subkey” might be due to the design by Gohr, as explained in [15].

as “the use of the initial width-1 convolutional layer is intended to make the learning of simple bit-sliced functions such as bit-wise addition easier”. Remarkably, for SIMON32/64, the \mathcal{ND} s seems to have also successfully peeled off the nonlinear bit-wise AND layer in the last round. For deeper look into the \mathcal{ND} s on SIMON32/64, please refer to Sect. G.

7 Conclusions and Future Work

This paper shows practical key-recovery attacks up to 13 rounds of SPECK32/64. This advances state of the art on practical attacks by one round. It shows that the way the underlying neural distinguishers were used in the previous differential-neural attacks is not optimal. Accordingly, the differential-neural cryptanalysis on SPECK32/64 has more potential than it originally exhibited.

The methods developed, particularly those generalized neutral bits, are not intrinsically linked to neural network-based cryptanalysis. They are expected to be useful for the conversion of a wider range of deep weak distinguishers to competitive key recovery attacks in general.

The experiments and comparisons made on various distinguishers on round-reduced SIMON32/64 indicate that differential-based neural-distinguishers should work well in general on modern ciphers. Still, they may not always be superior to their classical counterparts. Their advantages might be easier to show on ciphers, on which the differential-like properties have not been accurately evaluated using existing tools.

The provided rules of thumb on turning parameters in the UCB and Bayesian optimization-based key-recovery phase are helpful but far from perfect. For this advanced key-recovery strategy to be widely applied, a rigorous theoretical model on the relation between attack parameters, attack complexity, and success probability is missing, and the building of which is left as future work.

Acknowledgments

The authors would like to thank anonymous reviewers for their insightful and helpful comments which helped us improve the manuscript significantly. This research is partially supported by Nanyang Technological University in Singapore under Start-up Grant 04INS000397C230, and Ministry of Education in Singapore under Grants RG91/20 and MOE2019-T2-1-060; Zhenzhen Bao was supported by the Gopalakrishnan – NTU Presidential Postdoctoral Fellowship 2020; the Tsinghua University in China under Start-up Grant 533344001; the National Key R&D Program of China (Grant No. 2018YFA0704701), the Major Program of Guangdong Basic and Applied Research (Grant No. 2019B030302008), the Shandong Province Key R&D Project (Nos. 2020ZLYS09 and 2019JZZY010133). Meicheng Liu was supported by the National Natural Science Foundation of China (Grant Nos. 62122085 and 12231015), and the Youth Innovation Promotion Association of Chinese Academy of Sciences.

References

1. M. Abadi and D. G. Andersen. Learning to protect communications with adversarial neural cryptography. *arXiv preprint arXiv:1610.06918*, 2016.
2. F. Abed, E. List, S. Lucks, and J. Wenzel. Differential cryptanalysis of round-reduced Simon and Speck. In C. Cid and C. Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 525–545. Springer, Heidelberg, Mar. 2015.
3. H. A. Alkhzaimi and M. M. Lauridsen. Cryptanalysis of the SIMON family of block ciphers. Cryptology ePrint Archive, Report 2013/543, 2013. <https://eprint.iacr.org/2013/543>.
4. J.-P. Aumasson, S. Fischer, S. Khazaei, W. Meier, and C. Rechberger. New features of latin dances: Analysis of Salsa, ChaCha, and Rumba. In K. Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 470–488. Springer, Heidelberg, Feb. 2008.
5. Z. Bao, J. Guo, M. Liu, L. Ma, and Y. Tu. Enhancing differential-neural cryptanalysis. Cryptology ePrint Archive, Report 2021/719, 2021. <https://eprint.iacr.org/2021/719>.
6. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. <https://eprint.iacr.org/2013/404>.
7. A. Benamira, D. Gérault, T. Peyrin, and Q. Q. Tan. A deeper look at machine learning-based cryptanalysis. In A. Canteaut and F.-X. Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 805–835. Springer, Heidelberg, Oct. 2021.
8. T. Beyne and V. Rijmen. Differential cryptanalysis in the fixed-key model. Cryptology ePrint Archive, Paper 2022/837, 2022. <https://eprint.iacr.org/2022/837>.
9. E. Biham and R. Chen. Near-collisions of SHA-0. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 290–305. Springer, Heidelberg, Aug. 2004.
10. A. Biryukov, A. Roy, and V. Velichkov. Differential analysis of block ciphers SIMON and SPECK. In C. Cid and C. Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 546–570. Springer, Heidelberg, Mar. 2015.
11. M. Brickenstein, A. Dreyer, B. Erocal, M. Albrecht, S. King, and C. Bouillaguet. Sage 9.3 Reference Manual: Polynomials: Boolean Polynomials. https://doc.sagemath.org/html/en/reference/polynomial_rings/sage/rings/polynomial/pbori/pbori.html. Accessed: 2021-5.
12. C. De Cannière and C. Rechberger. Finding SHA-1 characteristics: General results and applications. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 1–20. Springer, Heidelberg, Dec. 2006.
13. I. Dinur. Improved differential cryptanalysis of round-reduced Speck. In A. Joux and A. M. Youssef, editors, *SAC 2014*, volume 8781 of *LNCS*, pages 147–164. Springer, Heidelberg, Aug. 2014.
14. A. Gohr. Implementation of the Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning. GitHub Repository. https://github.com/agohr/deep_speck, 2019.
15. A. Gohr. Improving attacks on round-reduced Speck32/64 using deep learning. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 150–179. Springer, Heidelberg, Aug. 2019.
16. A. N. Gomez, S. Huang, I. Zhang, B. M. Li, M. Osama, and L. Kaiser. Unsupervised cipher cracking using discrete gans. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

17. L. Grassi. Mixture differential cryptanalysis: a new approach to distinguishers and attacks on round-reduced AES. *IACR Trans. Symm. Cryptol.*, 2018(2):133–160, 2018.
18. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.
19. J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. Squeeze-and-excitation networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(8):2011–2023, 2020.
20. G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269. IEEE Computer Society, 2017.
21. A. Joux and T. Peyrin. Hash functions and the (amplified) boomerang attack. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 244–263. Springer, Heidelberg, Aug. 2007.
22. V. Klima. Tunnels in hash functions: MD5 collisions within a minute. *Cryptology ePrint Archive*, Report 2006/105, 2006. <https://eprint.iacr.org/2006/105>.
23. S. Knellwolf, W. Meier, and M. Naya-Plasencia. Conditional differential cryptanalysis of NLFSR-based cryptosystems. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 130–145. Springer, Heidelberg, Dec. 2010.
24. S. Kölbl, G. Leander, and T. Tiessen. Observations on the SIMON block cipher family. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 161–185. Springer, Heidelberg, Aug. 2015.
25. G. Leurent. Construction of differential characteristics in ARX designs application to Skein. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 241–258. Springer, Heidelberg, Aug. 2013.
26. J. Rijsdijk, L. Wu, G. Perin, and S. Picek. Reinforcement learning for hyper-parameter tuning in deep learning-based side-channel analysis. *IACR TCCHS*, 2021(3):677–707, 2021. <https://tches.iacr.org/index.php/TCCHS/article/view/8989>.
27. R. L. Rivest. Cryptography and machine learning (invited lecture). In H. Imai, R. L. Rivest, and T. Matsumoto, editors, *ASIACRYPT’91*, volume 739 of *LNCS*, pages 427–439. Springer, Heidelberg, Nov. 1993.
28. D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.
29. L. Song, Z. Huang, and Q. Yang. Automatic differential analysis of ARX block ciphers with application to SPECK and LEA. *Cryptology ePrint Archive*, Report 2016/209, 2016. <https://eprint.iacr.org/2016/209>.
30. Stefan Kölbl. CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives. <https://github.com/kste/cryptosmt>.
31. N. Wang, X. Wang, K. Jia, and J. Zhao. Differential attacks on reduced SIMON versions with dynamic key-guessing techniques. *Cryptology ePrint Archive*, Report 2014/448, 2014. <https://eprint.iacr.org/2014/448>.
32. X. Wang and H. Yu. How to break MD5 and other hash functions. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 19–35. Springer, Heidelberg, May 2005.

A SBfADs for XOR-addition modulo differentials

A.1 One type of SBfAD

The following introduces one type of SBfAD for XOR (\oplus) differential of addition modulo 2^n (\boxplus).

For simplicity, consider an \oplus -difference propagation through a single \boxplus , denoted by $\delta = (\alpha, \beta \mapsto \gamma)$. For an input pair $((x, y), (x \oplus \alpha, y \oplus \beta))$ to conform to the difference propagation δ , that is $(x \boxplus y) \oplus ((x \oplus \alpha) \boxplus (y \oplus \beta)) = \gamma$, the conforming conditions can be explicitly listed bit-by-bit. Among the bit-by-bit conditions, one type, denoted by xy-type, is linear and is in the form of $x[i] \oplus y[i] = a$; another type, denoted by c-type, is non-linear and is in the form of $x[i] \oplus c[i] = a$ or $y[i] \oplus c[i] = a$, where $a \in \{0, 1\}$, $c[i]$ is the i -th bit of the carry c , and $c = (x \boxplus y) \oplus (x \oplus y)$.

For two \oplus -difference propagation through a single \boxplus , δ_1 and δ_2 , if all conforming conditions are the same except one, e.g., $x[i] \oplus y[i] = 0$ for δ_1 and $x[i] \oplus y[i] = 1$ for δ_2 , and the i -th bit of x together with the i -th bit of y forms an SNBS of δ_1 , then for a conforming pair $((x, y), (x \oplus \alpha, y \oplus \beta))$ of δ_1 , flipping $x[i]$ or $y[i]$ and changing the input difference, the resulted pair conforms to δ_2 . Thus, the i -th bit of x or of y is an SBfAD for adjoining differentials δ_1 and δ_2 . For two \oplus -differentials propagating several rounds, the x means the input to the first \boxplus . That is, the i -th bit referred is the $((i+7) \bmod 16)$ -th bit of x in SPECK32/64.

A.2 An SBfAD for the 3-round Differentials of SPECK32/64

The following analysis how

$(0x8020, 0x4101) \rightarrow (0x0040, 0x0000)$ and $(0x8060, 0x4101) \rightarrow (0x0040, 0x0000)$ (resp.

$(0x8021, 0x4101) \rightarrow (0x0040, 0x0000)$ and $(0x8061, 0x4101) \rightarrow (0x0040, 0x0000)$) can be switched by bit 5 of x .

We note that each of the four differentials consists of a dominant trail. Four dominant trails of the four differentials share the same inner core (colored in gray in rows 2 and 3 of Table 3). Specifying to these dominant trails, there is an additional necessary condition (colored in gray in the last row of Table 3) for conforming to each of the dominant trails ¹³. For each of the dominant trails, the additional necessary condition together with the above three necessary conditions form necessary and sufficient conditions for conforming to the outermost round differential, which is the only different part among the four trails.

Accordingly, from Table 3, the dominant trails of differentials $(0x8020, 0x4101) \rightarrow (0x0040, 0x0000)$ and $(0x8060, 0x4101) \rightarrow (0x0040, 0x0000)$ (resp.

¹³ These additional conditions are not necessary to the differentials because they are not necessary for subordinate trails in the differentials. However, the subordinate trails have far small probabilities than the dominant ones. Thus, these additional conditions can be seen as almost necessary to the differentials (about 99.5% conforming pairs fulfill the additional conditions).

$(0x8021, 0x4101) \rightarrow (0x0040, 0x0000)$ and $(0x8061, 0x4101) \rightarrow (0x0040, 0x0000)$) have only one different condition, that is $x[5] \oplus y[14]$ equals 1 or 0. Besides, from Table 4, the bit 14 of y and the bit 5 of x (the 21 bit of $x||y$) forms an SNBS. Thus, according to Sect. A.1, $(0x8020, 0x4101) \rightarrow (0x0040, 0x0000)$ and $(0x8060, 0x4101) \rightarrow (0x0040, 0x0000)$ (resp. $(0x8021, 0x4101) \rightarrow (0x0040, 0x0000)$ and $(0x8061, 0x4101) \rightarrow (0x0040, 0x0000)$) can be seen as adjoining differentials, and the bit 5 of x is their switching bit.

B Visualizations and Experimental Investigations

B.1 Visualizing the framework and the components of the key-recovery attacks on SPECK32/64

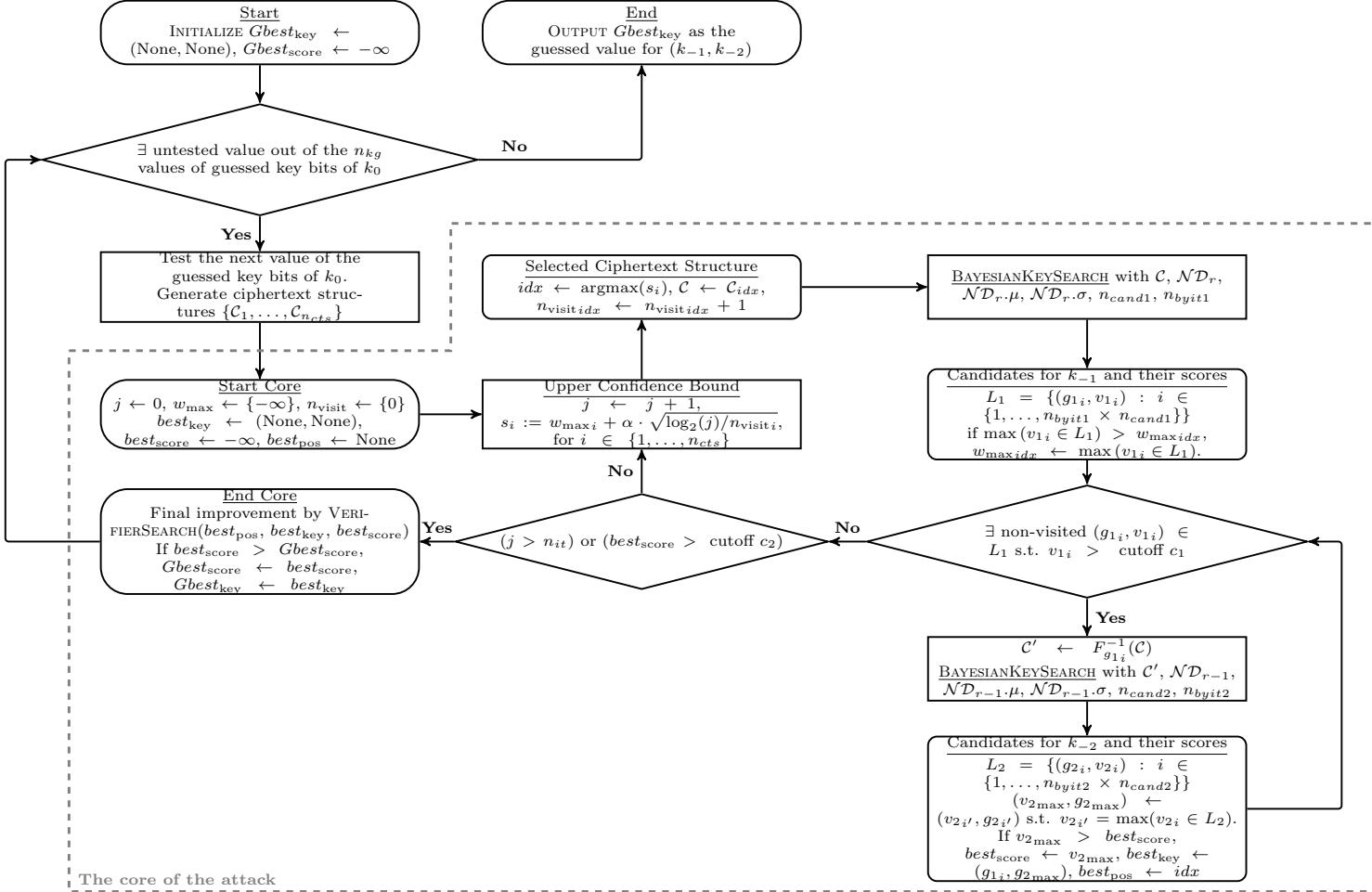
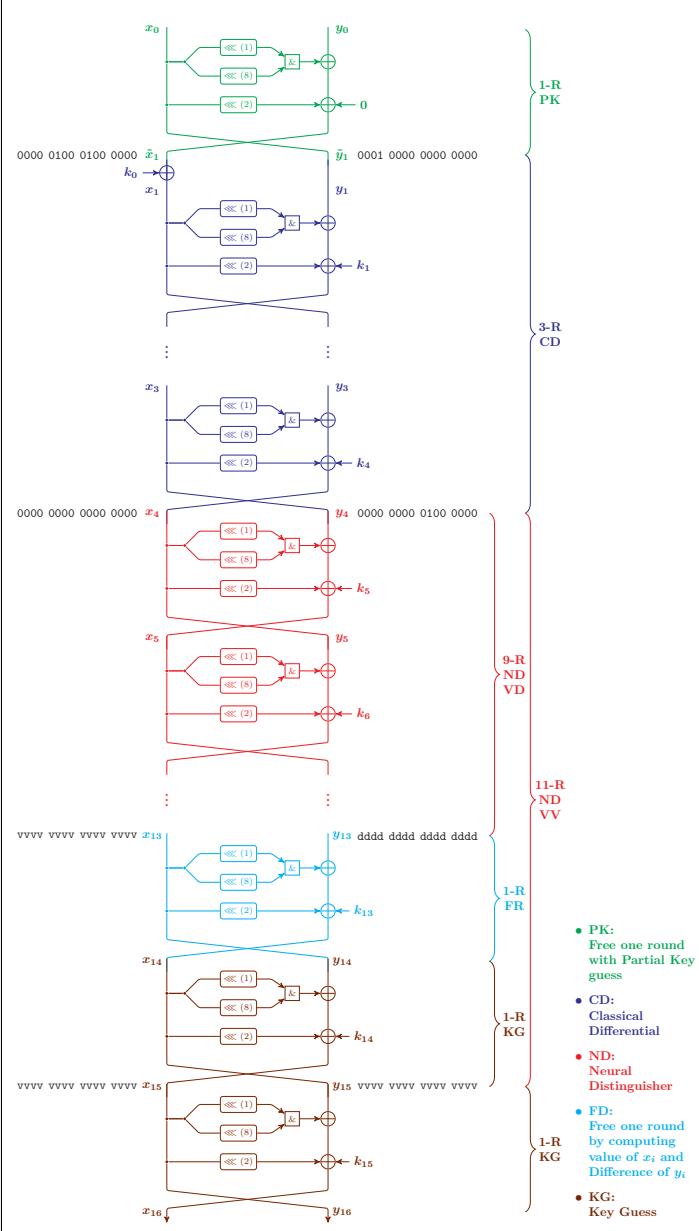
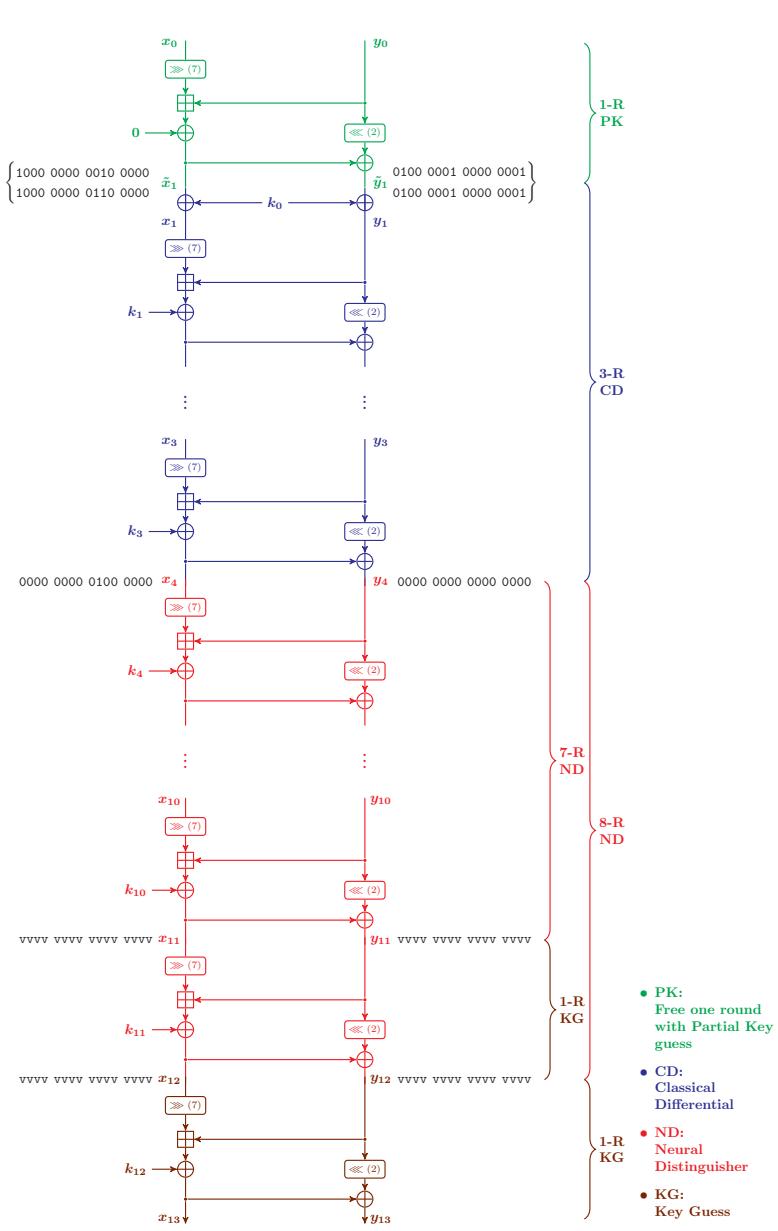


Fig. 2: Framework of the key-recovery attacks



B.2 Visualizing distributions of statistics for various attacks

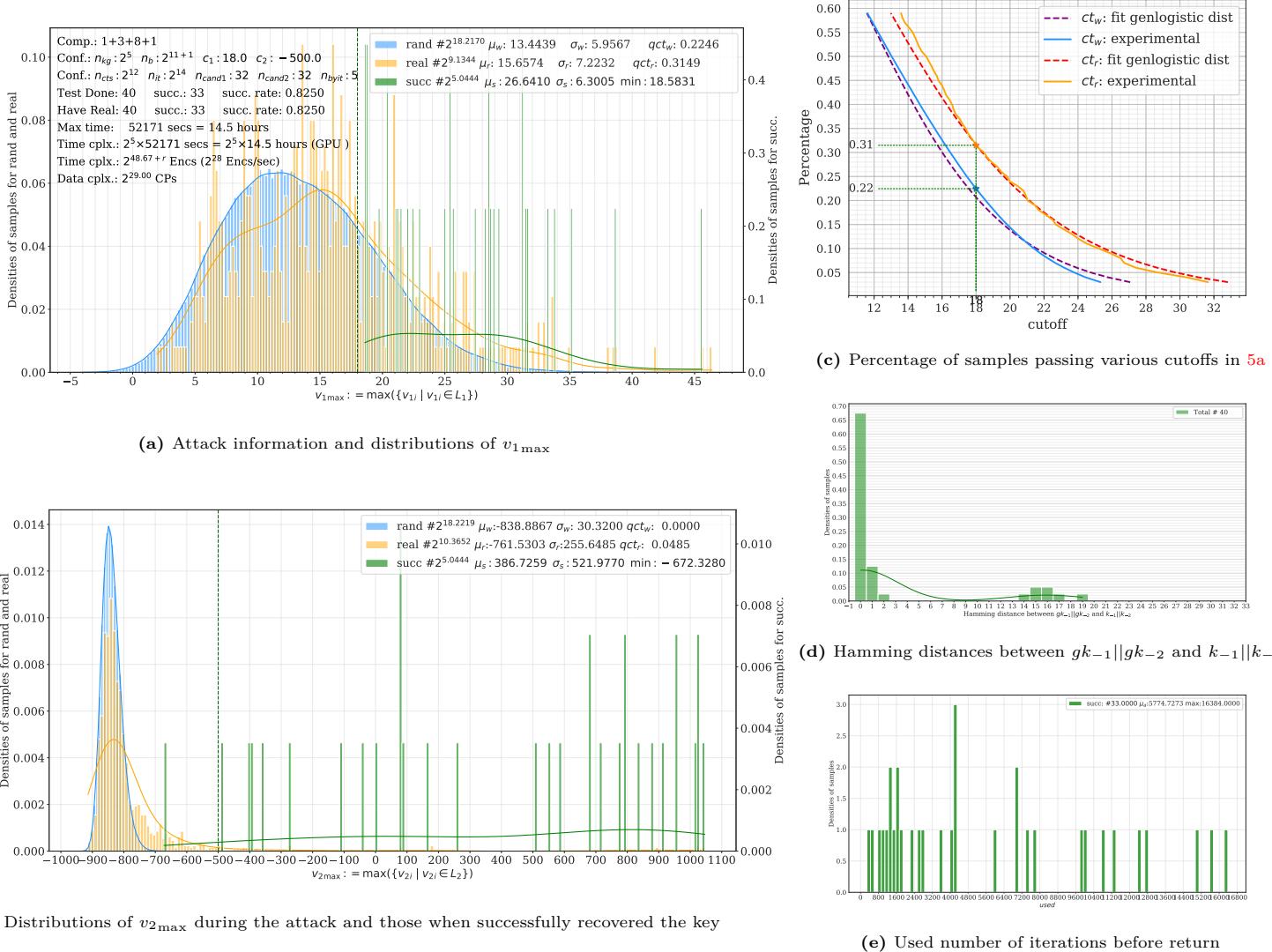


Fig. 5: Detailed information for attack $A_I^{SPECK13R}$

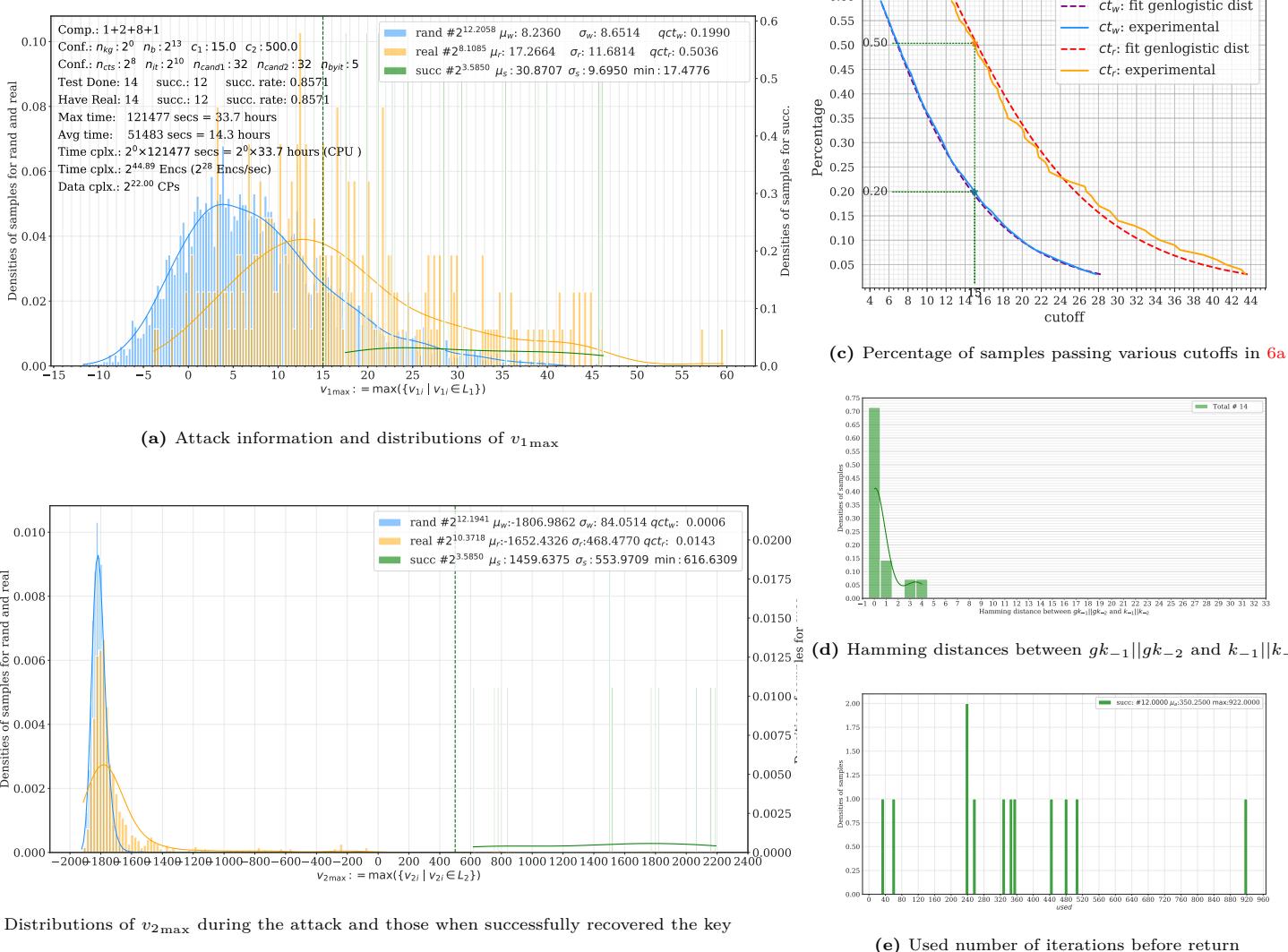


Fig. 6: Detailed information for attack $A_I^{SPECK12R}$

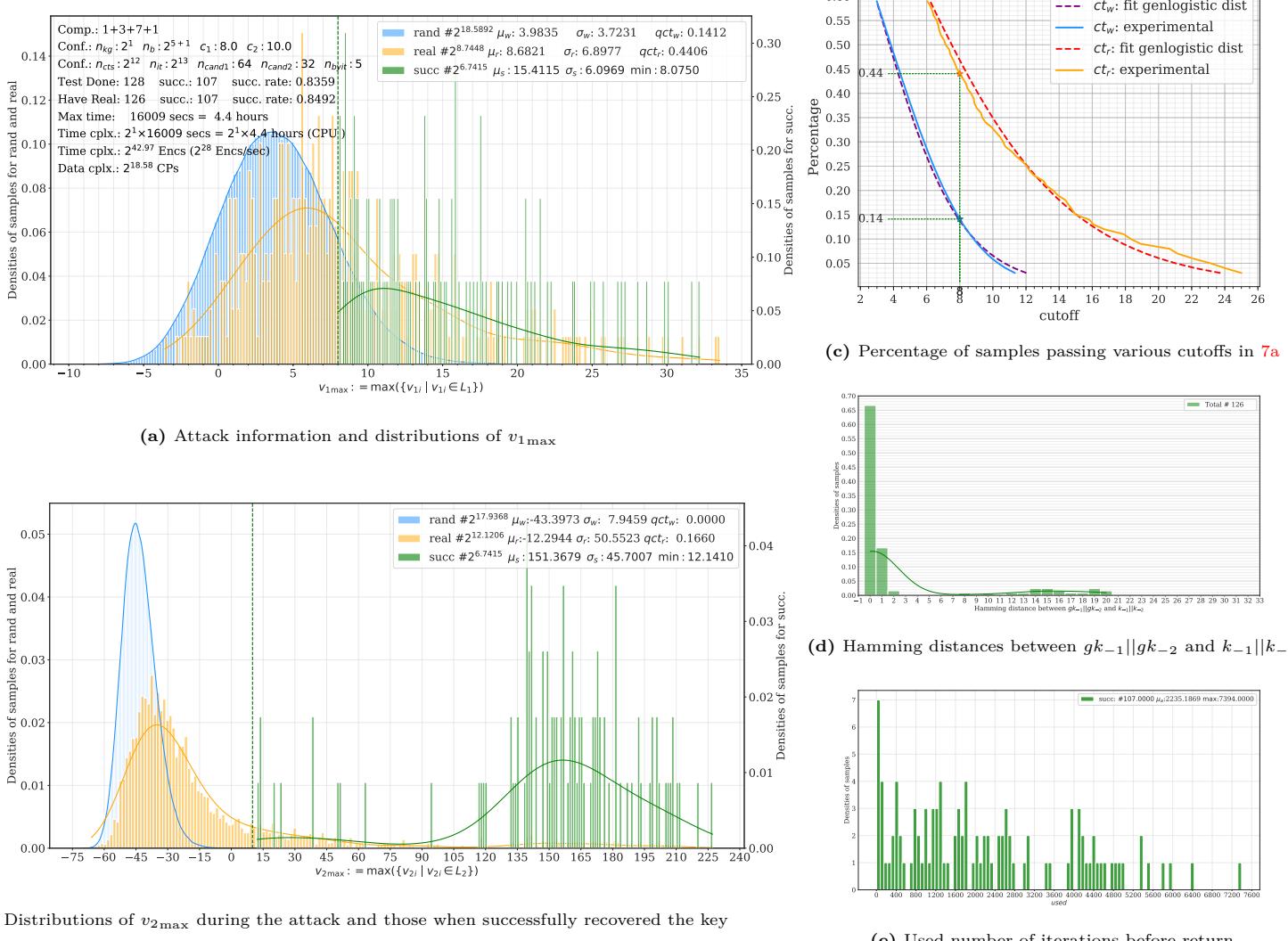


Fig. 7: Detailed information for attack $A_{II}^{SPECK12R}$

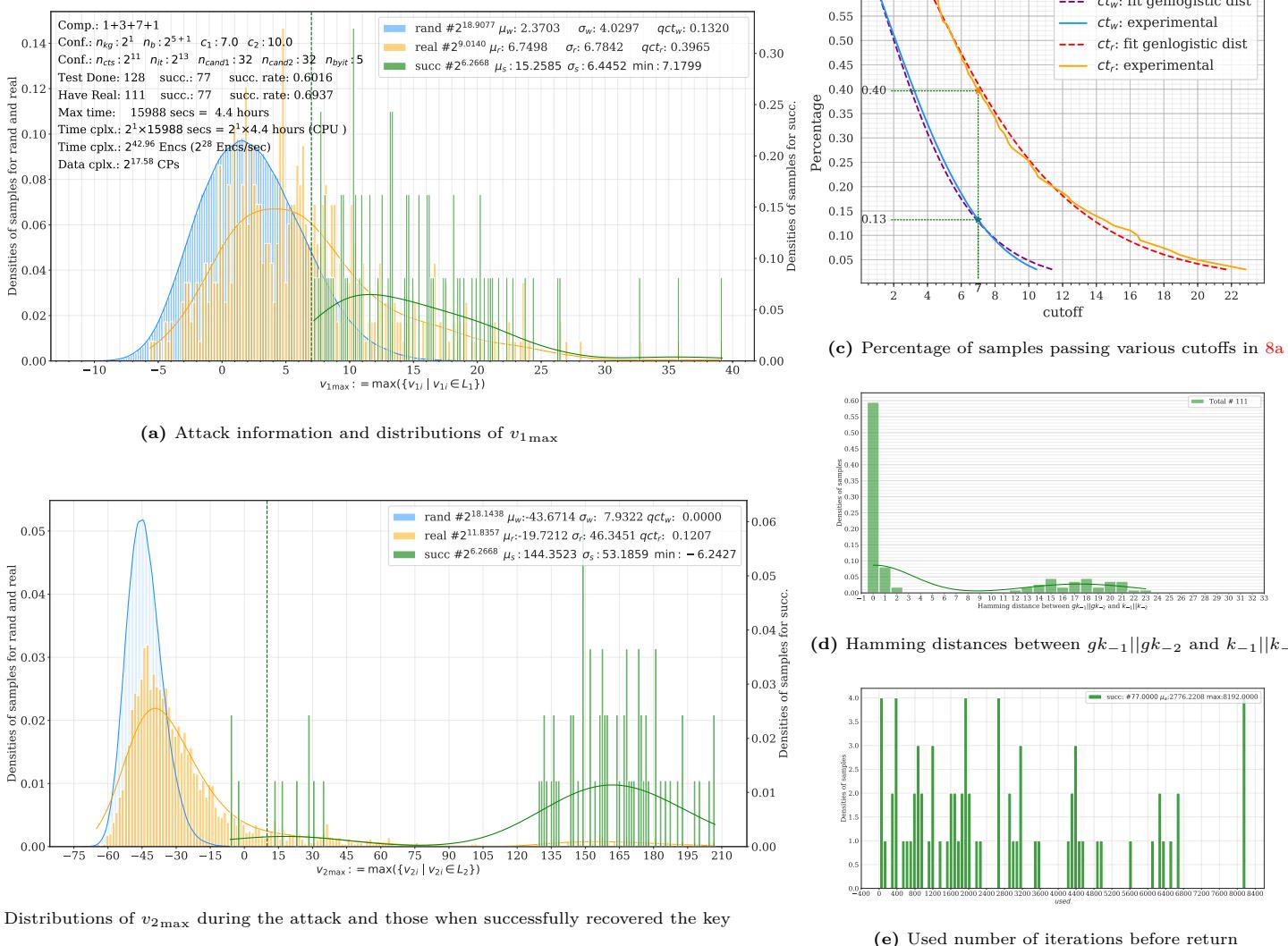


Fig. 8: Detailed information for attack $A_{III}^{SPECK12R}$

B.3 Investigations on $\mathcal{D}_r^{v_{1\max}}$ and $\mathcal{D}_w^{v_{1\max}}$

The experimental investigations on $\mathcal{D}_r^{v_{1\max}}$ and $\mathcal{D}_w^{v_{1\max}}$ were done through sampling about 2^{16} correct ciphertext structures and 2^{16} wrong ciphertext structures and analyzing the values of $v_{1\max}$ statistically (Figures 9 to 13). These experiments use exactly the same procedure of the key-recovery attack but generate ciphertext structures by accessing the subkeys; besides, it does not run into guessing the second to the last subkey. The same neutral bits used in actual attacks are also used here to generate the ciphertext structures. Since some neutral bits are probabilistic, using ciphertext structures generated by different neutral bits, the simulation of the $\mathcal{D}_r^{v_{1\max}}$ and $\mathcal{D}_w^{v_{1\max}}$ will be slightly different. This is aimed at using these investigations to guide the actual attacks.

Apart from $\mathcal{D}_r^{v_{1\max}}$ and $\mathcal{D}_w^{v_{1\max}}$, for correct ciphertext structures, distribution of $v_{1\max}$ corresponding to the recommended subkeys of low Hamming distances (from 0 to 3 bits) towards the real subkey are investigated and presented together (*e.g.*, Figures 11 to 13).

For experimental results, we have the following observation.

Observation 2 *Among various distributions, including Normal, Chi-squared, Generalized logistic, Logistic, and Gamma, the Generalized logistic distribution¹⁴ (denote by genlogistic) provides the best fit for both $\mathcal{D}_r^{v_{1\max}}$ and $\mathcal{D}_w^{v_{1\max}}$. Besides, $\mathcal{D}_r^{v_{1\max}}$'s are heavy right-skewed.*

In figures that display $\mathcal{D}_r^{v_{1\max}}$ and $\mathcal{D}_w^{v_{1\max}}$, the parameters of the best fitting generalized logistic distributions are thus provided.

The influence on $\mathcal{D}_r^{v_{1\max}}$ and $\mathcal{D}_w^{v_{1\max}}$ when changing the size of ciphertext structures (n_b) (determined by the number of used neutral bits), the number of recommended keys in each iteration inside BayesianKeySearch (n_{cand}), and the number of iterations in each BayesianKeySearch (n_{byit}) are investigated.

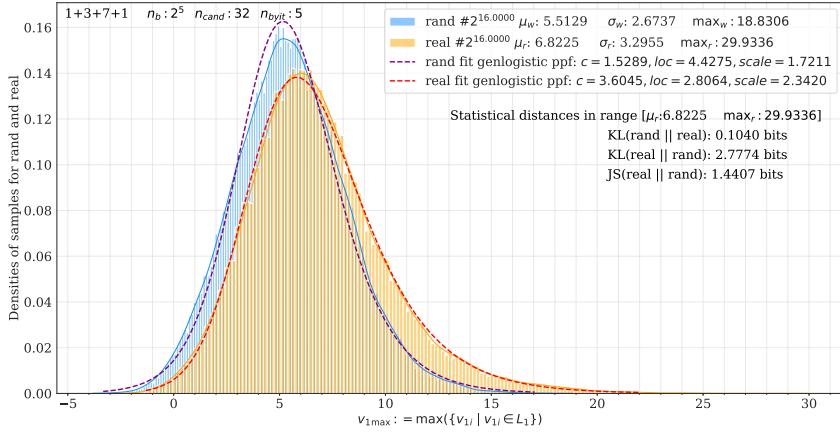
To quantify the influence, the Kullback-Leibler Divergence $KL(\text{real}||\text{rand})$ in the range $[\mu_r, \max_r]$ is considered, where μ_r is the mean of $v_{1\max}$ from correct ciphertext structures and \max_r is the maximum. Considering only this range is because, in the actual attacks, the cutoffs are generally selected to be no less than μ_r . Subfigures in Fig. 9 show how the distributions change when changing n_b . From Figures 9a and 9c, for that attack configuration, increasing n_b from 2^5 to 2^6 , the $KL(\text{real}||\text{rand})$ increases considerably. Increasing from 2^6 to 2^7 (Fig. 9c and 9e), the mean increase approximately 2 times, however, the $KL(\text{real}||\text{rand})$ does not increase but slightly decreases. That can be understood by looking at Figures 17g and 17h, which show that if combining responses on 2^6 samples, the two distributions of combined-response on random samples and real samples are already separated. From the quantile plot in Figures 9b and 9d, setting cutoff to be 8, when $n_b = 2^5$, approximately 17% of $v_{1\max}$ from the wrong ciphertext structures pass, and 30% from the correct ciphertext structures pass; whereas when $n_b = 2^6$, 9% from the wrong ciphertext structures pass and 35% from the

¹⁴ Type I generalized logistic distribution with probability density function $f(y, c) = c \frac{e^{-y}}{(1+e^{-y})^{c+1}}$, where $y = \frac{(x-\text{loc})}{\text{scale}}$, for $x \geq 0$ and $c > 0$.

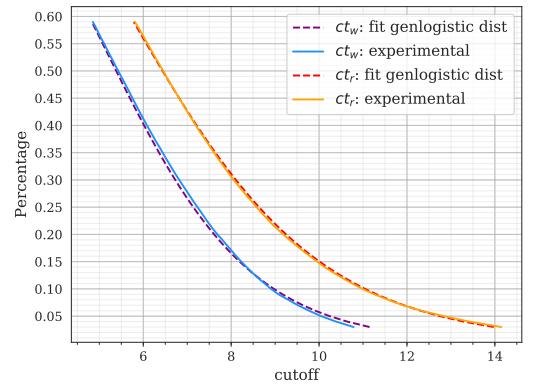
correct ciphertext structures pass. Thus, the latter is much better for achieving a good trade-off between time complexity and success rate for the attack. Such an obvious advantage cannot be seen for $n_b = 2^7$ over $n_b = 2^6$. Thus, $n_b = 2^6$ is sufficient for the corresponding attacks.

Similar comparisons among Figures (10a, 10b), (10c, 10d), and (10e, 10f) indicate that increasing n_{cand} is more effective than increasing n_{byit} for separating the two distributions. Thus, tuning n_{cand} could achieve better trade-offs between time complexity and success rate than tuning n_{byit} .

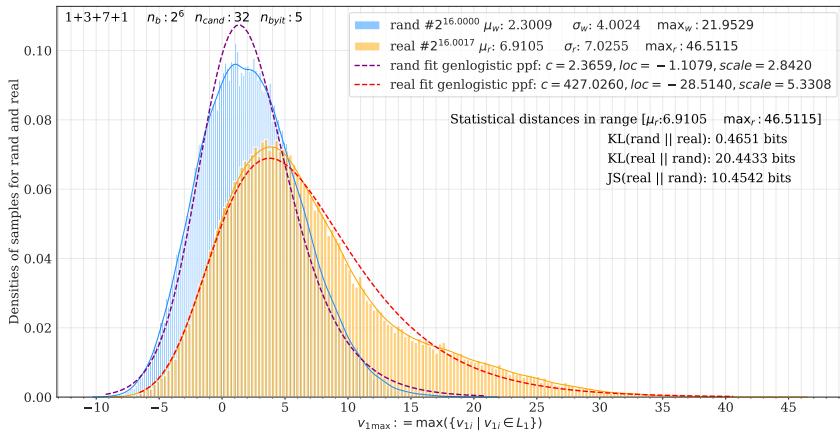
For different attacks, the significance of the influence by increasing n_{cand} are different. Increasing n_{cand} might fail to result in considerable improvements in separating the distributions (see Fig. 11 and 12 for attacks of composition 1+3+8+1). However, the probability that guessed subkeys with low hamming distances to the real subkey can increase to be doubled (comparing Fig. 11c and 12c). Thus, n_{cand} can still be used to make trade-offs between time complexity and success probability without changing data complexity for the attacks.



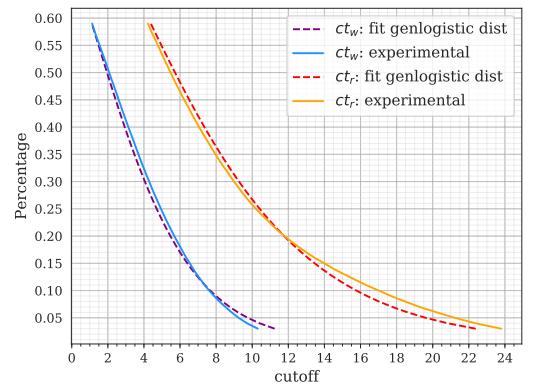
(a) Distributions $\mathcal{D}_r^{v_{1\max}}$ and $\mathcal{D}_w^{v_{1\max}}$ when using 5 NBs



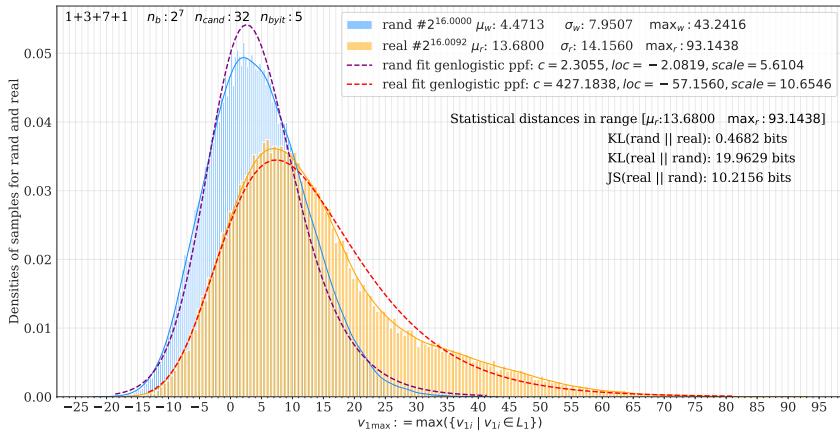
(b) Percentage of samples passing various cutoffs in 9a



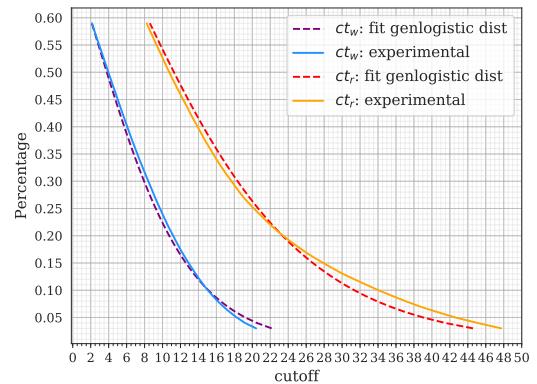
(c) Distributions $\mathcal{D}_r^{v_{1\max}}$ and $\mathcal{D}_w^{v_{1\max}}$ when using 6 NBs



(d) Percentage of samples passing various cutoffs in 9c

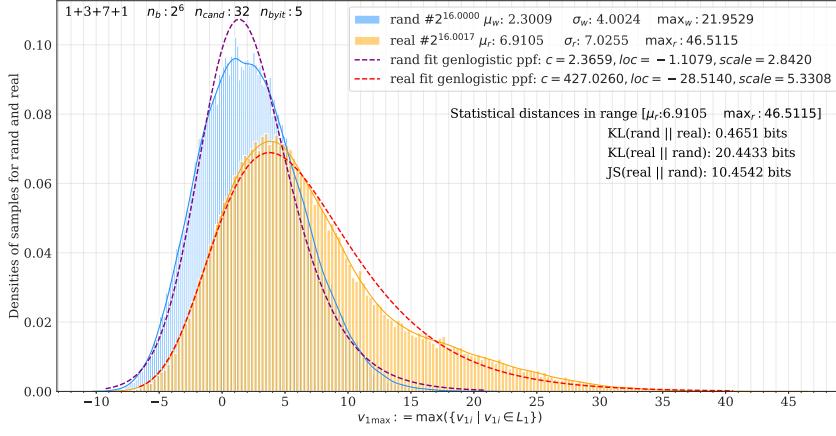


(e) Distributions $\mathcal{D}_r^{v_{1\max}}$ and $\mathcal{D}_w^{v_{1\max}}$ when using 7 NBs

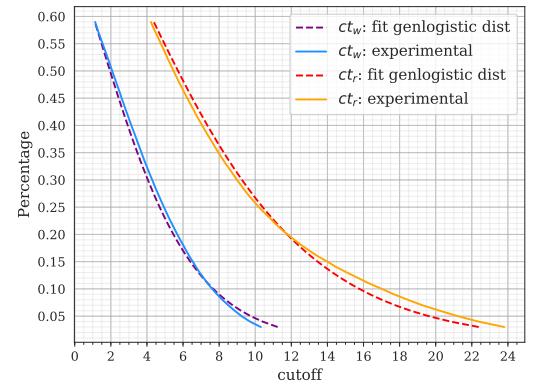


(f) Percentage of samples passing various cutoffs in 9e

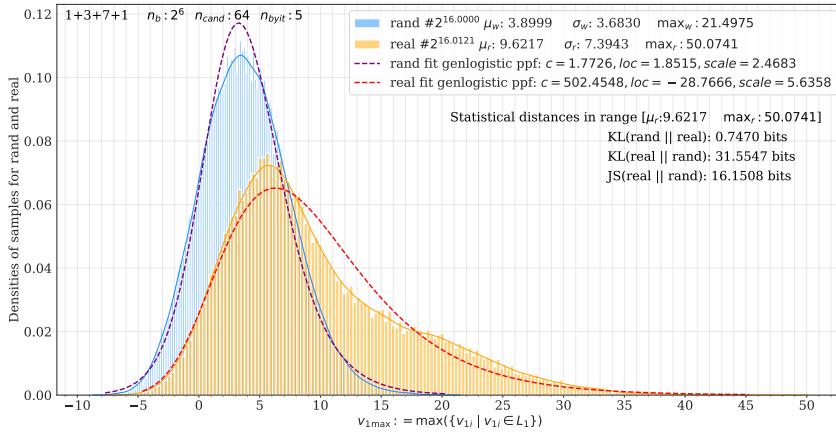
Fig. 9: Distributions of $v_{1\max}$ from correct ciphertext structures (real) and from wrong ciphertext structures (rand) of size n_b for $n_b \in \{2^5, 2^6, 2^7\}$



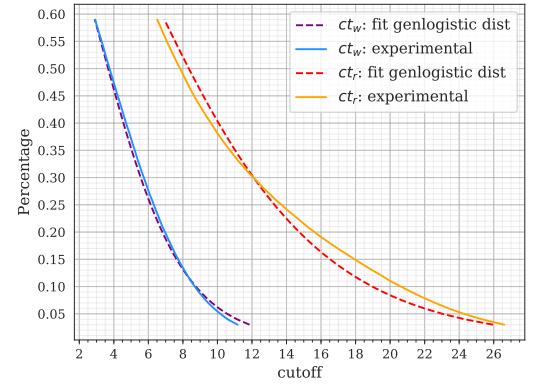
(a) Distributions $\mathcal{D}_r^{v1\max}$ and $\mathcal{D}_w^{v1\max}$ when $n_b = 2^6$, $n_{cand} = 32$, $n_{byit} = 5$



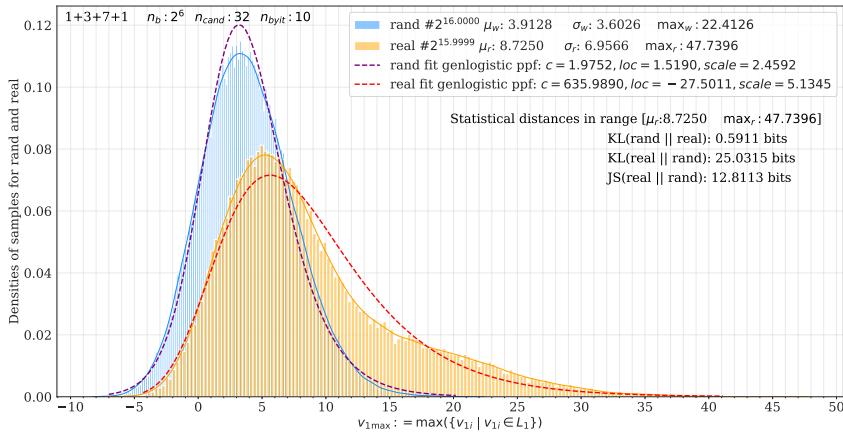
(b) Percentage of samples passing various cutoffs in 10a



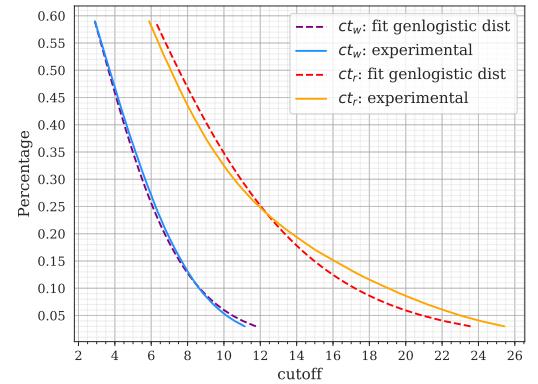
(c) Distributions $\mathcal{D}_r^{v1\max}$ and $\mathcal{D}_w^{v1\max}$ when $n_b = 2^6$, $n_{cand} = 64$, $n_{byit} = 5$



(d) Percentage of samples passing various cutoffs in 10c



(e) Distributions $\mathcal{D}_r^{v1\max}$ and $\mathcal{D}_w^{v1\max}$ when $n_b = 2^6$, $n_{cand} = 32$, $n_{byit} = 10$



(f) Percentage of samples passing various cutoffs in 10e

Fig. 10: Distributions of $v_{1\max}$ from correct ciphertext structures (**real**) and from wrong ciphertext structures (**rand**) with n_b fix to 2^6 , $n_{cand} \in \{32, 64\}$, $n_b \in \{5, 10\}$

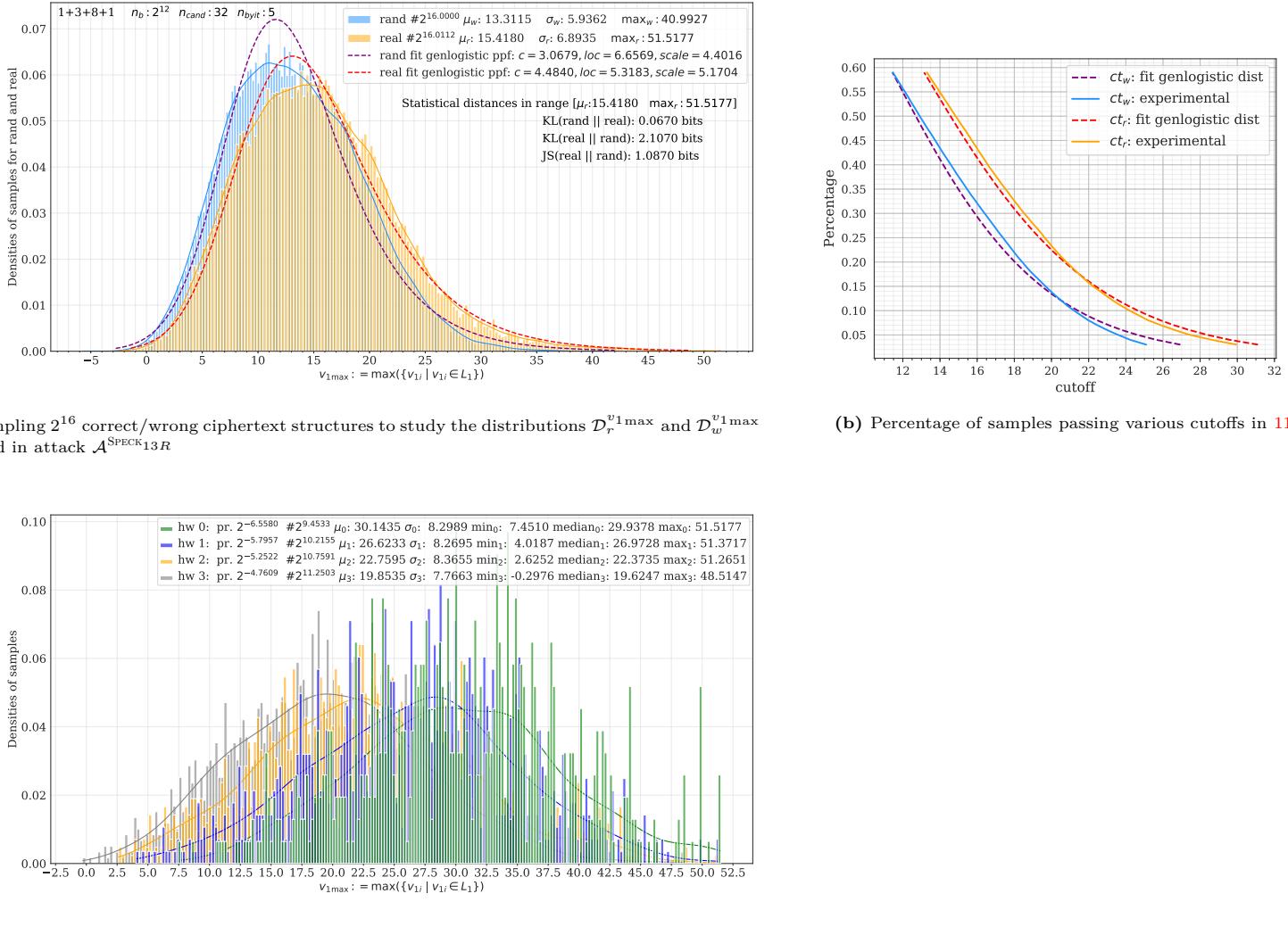
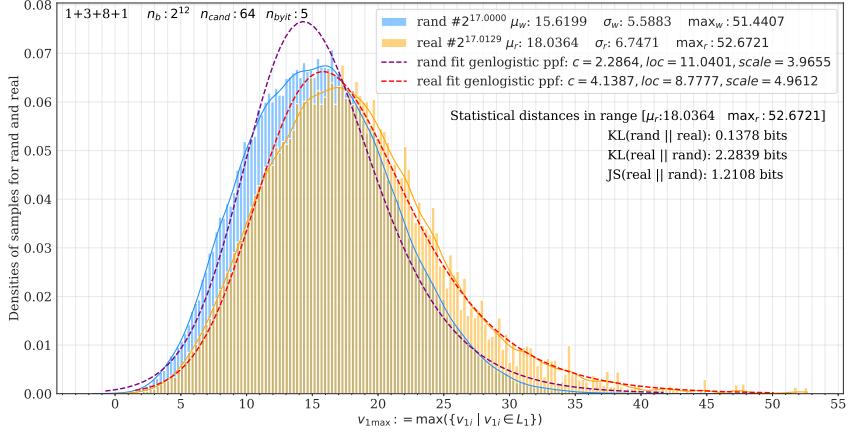
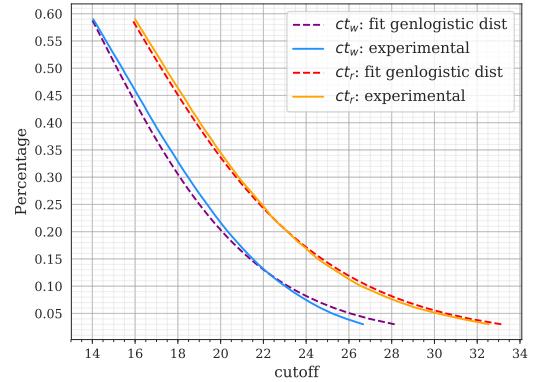


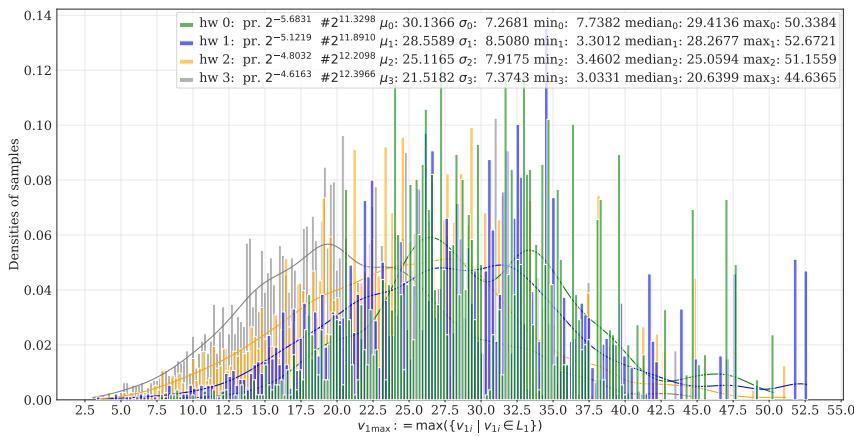
Fig. 11: Distribution of the combined responses on outputs of decrypting one round using recommended subkeys from correct ciphertext structures (**real**) and from wrong ciphertext structures (**rand**)



(a) Sampling 2^{17} correct/wrong ciphertext structures to study the distributions $\mathcal{D}_r^{v_1 \max}$ and $\mathcal{D}_w^{v_1 \max}$ involved in attack $\mathcal{A}_{\text{SPECK}13R}$



(b) Percentage of samples passing various cutoffs in 12a



(c) Distribution of combined responses using correct ciphertext structures when the corresponding recommended subkey has Hamming distance hw with the real subkey

Fig. 12: Distribution of the combined responses on outputs of decrypting one round using recommended subkeys from correct ciphertext structures (**real**) and from wrong ciphertext structures (**rand**)

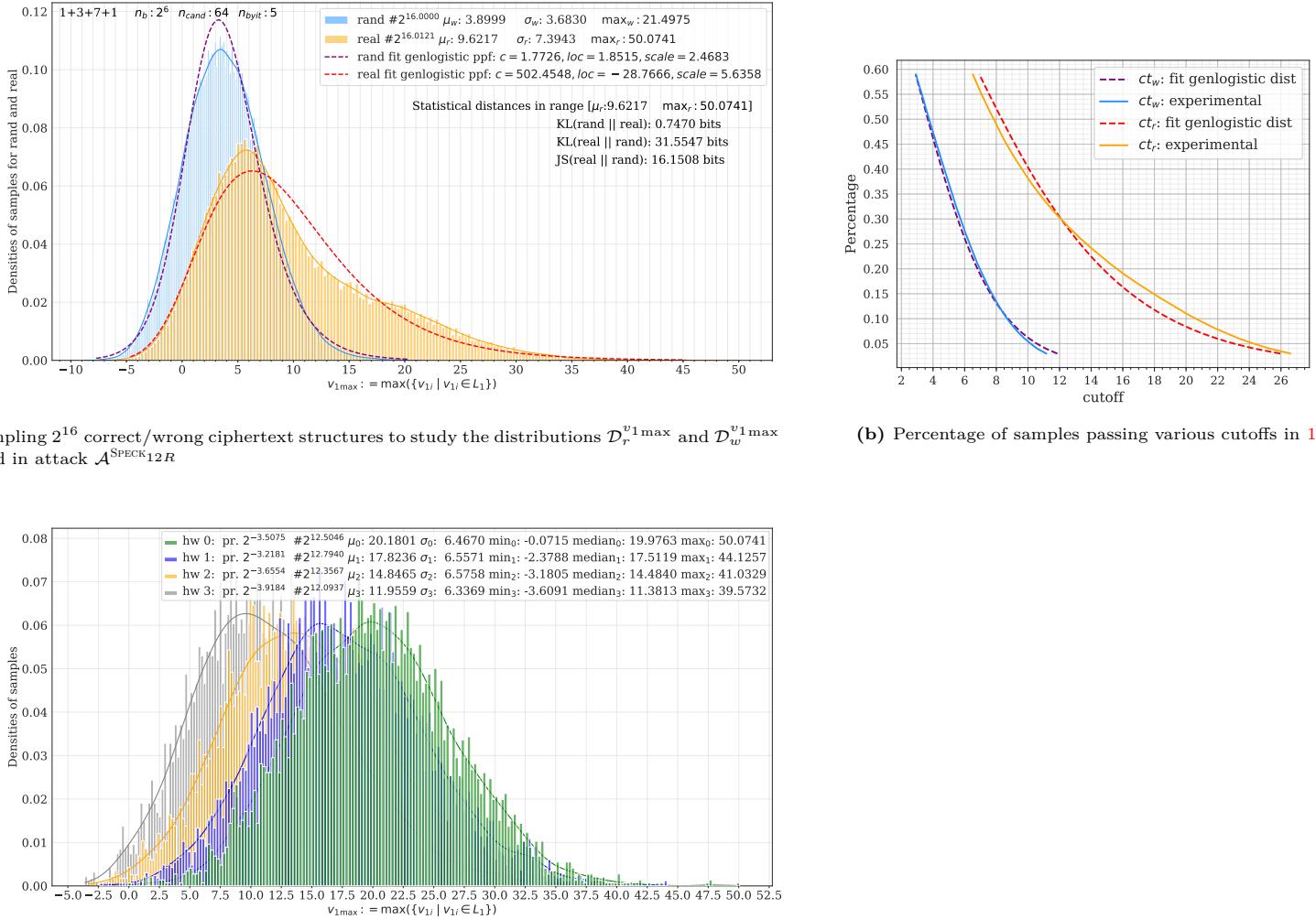


Fig. 13: Distribution of the combined responses on outputs of decrypting one round using recommended subkeys from correct ciphertext structures (**real**) and from wrong ciphertext structures (**rand**)

C Distributions of the Combined-response with Various Number of Blocks

To investigate how many samples from the same distribution should be combined to achieve a good combine-response distinguisher (CRD), that is how many neutral bits are necessary, the accuracy of the CRD and the distributions of the combine-responses were experimentally investigated. The relation curves between the accuracy of the resulting $CRDs$ and the number of combined samples can be seen in Fig. 14. The resulting distributions are illustrated using histogram plots. Besides, parameters of the best fitting distributions among {Normal, Chi-squared, Generalized logistic, Logistic, Gamma} are provided.

From Figures 15, 16, 17, 18, for Gohr’s lightweight 5-, 6-, 7-, 8-round neural distinguishers (with accuracy listed in Table 10), the distributions corresponding to wrong and correct ciphertext structures can be separated when combining 2^0 , $2^3 \sim 2^4$, $2^6 \sim 2^7$, and $2^{12} \sim 2^{13}$ samples, respectively. Combining that many samples, the accuracy of the resulting $CRDs$ can be higher than 0.95, which can be seen in Fig. 14. Thus, for using these distinguishers to do key-recovery and when there are only several correct ciphertext structures could occur, roughly, one needs to exploit about 0, 3 \sim 4, 6 \sim 7, and 12 \sim 13 neutral bits.

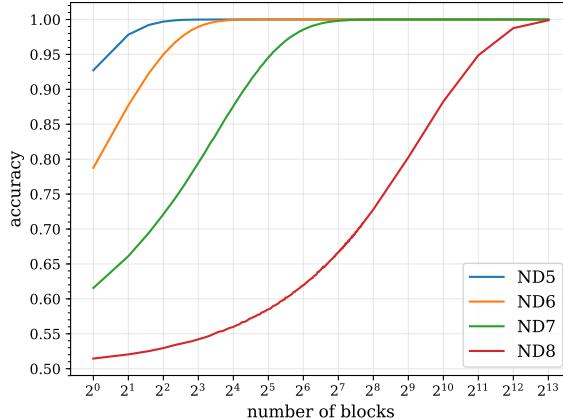


Fig. 14: The accuracy curve when combining a various number of samples from the same distribution

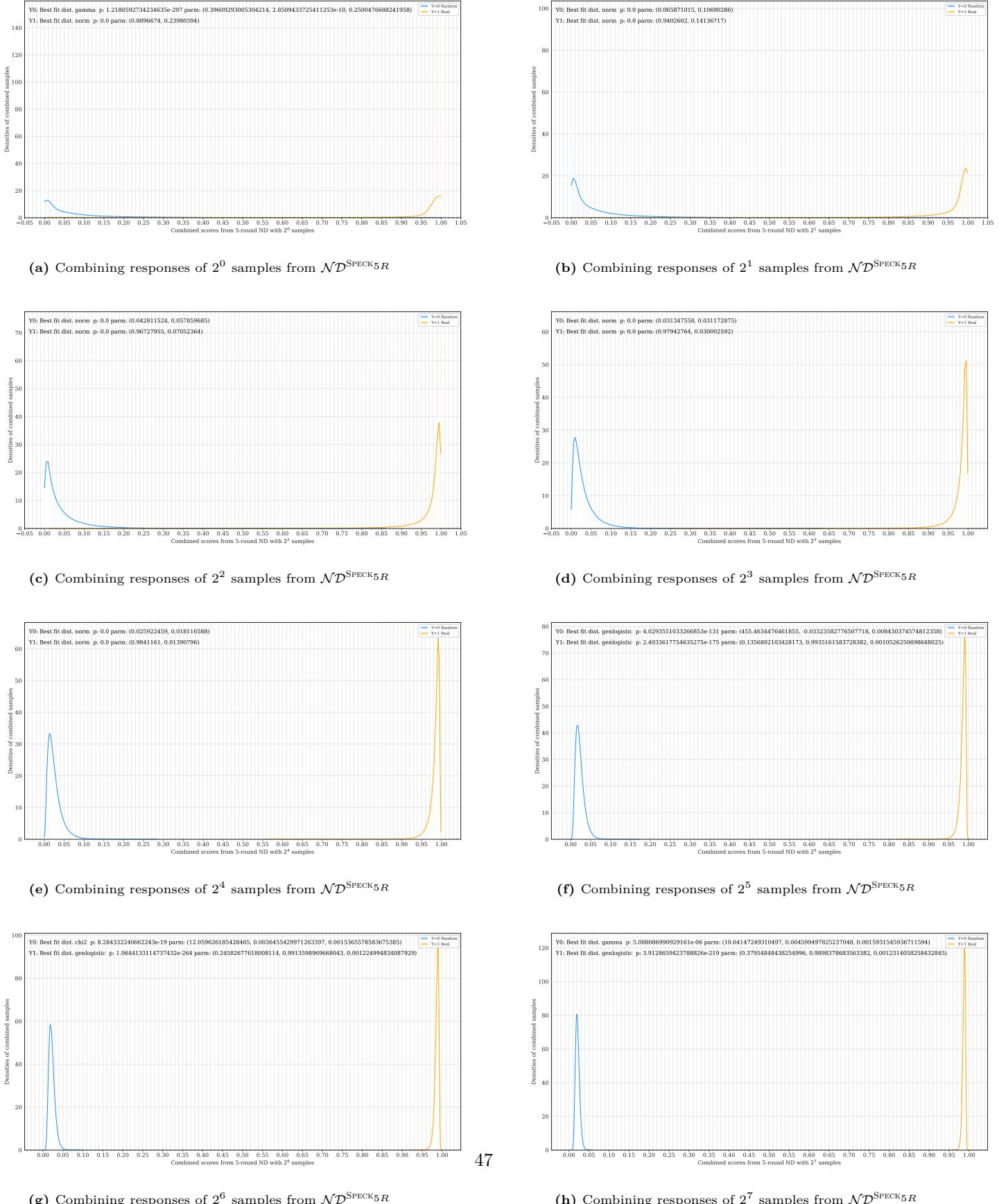
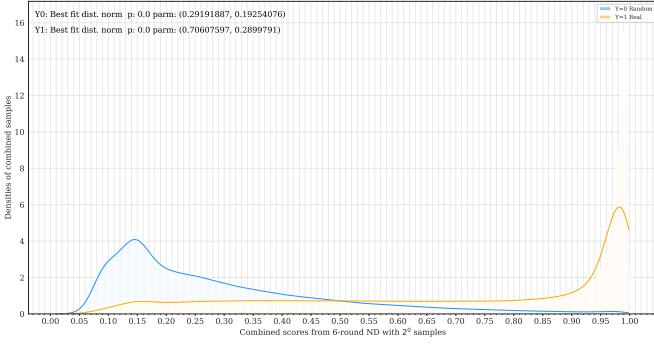
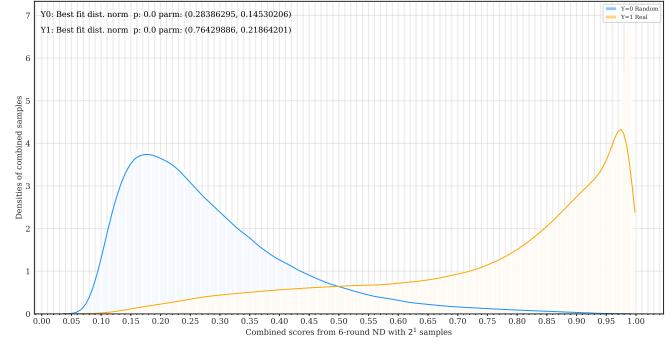


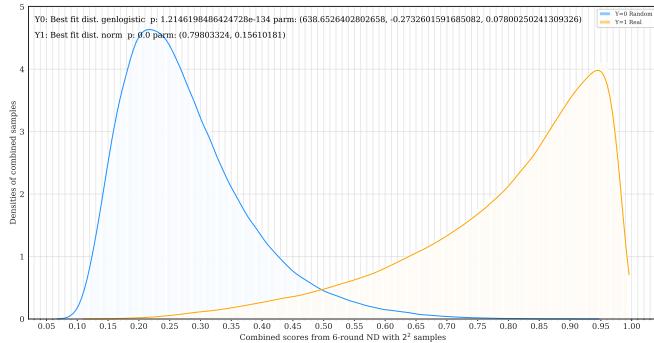
Fig. 15: The distribution of combined responses from $\mathcal{ND}^{\text{SPECK5R}}$ (sampled with 2^{20} combined ciphertext-pairs)



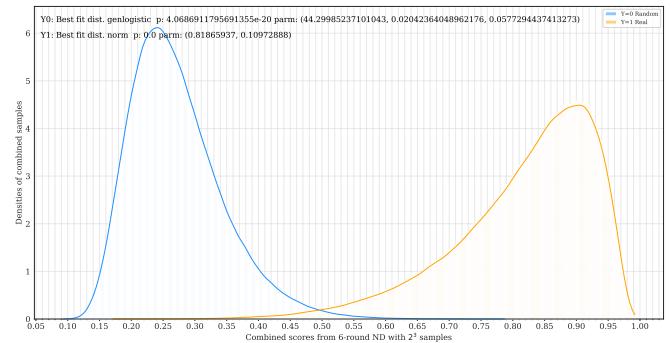
(a) Combining responses of 2^0 samples from $\mathcal{ND}^{\text{SPECK6R}}$



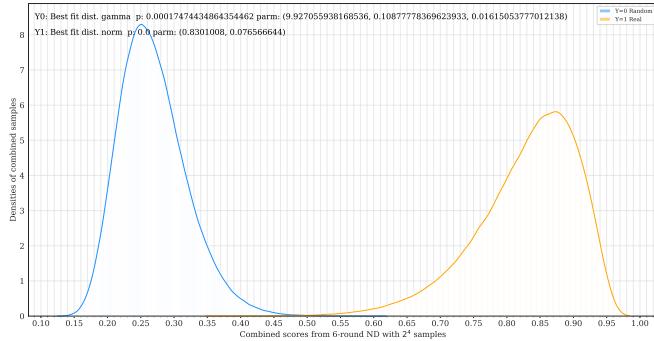
(b) Combining responses of 2^1 samples from $\mathcal{ND}^{\text{SPECK6R}}$



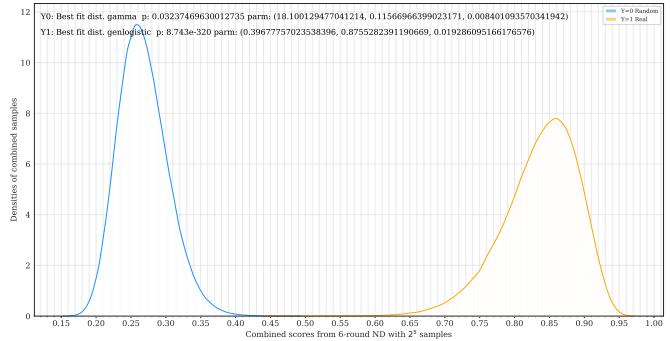
(c) Combining responses of 2^2 samples from $\mathcal{ND}^{\text{SPECK6R}}$



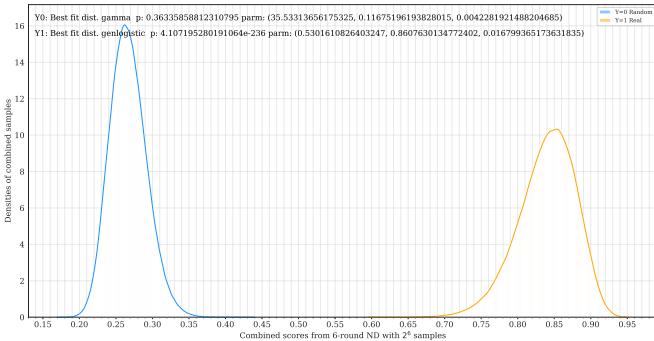
(d) Combining responses of 2^3 samples from $\mathcal{ND}^{\text{SPECK6R}}$



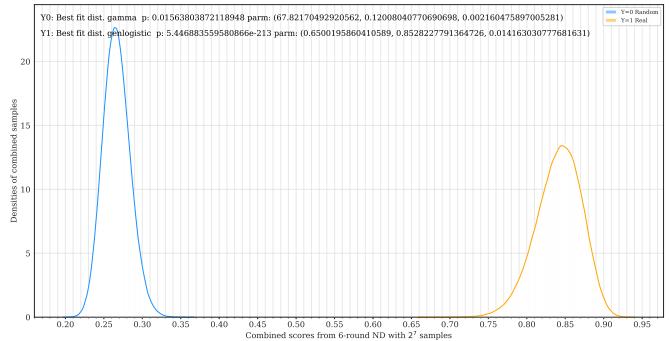
(e) Combining responses of 2^4 samples from $\mathcal{ND}^{\text{SPECK6R}}$



(f) Combining responses of 2^5 samples from $\mathcal{ND}^{\text{SPECK6R}}$

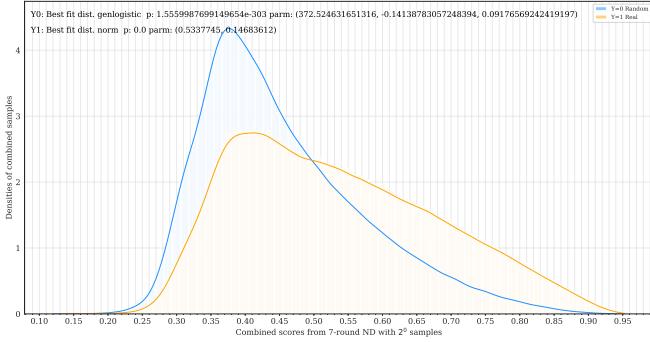


(g) Combining responses of 2^6 samples from $\mathcal{ND}^{\text{SPECK6R}}$

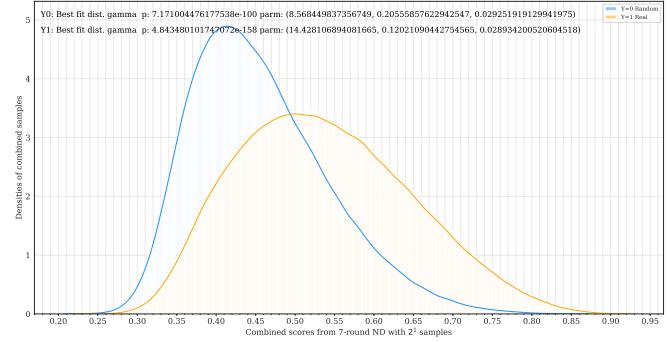


(h) Combining responses of 2^7 samples from $\mathcal{ND}^{\text{SPECK6R}}$

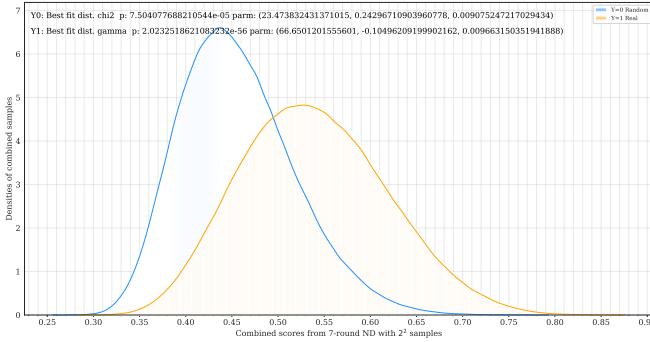
Fig. 16: The distribution of combined responses from $\mathcal{ND}^{\text{SPECK6R}}$ (sampled with 2^{20} combined ciphertext-pairs)



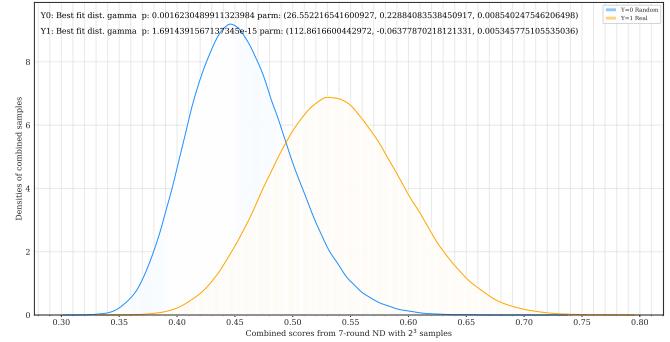
(a) Combining responses of 2^0 samples from $\mathcal{ND}^{\text{SPECK7R}}$



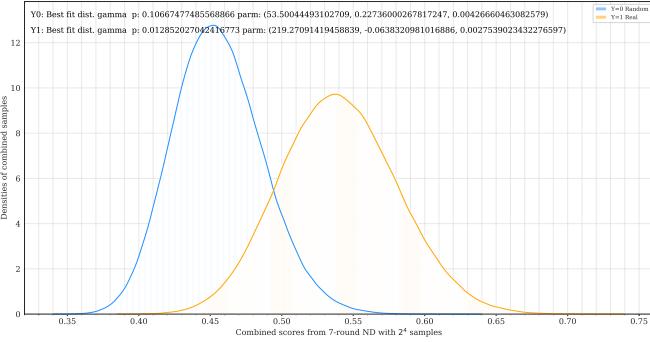
(b) Combining responses of 2^1 samples from $\mathcal{ND}^{\text{SPECK7R}}$



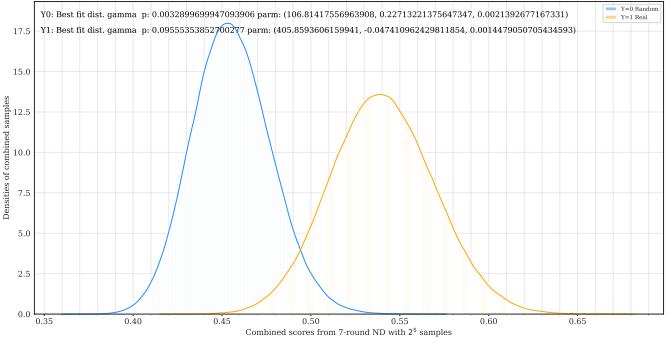
(c) Combining responses of 2^2 samples from $\mathcal{ND}^{\text{SPECK7R}}$



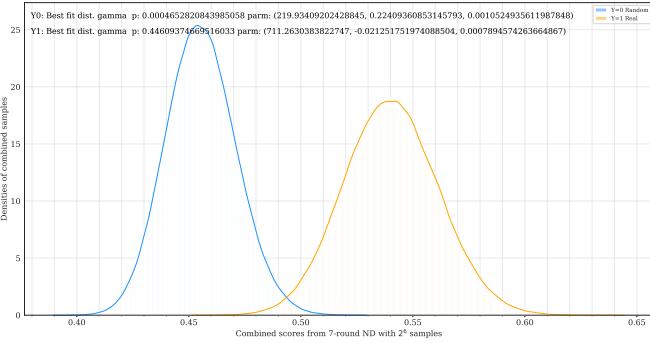
(d) Combining responses of 2^3 samples from $\mathcal{ND}^{\text{SPECK7R}}$



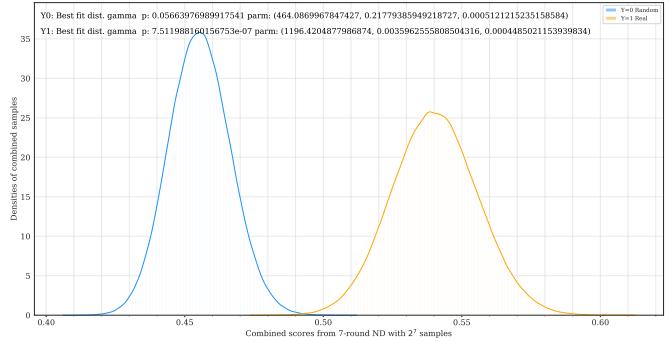
(e) Combining responses of 2^4 samples from $\mathcal{ND}^{\text{SPECK7R}}$



(f) Combining responses of 2^5 samples from $\mathcal{ND}^{\text{SPECK7R}}$

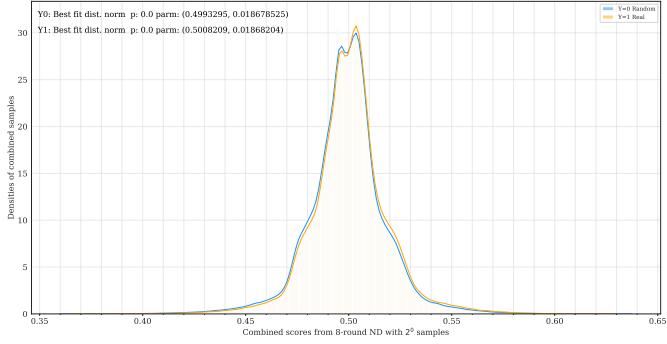


(g) Combining responses of 2^6 samples from $\mathcal{ND}^{\text{SPECK7R}}$

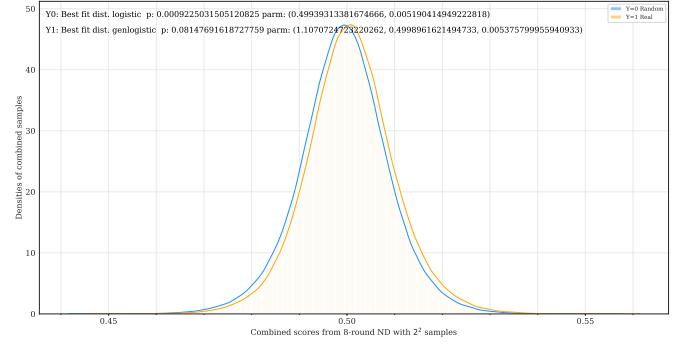


(h) Combining responses of 2^7 samples from $\mathcal{ND}^{\text{SPECK7R}}$

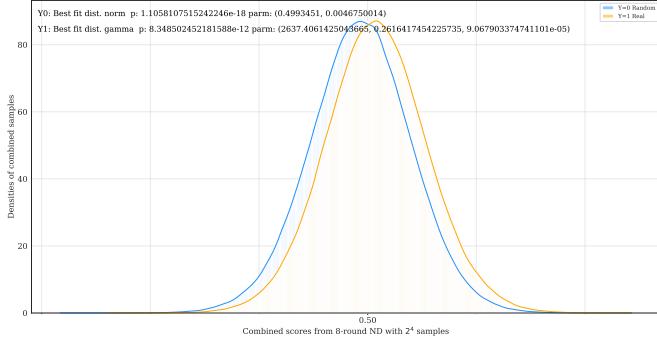
Fig. 17: The distribution of combined responses from $\mathcal{ND}^{\text{SPECK7R}}$ (sampled with 2^{20} combined ciphertext-pairs)



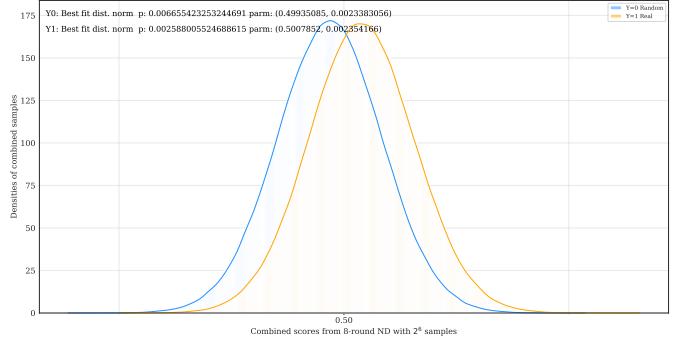
(a) Combining responses of 2^0 samples from $\mathcal{ND}^{\text{SPECK8R}}$



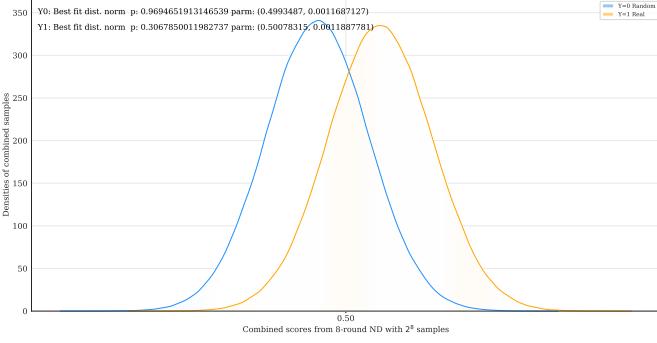
(b) Combining responses of 2^2 samples from $\mathcal{ND}^{\text{SPECK8R}}$



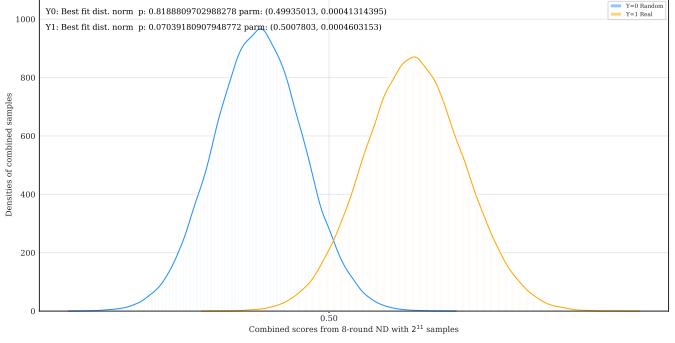
(c) Combining responses of 2^4 samples from $\mathcal{ND}^{\text{SPECK8R}}$



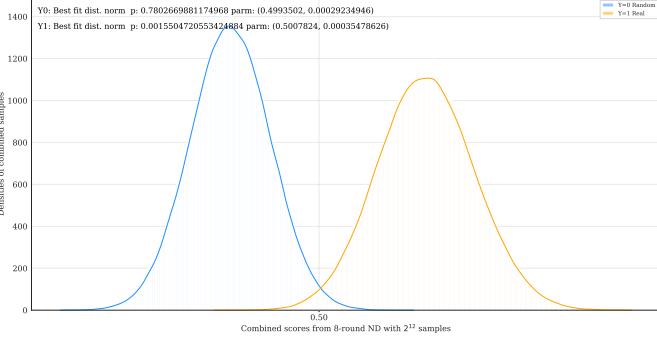
(d) Combining responses of 2^6 samples from $\mathcal{ND}^{\text{SPECK8R}}$



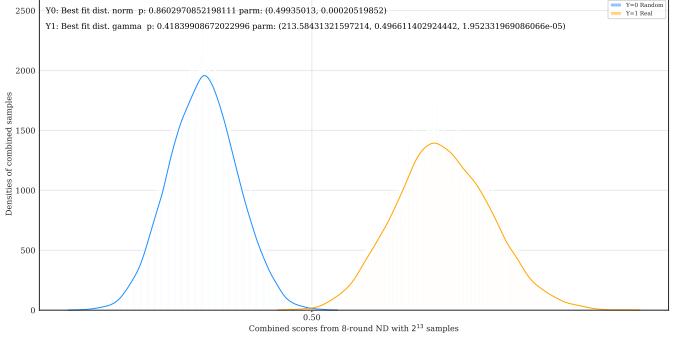
(e) Combining responses of 2^8 samples from $\mathcal{ND}^{\text{SPECK8R}}$



(f) Combining responses of 2^{11} samples from $\mathcal{ND}^{\text{SPECK8R}}$



(g) Combining responses of 2^{12} samples from $\mathcal{ND}^{\text{SPECK8R}}$



(h) Combining responses of 2^{13} samples from $\mathcal{ND}^{\text{SPECK8R}}$

Fig. 18: The distribution of combined responses from $\mathcal{ND}^{\text{SPECK8R}}$ (sampled with 2^{20} combined ciphertext-pairs when combining no more than 2^{10} , 2^{18} for combining $2^{10} \sim 2^{12}$, and 2^{15} for combining 2^{13} responses)

D Additional Constraints on Differential Trails of SPECK32/64

During this work, additional constraints beyond the XOR-differences on some differential trails of SPECK32/64 were found. Moreover, unexpectedly, in some differential trails, constraints are on subkeys. In such a situation, the attacks using differentials whose major contributed trails must fulfill these constraints on keys can only work for a fraction of the keyspace.

This happens to the presented attacks that use the 3-round differentials $(0x8020, 0x4101) \rightarrow (0x0040, 0x0000)$, $(0x8060, 0x4101) \rightarrow (0x0040, 0x0000)$, $(0x8021, 0x4101) \rightarrow (0x0040, 0x0000)$, and $(0x8061, 0x4101) \rightarrow (0x0040, 0x0000)$. Note that this also happens to previous best differential attacks on SPECK32/64, including those covering the most rounds (14-round) in [10, 13, 29], which was not noticed before.

Table 7 presents the generalized constraints of the differential trails involved in the proposed attacks (and other potentially useful trails), and Table 8 presents those used in previous attacks [10, 13, 29]. These constraints can be obtained using the existing tool ARXToolkit from [25]. A recent work in [8] also independently found this key-dependent phenomenon of the differential on SPECK and provided a theoretical treatment.

For the used 3-round differential trails, there is 1 bit constraint on one subkey (all four 3-round differential trails share this constraint), as shown in Table 7. In the 11-round attack in [10] and the 14-round attacks in [13, 29], the best 9-round trail was used. In this trail, there is 1 bit constraint on one subkey, as shown in Table 8. In [29], the best 10-round differential trail was found. In this trail, there are 3 bits constraints on the subkeys as shown in Table 8.

Since the probabilities of these trails are calculated using the Markov model, and it is averaged over the whole keyspace, the real probability of these trails should be about 2^c times larger than the previous estimation for 2^{64-c} keys, and 0 for other keys, where c is the number of constraints on the subkeys. Accordingly, the used 3-round differential trails whose probability was previously estimated as 2^{-12} should have a probability 2^{-11} for 2^{63} keys and 0 for other keys. Similarly, the 9-round differential trail, whose probability was previously estimated as 2^{-30} used in [10, 13, 29] should be about 2^{-29} for 2^{63} keys and 0 for other keys; the 10-round differential trail, whose probability is previously estimated as 2^{-35} found in [29] should be about 2^{-32} for 2^{61} keys and 0 for other keys.

Additionally, we found that for one of the best differential trails with input difference $0x0040/0000$ of 8-round SPECK32/64, there are additional constraints on subkeys (see Table 7). For the corresponding keys that fulfill constraint $k_1[9] = k_1[8]$, we trained a 7-round ResNet neural distinguisher with an accuracy of 0.6228 (while for keys such that $k_1[9] \neq k_1[8]$, the resulted neural distinguisher has an accuracy 0.6164). This indicates the existence of better weak-key neural distinguishers.

Table 7: Generalized differences of the 3-round, 4-round, and 8-round differential trails used/involved in the attacks (obtained using tools in [25, 30])

3-round				4-round				8-round			
R	differences hex($x_i y_i$)	vars	generalized differences	R	differences hex($x_i y_i$)	vars	generalized differences	R	differences hex($x_i y_i$)	vars	generalized differences
0	8020 4101	x_0	x-----0-x-----	0	1488 1008	x_0	-----x-x-x-x-x-----	0	0040 0000	x_0	-----x-----
		y_0	-x-----x-----x			y_0	-----x-----x-0			y_0	-----
		z_0	-----x-----x			z_0	-----x-----x-----x-----			z_0	x-----
		k_0	-----			k_0	-----			k_0	-----
		x_1	-----x-----x			x_1	-----0-x-----x			x_1	x-----
		y_1	-----<x-----x--			y_1	-x-----!-----x			y_1	x-----
		z_1	----->x-----c-----0-----			z_1	----->x-----x			z_1	x----->x-----
		k_1	-----!			k_1	-----=-----			k_1	-----=-----
1	0201 0604	x_2	-----<x-----	1	0021 4001	x_2	----->x-----x	1	8000 8000	x_2	x----->x-----
		y_2	-----=-----x-----			y_2	-----<x-----x-----			y_2	x-----xx-----x-----
		z_2	-----x-----			z_2	----->x-----c-----1-----			z_2	x-----><x-----f-----0-----
		k_2	-----			k_2	-----!			k_2	-----4-----
2	1800 0010	x_3	-----x-----	2	0601 0604	x_3	-----<x-----	2	8300 8302	x_3	x-----xx-----x-----
		y_3	-----			y_3	-----=-----x-----			y_3	x-----x-----x-----0-----
		z_3	-----			z_3	-----x-----			z_3	x-----xx-----x-----1-----0-----
		k_3	-----			k_3	-----			k_3	-----
		x_4	-----x-----			x_4	-----xx-----0-----0-----x-----8-----			x_4	x-----xx-----0-----0-----x-----8-----
		y_4	-----			y_4	-----x-----<x-----x-----			y_4	x-----x-----<x-----x-----
		z_4	-----			z_4	-----x-----0-----x-----			z_4	-----x-----0-----x-----
		k_4	-----			k_4	-----			k_4	-----
3	0040 0000	x_5	-----x-----x-----	3	1800 0010	x_5	-----x-----x-----	3	8e00 820a	x_5	-----x-----x-----
		y_5	-----x-----x-----			y_5	-----x-----x-----			y_5	-----x-----x-----
		z_5	-----			z_5	-----x-----			z_5	-----x-----
		k_5	-----			k_5	-----			k_5	-----
		x_6	-----x-----			x_6	-----x-----			x_6	-----x-----
		y_6	-----x-----			y_6	-----x-----			y_6	-----x-----
		z_6	-----x-----			z_6	-----x-----			z_6	-----x-----
		k_6	-----			k_6	-----			k_6	-----
		x_7	-----x-----4-----0-----			x_7	-----x-----4-----0-----			x_7	-----x-----4-----0-----
		y_7	-----xx-----x-----			y_7	-----xx-----x-----			y_7	-----xx-----x-----
		z_7	-----x-----x-----x-----x-----			z_7	-----x-----x-----x-----x-----			z_7	-----x-----x-----x-----x-----
		k_7	-----			k_7	-----			k_7	-----
		x_8	-----x-----x-----x-----			x_8	-----x-----x-----x-----			x_8	-----x-----x-----x-----
		y_8	-----x-----x-----x-----xx-----			y_8	-----x-----x-----x-----xx-----			y_8	-----x-----x-----x-----xx-----
		z_8	-----			k_8	-----			k_8	-----
		x_9	-----			y_9	-----			z_9	-----
		k_9	-----			x_{10}	-----			y_{10}	-----
		y_{11}	-----			z_{11}	-----			k_{11}	-----
		x_{12}	-----			y_{12}	-----			z_{12}	-----
		k_{12}	-----			x_{13}	-----			y_{13}	-----
		y_{14}	-----			z_{14}	-----			k_{14}	-----
		x_{15}	-----			y_{15}	-----			z_{15}	-----
		k_{15}	-----			x_{16}	-----			y_{16}	-----
		y_{17}	-----			z_{17}	-----			k_{17}	-----
		x_{18}	-----			y_{18}	-----			z_{18}	-----
		k_{18}	-----			x_{19}	-----			y_{19}	-----
		y_{20}	-----			z_{20}	-----			k_{20}	-----
		x_{21}	-----			y_{21}	-----			z_{21}	-----
		k_{21}	-----			x_{22}	-----			y_{22}	-----
		y_{23}	-----			z_{23}	-----			k_{23}	-----
		x_{24}	-----			y_{24}	-----			z_{24}	-----
		k_{24}	-----			x_{25}	-----			y_{25}	-----
		y_{26}	-----			z_{26}	-----			k_{26}	-----
		x_{27}	-----			y_{27}	-----			z_{27}	-----
		k_{27}	-----			x_{28}	-----			y_{28}	-----
		y_{29}	-----			z_{29}	-----			k_{29}	-----
		x_{30}	-----			y_{30}	-----			z_{30}	-----
		k_{30}	-----			x_{31}	-----			y_{31}	-----
		y_{32}	-----			z_{32}	-----			k_{32}	-----
		x_{33}	-----			y_{33}	-----			z_{33}	-----
		k_{33}	-----			x_{34}	-----			y_{34}	-----
		y_{35}	-----			z_{35}	-----			k_{35}	-----
		x_{36}	-----			y_{36}	-----			z_{36}	-----
		k_{36}	-----			x_{37}	-----			y_{37}	-----
		y_{38}	-----			z_{38}	-----			k_{38}	-----
		x_{39}	-----			y_{39}	-----			z_{39}	-----
		k_{39}	-----			x_{40}	-----			y_{40}	-----
		y_{41}	-----			z_{41}	-----			k_{41}	-----
		x_{42}	-----			y_{42}	-----			z_{42}	-----
		k_{42}	-----			x_{43}	-----			y_{43}	-----
		y_{44}	-----			z_{44}	-----			k_{44}	-----
		x_{45}	-----			y_{45}	-----			z_{45}	-----
		k_{45}	-----			x_{46}	-----			y_{46}	-----
		y_{47}	-----			z_{47}	-----			k_{47}	-----
		x_{48}	-----			y_{48}	-----			z_{48}	-----
		k_{48}	-----			x_{49}	-----			y_{49}	-----
		y_{50}	-----			z_{50}	-----			k_{50}	-----
		x_{51}	-----			y_{51}	-----			z_{51}	-----
		k_{51}	-----			x_{52}	-----			y_{52}	-----
		y_{53}	-----			z_{53}	-----			k_{53}	-----
		x_{54}	-----			y_{54}	-----			z_{54}	-----
		k_{54}	-----			x_{55}	-----			y_{55}	-----
		y_{56}	-----			z_{56}	-----			k_{56}	-----
		x_{57}	-----			y_{57}	-----			z_{57}	-----
		k_{57}	-----			x_{58}	-----			y_{58}	-----
		y_{59}	-----			z_{59}	-----			k_{59}	-----
		x_{60}	-----			y_{60}	-----			z_{60}	-----
		k_{60}	-----			x_{61}	-----			y_{61}	-----
		y_{62}	-----			z_{62}	-----			k_{62}	-----
		x_{63}	-----			y_{63}	-----			z_{63}	-----
		k_{63}	-----			x_{64}	-----			y_{64}	-----
		y_{65}	-----			z_{65}	-----			k_{65}	-----
		x_{66}	-----			y_{66}	-----			z_{66}	-----
		k_{66}	-----			x_{67}	-----			y_{67}	-----
		y_{68}	-----			z_{68}	-----			k_{68}	-----
		x_{69}	-----			y_{69}	-----			z_{69}	-----
		k_{69}	-----			x_{70}	-----			y_{70}	-----
		y_{71}	-----			z_{71}	-----			k_{71}	-----
		x_{72}	-----			y_{72}	-----			z_{72}	-----
		k_{72}	-----			x_{73}	-----			y_{73}	-----
		y_{74}	-----			z_{74}	-----			k_{74}	-----
		x_{75}	-----			y_{75}	-----			z_{75}	-----
		k_{75}	-----			x_{76}	-----			y_{76}	-----
		y_{77}	-----			z_{77}	-----			k_{77}	-----
		x_{78}	-----			y_{78}	-----			z_{78}	-----

Pr_a is the previous estimated probability over all keys, Pr_r is the revisited estimated probability for different keys.

$$-: a_i = a'_i$$

x: $a_i \neq a'_i$

$$0: a_i = a'_i = 0$$

$$1: a_i = a'_i = 1$$

$$! : a'_i = a_i \neq a_{i-1}$$

$$=: a'_i = a_i = a_{i-1}$$

$\Leftarrow a'_i \neq a_i = a_{i-1}$

$>: a_i \neq a_i \neq a_{i-1}$

^c₀: uncommon constraint “c20000c3”

^c₁: uncommon constraint “c30000c2”

^c: uncommon constraint “c2000043”

⁴: uncommon constraint “430000c2”

: uncommon constraint “ff0000f0”

¹: uncommon constraint “ff00000f”

52

⁰₈: uncommon constraint “00824100”

⁰₃: uncommon constraint “00381c00”

: uncommon constraint “00418200”

: uncommon constraint “1400003c”

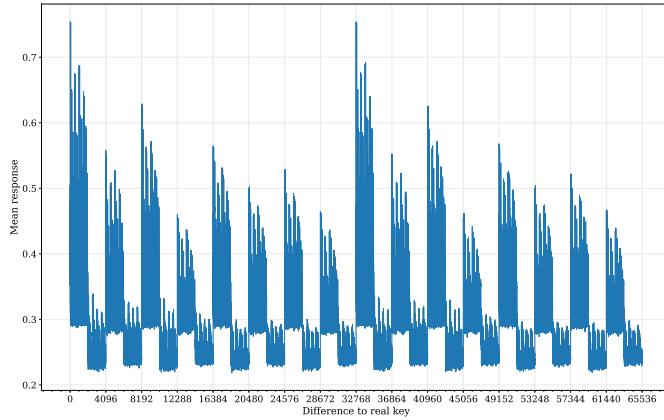
²: uncommon constraint “2800003c”

Constraints in red are on sub-keys.

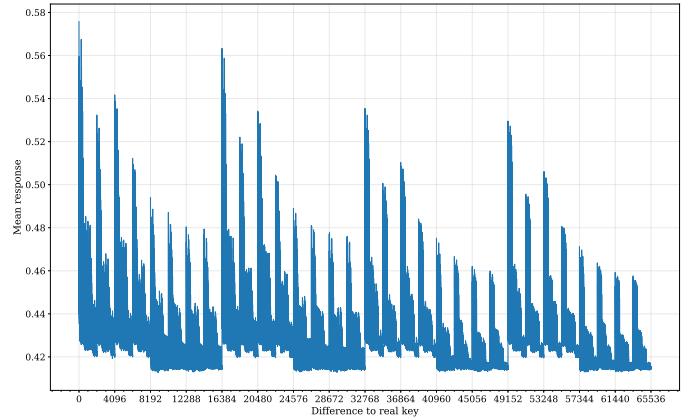
Table 8: Generalized differences of the 9-round and 10-round differential trails used in [10, 29] (obtained using tools in [25])

9-round				10-round					
R	differences hex(x_i y_i)	vars	generalized differences	R	differences hex(x_i y_i)	vars	generalized differences		
0	8054 a900	x_0	$x-----x-x-x-$	0	2040 0040	x_0	$--x-----x-----$		
		y_0	$x-x-x--x-----$			y_0	$-----x-----$		
		z_0	$-----c-----2-----$			z_0	$x-----$		
		k_0	$-----$			k_0	$-----$		
		x_1	$-----f-----1-----$			x_1	$x-----$		
		y_1	$x-x-x-----x-----$			y_1	$x-----x-----$		
		z_1	$x-x-x-----x-----$			z_1	$x-----$		
		k_1	$-----$			k_1	$-----$		
		x_2	$^0_4-x-x-----x-----$			x_2	$x-----f-----!-----$		
		y_2	$--<x-x-----x-----$			y_2	$x-----x!-----x-----$		
1	0000 a402	z_2	$-^0_4-x-^c_2-<x-----$	1	8000 8100	z_2	$x--->x-x-----x-----$		
		k_2	$-----$			k_2	$----=-----$		
		x_3	$-x-x-----xx-----$			x_3	$x--->x-x-----x-----$		
		y_3	$x-----x<x-----0-----$			y_3	$x---<x-x-----x-^f_1-----$		
		z_3	$-----xx-----x-----$			z_3	$-<x-^4_0-----x-----$		
		k_3	$-----$			k_3	$-!-----$		
		x_4	$-----xx-----x-----$			x_4	$->x-----x-----$		
		y_4	$-----x-----xx-----$			y_4	$-!-x-x-----x-----$		
		z_4	$----->x01-----$			z_4	$---x-----xx-----$		
		k_4	$-----!-----$			k_4	$-----$		
2	a402 3408	x_5	$-----<x-----$	2	8000 8402	x_5	$-!-x-----xx-----$		
		y_5	$-----x-----$			y_5	$x-----x>x-----$		
		z_5	$-----x-----$			z_5	$-----><x-----$		
		k_5	$-----$			k_5	$-----!-----$		
		x_6	$-----x-----$			x_6	$-----x>x-----$		
		y_6	$-----$			y_6	$-----=====x-----$		
		z_6	$-----x-----$			z_6	$-----x-0-----$		
		k_6	$-----$			k_6	$-----$		
		x_7	$-----x-----$			x_7	$-----x-----$		
		y_7	$-----x-----$			y_7	$-----$		
3	50c0 80e0	z_7	$x-----x-----$	3	8d02 9d08	z_7	$-----x-----$		
		k_7	$-----$			k_7	$-----$		
		x_8	$x-----x-----$			x_8	$-----x-----$		
		y_8	$x-----x-x-----$			y_8	$-----x-----$		
		z_8	$-----x-----$			z_8	$-----x-----x-----$		
		k_8	$-----$			k_8	$-----$		
		x_9	$-----x-----$			x_9	$-----x-----x-----$		
		y_9	$-----x-x-x-----x-----$			y_9	$-----x-x-----x-----$		
						z_9	$-----x-----$		
						k_9	$-----$		
4	0181 0203			4	6002 1420	x_{10}	$-----x-----$		
						y_{10}	$x-x-x-----x-----$		
5	000c 0800			5	1060 40e0				
6	2000 0000			6	0380 0001				
7	0040 0040			7	0004 0000				
8	8040 8140			8	0800 0800				
9	0040 0542			9	0810 2810				
$Pr_a = 2^{-30}$				$Pr_a = 2^{-35}$					
$Pr_r = \begin{cases} 2^{-29} & \text{for } 2^{63} \text{ keys} \\ 0 & \text{for others} \end{cases}$				$Pr_r = \begin{cases} 2^{-32} & \text{for } 2^{61} \text{ keys} \\ 0 & \text{for others} \end{cases}$					

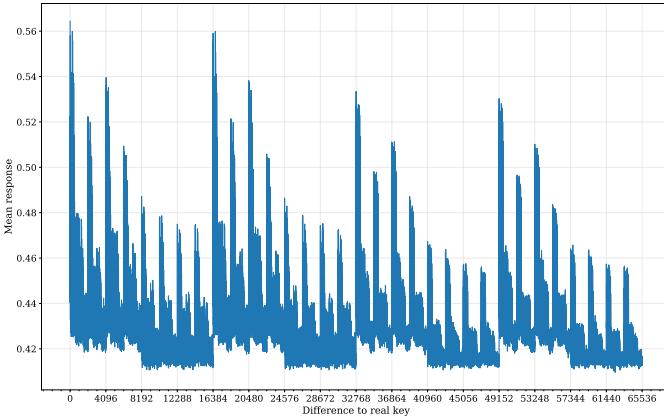
Symbols used are same as in Table 7



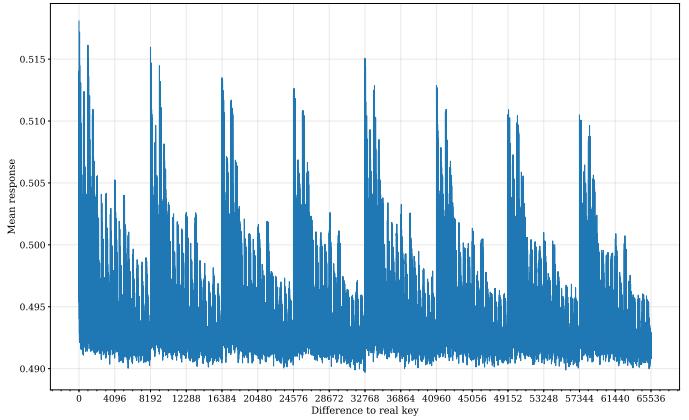
(a) $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON8}R}$: directly trained with data of the form $((x, y, x', y'))$



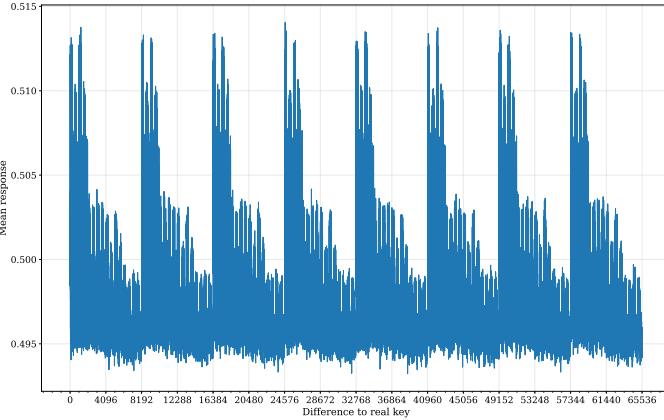
(b) $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON8}R}$: directly trained with data of the form $((x, x', y \oplus y'))$



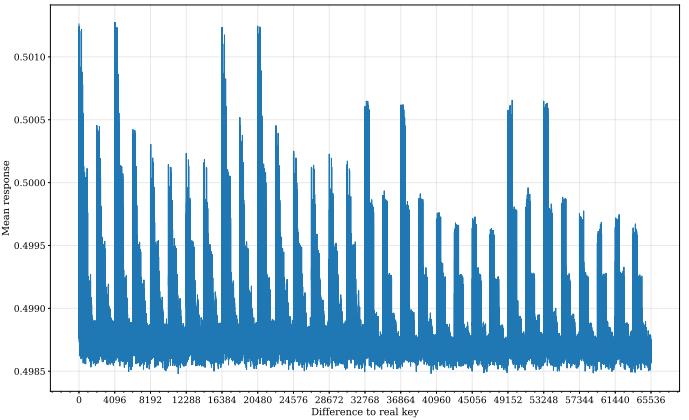
(c) $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON9}R}$: directly trained with data of the form $((x, y, x', y'))$



(d) $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON9}R}$: directly trained with data of the form $((x, x', y \oplus y'))$



(e) $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON10}R}$: trained using $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON9}R}$ and KeyAveraging algorithm



(f) $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON11}R}$: trained using $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON9}R}$ and $\mathcal{DD}_{(0440,0100)}^{\text{SIMON8}R}$ in staged training method

Fig. 19: Wrong key response profile (only μ_δ shown) for neural distinguishers on SIMON32/64 (used 2^{14} ciphertexts for (a-e) and 2^{18} for (f))
54

E Key-recovery Attacks on Round-Reduced SIMON32/64

E.1 Wrong Key Response Profile for the Neural Distinguishers on SIMON32/64

Figure 19 shows the wrong key response profiles of the trained \mathcal{ND} s for SIMON32/64. From Fig. 19, the wrong key response profiles of $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{8R}}$ (Fig. 19b) and that of $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{9R}}$ (Fig. 19c) share observable patterns and symmetry. For key values that have little different from the real value, responses from $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{8R}}$ are higher than responses from $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{9R}}$. Similar observations can be derived from a comparison between that of $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{9R}}$ (Fig. 19d) and $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{10R}}$ (Fig. 19e).

E.2 An Attack on 16-round SIMON32/64

Under a similar framework to the key-recovery attacks on SPECK32/64, the trained neural distinguishers can be prepended with a classical differential to perform key-recovery attacks. This section presents such an attack, named $\mathcal{A}_I^{\text{SIMON}_{16R}}$, which combines the longest but weak \mathcal{ND} with a differential that has many SNBSs.

The classical component in the attack $\mathcal{A}_I^{\text{SIMON}_{16R}}$ presented in the sequel is a 3-round differential $(0x0440, 0x1000) \rightarrow (0x0000, 0x0040)$ (prob. $\approx 2^{-8}$).

Similar to attacks on SPECK32/64, to obtain decent scores from the responses of the \mathcal{ND} s, combined responses from the \mathcal{ND} s over a number of samples from the same distribution are to be used. Thus, to obtain enough samples from the same distribution, neutral bits of the prepended \mathcal{CD} are exploited.

E.3 Finding Neutral Bits for the Classical Differentials

Finding SNBSs for 3-round Differential. For the 3-round differential to be prepended to the \mathcal{ND} s, one can obtain all neutral bits and SNBSs (simultaneously complementing up to 4 bits) using the following algebraic method.

Given the input and output differences $(0x0440, 0x1000)$ and $(0x0000, 0x0040)$, one can build the non-linear equations on the derivative functions. Because the degrees of the derivative functions corresponding to this 3-round differential is low (*i.e.*, 4), this system of non-linear equations can be solved by computing the reduced Gröbner basis, which can be done using the PolyBoRi library integrated in SageMath [11]. The reduced Gröbner basis is a unique representation for the ideal with respect to a fixed monomial order. Thus, the computed is a unique representation corresponding to the variety of the equations defined by the derivative function corresponding to the differential. Thus, not changing the reduced Gröbner basis is a necessary and sufficient condition for being an NB/SNBS. Accordingly, to find SNBSs (including single-bit NBs), the following method is used. For each of the 41448 sets (*i.e.*, $32 + 496 + 4960 + 35960$ sets) of at most four bits, in the resulting reduced Gröbner basis, replace this set of

variables with their complements simultaneously; if the Gröbner basis does not change, the set of variables corresponds to an SNBS.

Using the above algebraic method and experimental double-verification, all the single-bit NBs and SNBSs are obtained. There are 9 single-bit NBs {[2], [3], [4], [6], [8], [9], [10], [18], [22]}, 2 2-SNBSs {[0, 24], [12, 26]} (actually, there are 38 2-SNBSs; but 36 out 38 are formed by combinations of the 9 single-bit NBs); all 3-SNBSs and 4-SNBSs are formed by combinations of the 9 single-bit NBs and 2 2-SNBSs. Thus, there are 11 independent single-bit NBs and SNBSs in total.

From the resulting Gröbner basis (also observed by experiments), for an input pair $((x, y), (x', y'))$ to conform to the 3-round differential $(0x0440, 0x1000) \rightarrow (0x0000, 0x0040)$, one has $\begin{cases} x[1] = x'[1] = 0, \\ x[3] = x'[3] = 0. \end{cases}$ (2)

E.4 Key Recovery Attack on 16-round SIMON32/64

The components of $\mathcal{A}_I^{\text{SIMON}_{16R}}$ are as follows (refer to Fig. 4).

1. A 3-round \mathcal{CD} $(0x0440, 0x1000) \rightarrow (0x0000, 0x0040)$ (refer to the rounds colored in blue in Fig. 4), and a set of its 11 NBs {[2], [3], [4], [6], [8], [9], [10], [18], [22], [0, 24], [12, 26]};
2. A 11-round $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{11R}}$ trained using the staged approach under difference $(0x0000, 0x0040)$, and its wrong key response profiles $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{11R}}.\mu$ and $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{11R}}.\sigma$.
3. A 9-round $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{9R}}$ trained under difference $(0x0000, 0x0040)$ and fed with data of type $(x, x', y \oplus y')$, and its wrong key response profiles $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{9R}}.\mu$ and $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{9R}}.\sigma$.

The goal is to recover the last two subkeys k_{15} and k_{14} . A difference with the attack $\mathcal{A}^{\text{SPECK}_{13R}}$ is that, as one of the $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{9R}}$ accepts data of type $(x, x', y \oplus y')$, after guessing k_{15} and k_{14} and decrypting a ciphertext pair to $(x_{14}, y_{14}), (x'_{14}, y'_{14})$, one can compute $(x_{13}, x'_{13}, y_{13} \oplus y'_{13})$ by inverting one round with 0 as the subkey, and thus can be feed to $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{9R}}$.

At the beginning, we guess two key bits of k_0 , that is $k_0[1]$ and $k_0[3]$, because for the 3-round differential, the conditions for correct pairs are $x_1[1] = x'_1[1] = 0$ and $x_1[3] = x'_1[3] = 0$ (refer to Eq. 2); no more key bits need to be guessed because the number of non-conditional neutral bits is enough). Thus, n_{kg} is 2, and there are 2^2 outermost loops.

The framework of attack $\mathcal{A}_I^{\text{SIMON}_{16R}}$ is the same as that of $\mathcal{A}^{\text{SPECK}_{13R}}$ on SPECK32/64 (refer to Fig. 2). The concrete parameters of the attack are as follows. The accuracy of $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{11R}}$ is 0.5173, and that of $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{9R}}$ is 0.5629.

$$\begin{aligned} n_{kg} &= 2^2, & n_{cts} &= 2^7, & n_b &= 2^{11}, & n_{it} &= 2^9 \\ c_1 &= 25, & c_2 &= 100, & n_{byit1} &= n_{byit2} = 5, & n_{cand1} &= n_{cand2} = 32 \end{aligned}$$

The data complexity is $n_{kg} \times n_{cts} \times n_b \times 2$, that is, 2^{21} plaintexts. To examine the performance of the attack, experiments are done on the same GPU server testing

$\mathcal{A}^{\text{SPECK}_{13R}}$. In total 99 trials are run. Within the 99 trials, all trials have correct ciphertext pairs and all called the \mathcal{ND} s. There are 49 success trials, for which the returned last two subkeys have a Hamming distance to the real subkeys of at most two. Thus, the success rate is computed as 49/99, *i.e.*, 0.49.

The 99 (+7) trials took 78 core hours in total. For a trial, it shows that running full 512 iterations requires less than 1 hour. Thus, the worst case to run 2^2 outermost loops (on guessed values of $k_0[0]$ and $k_0[3]$) for a full attack takes less than 4 GPU hours.

F Details of the Key-recovery Attack in [15]

F.1 Neutral bits Used in [14, 15]

Table 9: (Probabilistic) single-bit neutral bits for 2-round differential $(0x0211, 0x0a04) \rightarrow (0x0040, 0x0000)$ of SPECK32/64 used in the 11-round [15] and/or 12-round [14] attacks by Gohr.

NBs	Pr.										
[20]	1	[21]	1	[22]	1	[14]	0.965	[15]	0.938	[23]	0.812
[30]	0.809	[0]	0.763	[8]	0.664	[24]	0.649	[31]	0.644	[1]	0.574

F.2 Accuracy of the Neural Distinguishers on SPECK32/64 in [15]

Table 10: Accuracy of Gohr’s neural distinguishers on SPECK32/64 [15]

#R	Name	Accuracy	True Positive Rate	True Negative Rate
5	$\mathcal{DD}^{\text{SPECK}_5R}$	0.911	0.877	0.947
5	$\mathcal{ND}^{\text{SPECK}_5R}$	$0.929 \pm 5.13 \times 10^{-4}$	$0.904 \pm 8.33 \times 10^{-4}$	$0.954 \pm 5.91 \times 10^{-4}$
6	$\mathcal{DD}^{\text{SPECK}_6R}$	0.758	0.680	0.837
6	$\mathcal{ND}^{\text{SPECK}_6R}$	$0.788 \pm 8.17 \times 10^{-4}$	$0.724 \pm 1.26 \times 10^{-3}$	$0.853 \pm 1.00 \times 10^{-3}$
7	$\mathcal{DD}^{\text{SPECK}_7R}$	0.591	0.543	0.640
7	$\mathcal{ND}^{\text{SPECK}_7R}$	$0.616 \pm 9.70 \times 10^{-4}$	$0.533 \pm 1.41 \times 10^{-3}$	$0.699 \pm 1.30 \times 10^{-3}$
8	$\mathcal{DD}^{\text{SPECK}_8R}$	0.512	0.496	0.527
8	$\mathcal{ND}^{\text{SPECK}_8R}$	$0.514 \pm 1.00 \times 10^{-3}$	$0.519 \pm 1.41 \times 10^{-3}$	$0.508 \pm 1.42 \times 10^{-3}$

F.3 BayesianKeySearch Algorithm in [15]

Algorithm 4: BAYESIANKEYSEARCH Algorithm [15]

```
/* The description of this BAYESIANKEYSEARCH Algorithm in [15] has
   a small typo and is inconsistent with that in the implementation
   codes [14], the description here corrects it according to [14]. */
Input: Ciphertext structure  $\mathcal{C} := \{C_0, \dots, C_{n_b-1}\}$ , a neural distinguisher  $\mathcal{ND}$ ,
   and its wrong key response profile  $\mu$  and  $\sigma$ , the number of candidates
   to be generated within each iteration  $n_{cand}$ , the number of iterations
    $n_{byit}$ 
Output: The list  $L$  of tuples of recommended keys and their scores
1  $S := \{k_0, k_1, \dots, k_{n_{cand}-1}\} \leftarrow$  choose  $n_{cand}$  values at random without
   replacement from the set of all subkey candidates.
2  $L \leftarrow \{\}$ 
3 for  $t = 1$  to  $n_{byit}$  do
4   for  $\forall k_i \in S$  do
5     for  $j = 0$  to  $n_b - 1$  do
6        $C'_{j,k_i} = F_{k_i}^{-1}(C_j)$ 
7        $v_{j,k_i} = \mathcal{ND}(C'_{j,k_i})$ 
8        $s_{j,k_i} = \log_2(v_{j,k_i}/(1 - v_{j,k_i}))$ 
9     end
10     $s_{k_i} = \sum_{j=0}^{n_b-1} s_{j,k_i}$ ; /* the combined score of  $k_i$  */
11     $L \leftarrow L||(k_i, s_{k_i})$ 
12     $m_{k_i} = \sum_{j=0}^{n_b-1} v_{j,k_i}/n_b$ 
13  end
14  for  $k \in \{0, 1, \dots, 2^{16} - 1\}$  do
15     $\lambda_k = \sum_{i=0}^{n_{cand}-1} (m_{k_i} - \mu_{k_i \oplus k})^2 / \sigma_{k_i \oplus k}^2$ 
16  end
17   $S \leftarrow \text{argsort}_k(\lambda)[0 : n_{cand} - 1]$ ; /* Pick  $n_{cand}$  keys with the  $n_{cand}$ 
   smallest score to form the new set of candidate keys  $S$  */
18 end
19 return  $L$ 
```

G Deeper Look into the \mathcal{ND} s on SIMON32/64

G.1 On \mathcal{ND} s for SIMON32/64 having learned to take back the final round

It would be interesting to test further the observation about the $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_nR}$ s for SIMON32/64 having learned to take back the final round. Following the idea of one reviewer, we try to find good combiners among $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{n-1}R}$, $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{n-1}R}$, and $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_nR}$ to distinguish n -round output, where $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{n-1}R}$ is the $(n-1)$ -round distinguisher that depends on the full DDT, $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{n-1}R}$ is the $(n-1)$ -round \mathcal{ND} that uses data of form $(x_{n-1}, x'_{n-1}, y_{n-1} \oplus y'_{n-1})$, and $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_nR}$ is the n -round \mathcal{ND} that uses data of format (x_n, x'_n, y_n, y'_n) .

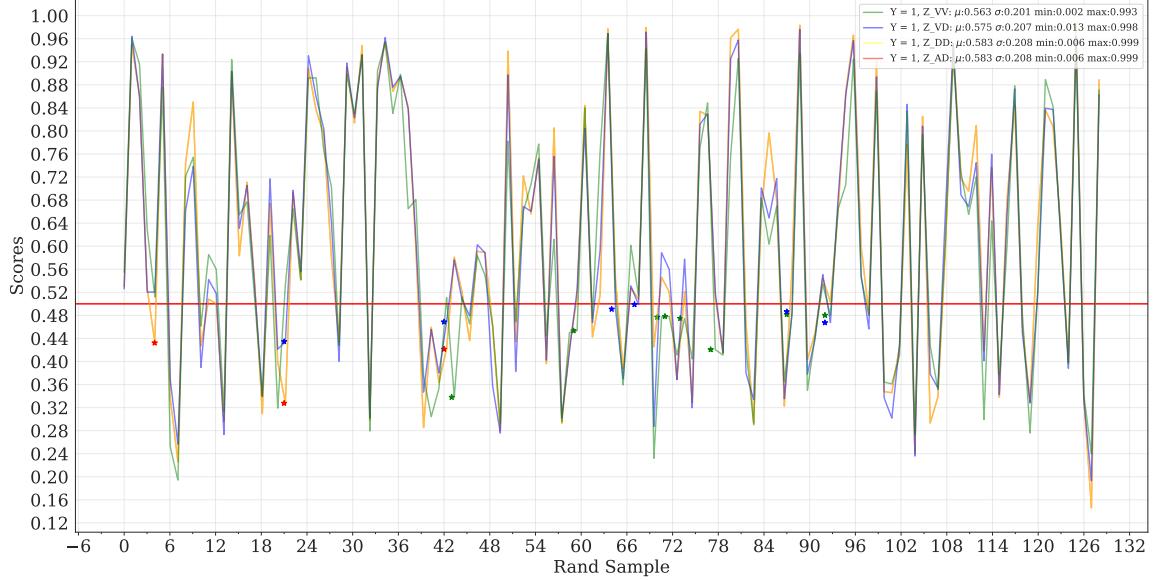
The searching and evaluation procedure for a good combiner is as follows:

1. Transform each value p in the DDT to a value in the range of $[0, 1]$ using the formula $p/(p + 2^{-32})$, such that the response of $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{n-1}R}$ is in the range of $[0, 1]$, and 2^{-32} corresponds to 0.5.
2. For a set of data X (about 2^{25} examples of n -round ciphertext pairs, half positive and half negative), obtain the scores from distinguishers $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{n-1}R}$, $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{n-1}R}$, and $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_nR}$.
3. Extract cases where all distinguishers agree, construct data set A where each entry is a tuple of scores given by different distinguishers on same examples of ciphertext pair, labeling with the label of the ciphertext pair. Use this data set A to fit a first linear regression model, denoted by LR_a .
4. Extract cases where there exists a distinguisher who disagree, construct data set D where each entry is a tuple of scores given by different distinguishers on same examples of ciphertext pair, labeling with the label of the ciphertext pair. Use this data set D to fit a second linear regression model, denoted by LR_b .
5. For a set of fresh data X (about 2^{23} examples of n -round ciphertext pairs, half positive and half negative) for testing, obtain the scores from distinguishers $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{n-1}R}$, $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{n-1}R}$, and $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_nR}$.
6. For cases where all distinguishers agree, use the fitted linear regression model LR_a to predict their score.
7. For cases where there exists a distinguisher who disagree, use the fitted linear regression model LR_d to predict their score.

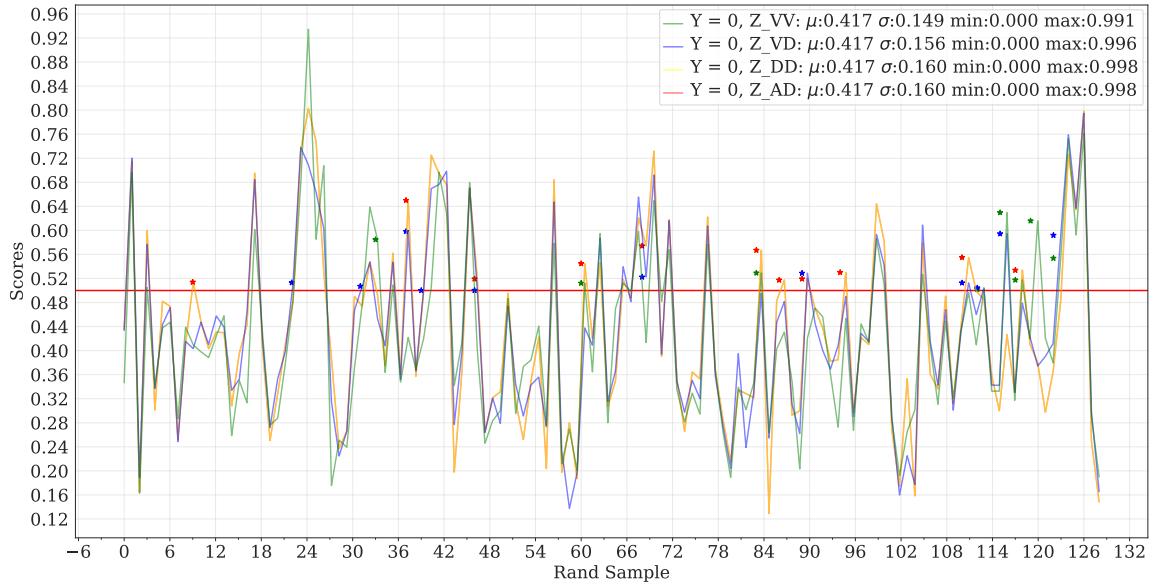
The best combiners obtained for $n = 10$ and $n = 9$ achieve almost the same performance as that of $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{n-1}R}$, which is the best among the three. They give almost 1 (e.g., 1.00353623 and 0.99973883) weights to the scores from $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{n-1}R}$ in both cases (agree and disagree).

Concrete results for $n = 9$ and 10 can be found in Table 11.

To look deeper into the relations of various distinguishers, we plot the responses of these distinguishers on a small set of examples. We found that on most examples, different distinguishers have consistent scores (see Figures 20

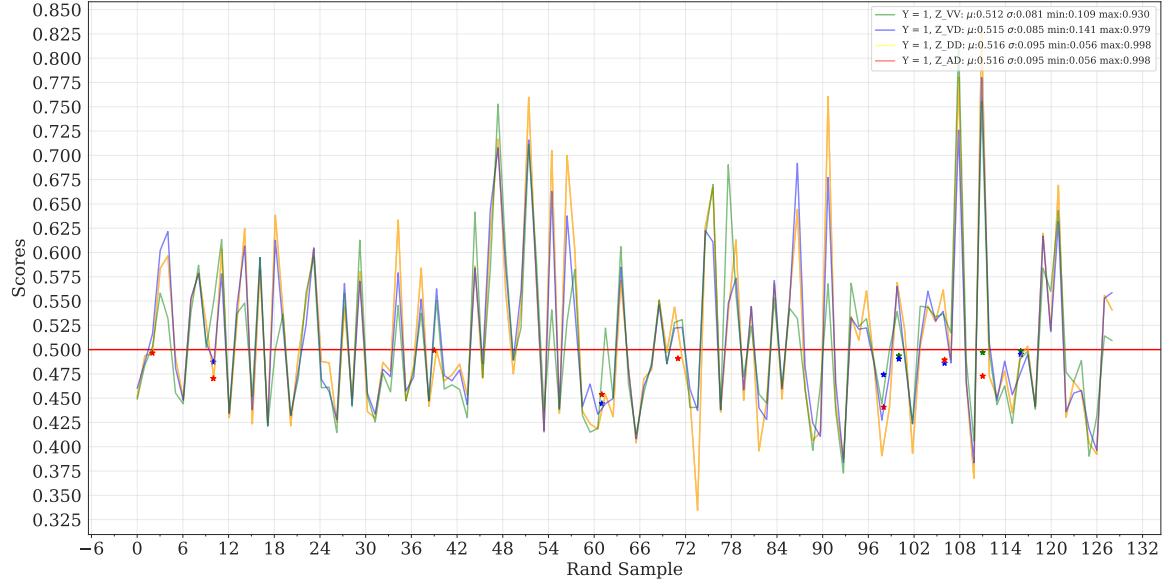


(a) Scores from distinguishers $\mathcal{DD}_{\text{DD}}^{\text{SIMON}_8R}$, $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_8R}$, and $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_9R}$ on 9-round positive examples

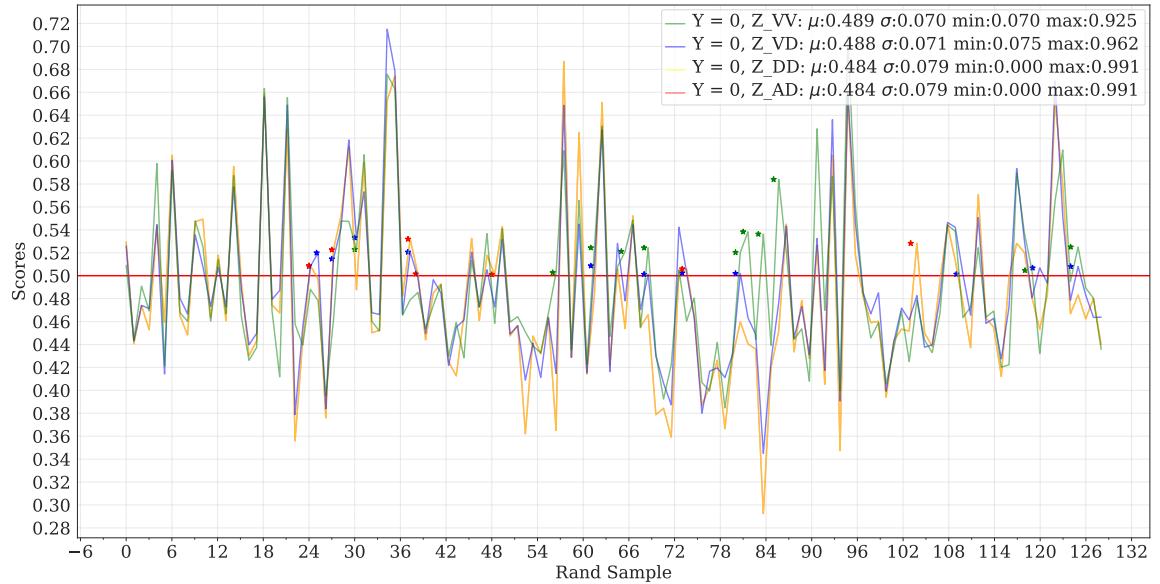


(b) Scores from distinguishers $\mathcal{DD}_{\text{DD}}^{\text{SIMON}_8R}$, $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_8R}$, and $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_9R}$ on 9-round negative examples

Fig. 20: Scores from distinguishers $\mathcal{DD}_{\text{AD}}^{\text{SIMON}_7R}$, $\mathcal{DD}_{\text{DD}}^{\text{SIMON}_8R}$, $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_8R}$, and $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_9R}$ on 9-round examples



(a) Scores from distinguishers $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}9R}$, $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}9R}$, and $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}10R}$ on 10-round positive examples



(b) Scores from distinguishers $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}9R}$, $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}9R}$, and $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}10R}$ on 10-round negative examples

Fig. 21: Scores from distinguishers $\mathcal{DD}_{\mathbf{AD}}^{\text{SIMON}8R}$, $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}9R}$, $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}9R}$, and $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}10R}$ on 9-round examples

Table 11: Accuracy of various distinguishers and combined ones on 9-round and 10-round SIMON32/64, and Key-averaging distinguisher on 5-round SPECK32/64 basing on 4-round DDT

On SIMON32/64					
#R	Name	ACC	TPR	TNR	MSE
9	$\mathcal{DD}_{\text{AD}}^{\text{SIMON}7R}$	0.663	0.579	0.748	0.211
9	$\mathcal{DD}_{\text{DD}}^{\text{SIMON}8R}$	0.663	0.579	0.748	0.211
9	$\mathcal{ND}_{\text{VD}}^{\text{SIMON}8R}$	0.659	0.559	0.759	0.210
9	$\mathcal{ND}_{\text{VV}}^{\text{SIMON}9R}$	0.652	0.534	0.770	0.214
9	$\mathcal{ND}_{\text{DD+VD+VV}}^{\text{SIMON}9R}$	0.663	0.578	0.747	0.209
10	$\mathcal{DD}_{\text{AD}}^{\text{SIMON}8R}$	0.568	0.469	0.668	0.243
10	$\mathcal{DD}_{\text{DD}}^{\text{SIMON}9R}$	0.568	0.469	0.668	0.243
10	$\mathcal{ND}_{\text{VD}}^{\text{SIMON}9R}$	0.566	0.474	0.657	0.243
10	$\mathcal{ND}_{\text{VV}}^{\text{SIMON}10R}$	0.561	0.476	0.646	0.244
10	$\mathcal{ND}_{\text{DD+VD+VV}}^{\text{SIMON}10R}$	0.568	0.468	0.669	0.242
On SPECK32/64					
#R	Name	ACC	TPR	TNR	MSE
5	$\mathcal{DD}_{\text{AD}}^{\text{SPECK}4R}$	0.936	0.917	0.956	0.048
5	$\mathcal{DD}_{\text{DD}}^{\text{SPECK}5R}$	0.911	0.877	0.944	0.066
6	$\mathcal{DD}_{\text{AD}}^{\text{SPECK}5R}$	0.795	0.731	0.858	0.142
6	$\mathcal{DD}_{\text{DD}}^{\text{SPECK}6R}$	0.758	0.679	0.837	0.162
7	$\mathcal{DD}_{\text{AD}}^{\text{SPECK}6R}$	0.624	0.543	0.705	0.227
7	$\mathcal{DD}_{\text{DD}}^{\text{SPECK}7R}$	0.591	0.543	0.639	0.236
8	$\mathcal{DD}_{\text{AD}}^{\text{SPECK}7R}$	0.519	0.492	0.547	0.249
8	$\mathcal{DD}_{\text{DD}}^{\text{SPECK}8R}$	0.512	0.497	0.527	0.250

Different distinguishers on the same rounds are tested with the same set of 2^{23} fresh examples.

and 21). At the same time, each distinguisher misses some cases when there exists another distinguisher be correct (outliers are marked in Figures 20 and 21), which indicates that there is no single super distinguisher that is better than all the others.

From the close performance between $\mathcal{DD}_{\text{DD}}^{\text{SIMON}_{n-1}R}$, $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{n-1}R}$ and $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_nR}$ for SIMON32/64 (see Table 11, Figures 22 and 23) and considering possible small deviation of the training process, we conclude that the performance of $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_{n-1}R}$ is the upper bound of the performance of $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_nR}$, and conjectured that accepting data of form (x_n, x'_n, y_n, y'_n) , $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_nR}$'s have learned to detect the features of data $(x_{n-1}, x'_{n-1}, y_{n-1} \oplus y'_{n-1})$ that can be obtained by manually taking (x_n, x'_n, y_n, y'_n) back by one round without knowing the last

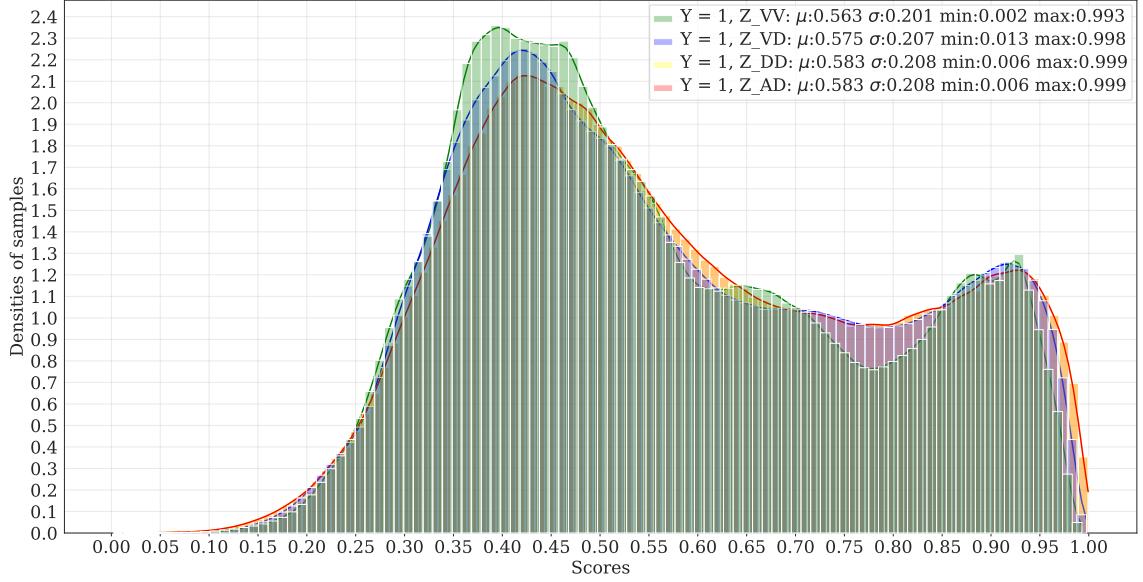
subkey. That is, in the ideal training case, the performance of an $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_n R}$ should be the same as that of an $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{n-1} R}$.

G.2 On \mathcal{ND} s and \mathcal{DD} s of SIMON32/64 failing to detect potential signals

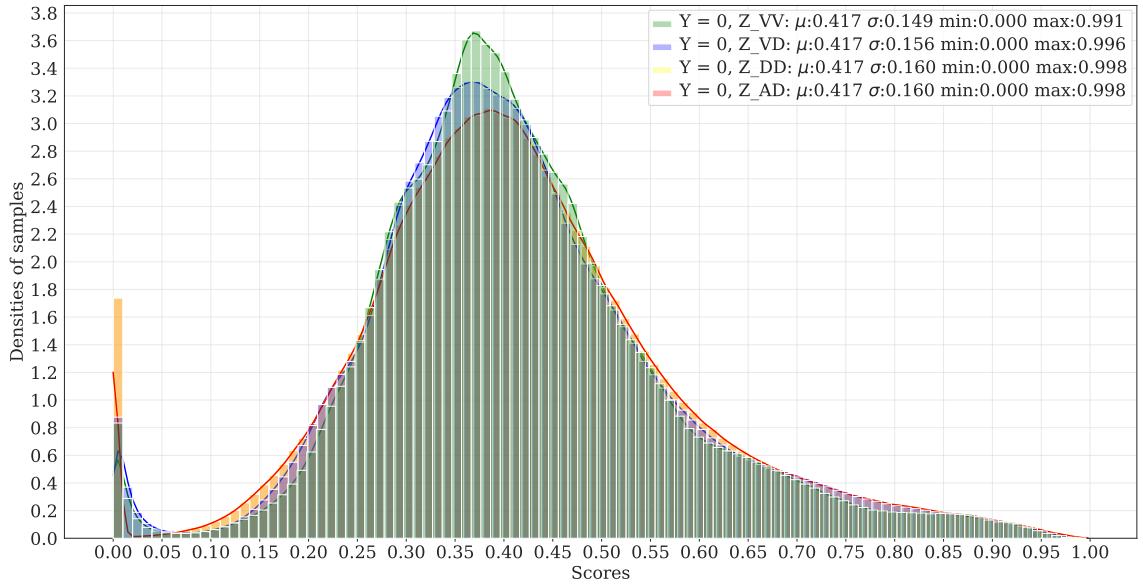
It would be interesting to learn whether there exists a signal beyond that found by a $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{n-1} R}$ and if it exists, whether a $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_n R}$ and a $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{n-1} R}$ for SIMON32/64 are failing to detect such a signal. As suggested by a reviewer, this could be done by running a key-averaging distinguisher (refer to Algorithm 1 in [15]) against n rounds that relies on the DDT at $n-2$ rounds. Accordingly, we built two key-averaging distinguishers against n rounds that rely on $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{n-2} R}$, denoted by $\mathcal{DD}_{\mathbf{AD}}^{\text{SIMON}_{n-2} R}$, where $n = 9$ and 10 .

Interestingly, these $\mathcal{DD}_{\mathbf{AD}}^{\text{SIMON}_{n-2} R}$ s have exactly the same performance (ACC, TPR, TNR, and MSE) with $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{n-1} R}$ s. This result is quite different from that of SPECK32/64 (see Table 11 for the experimental results on SPECK32/64, where $\mathcal{DD}_{\mathbf{AD}}^{\text{SPECK}_{n-1} R}$ is an n -round key-averaging distinguisher relying on $(n-1)$ -round full DDT). This indicates that, for SIMON32/64, the computation of the n -round DDT using $(n-1)$ -round full DDT and the algorithm in [24] is accurate in the sense of averaging over all subkeys. Therefore, there are no additional signals beyond that found by a $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{n-1} R}$ for a $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{n-1} R}$ or a $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_n R}$ to detect.

Please find the source code of the experiments and the raw data of the results via https://github.com/differential-neural-cryptanalysis/speck32_simon32.

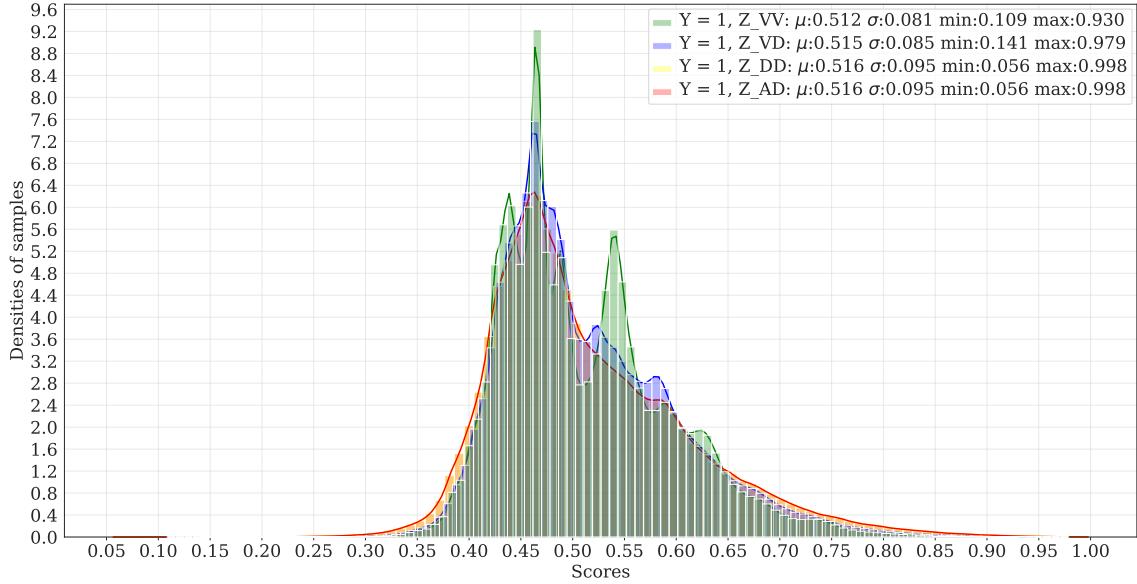


(a) Distribution of scores from distinguishers $\mathcal{DD}_{\text{AD}}^{\text{SIMON}_7R}$, $\mathcal{DD}_{\text{DD}}^{\text{SIMON}_8R}$, $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_8R}$, and $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_9R}$ on 9-round positive examples

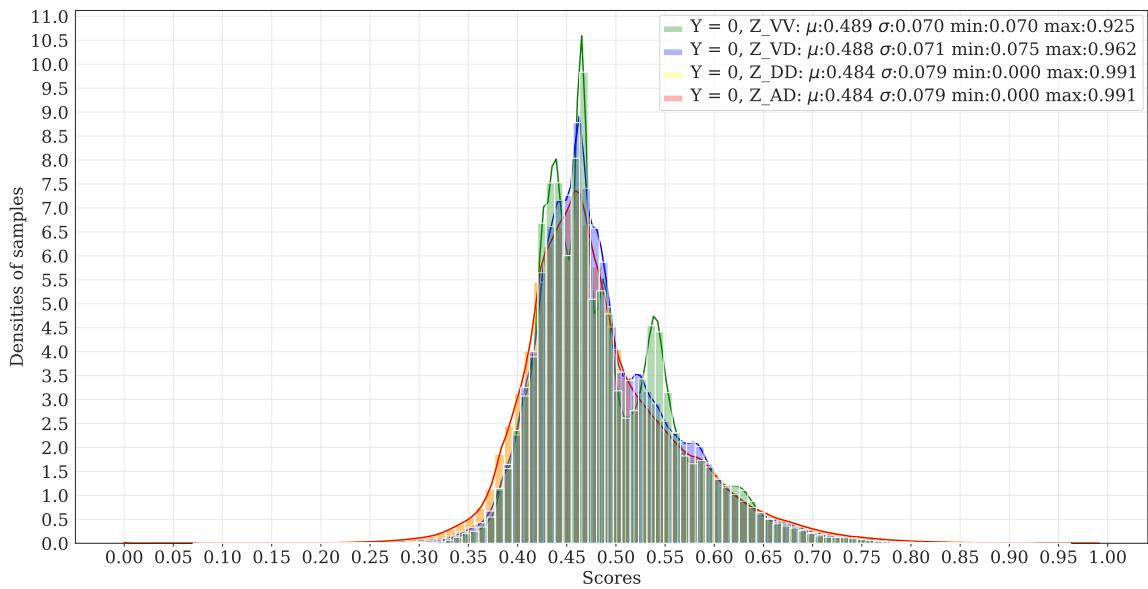


(b) Distribution of scores from distinguishers $\mathcal{DD}_{\text{AD}}^{\text{SIMON}_7R}$, $\mathcal{DD}_{\text{DD}}^{\text{SIMON}_8R}$, $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_8R}$, and $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_9R}$ on 9-round negative examples

Fig. 22: Distribution of scores from distinguishers $\mathcal{DD}_{\text{AD}}^{\text{SIMON}_7R}$, $\mathcal{DD}_{\text{DD}}^{\text{SIMON}_8R}$, $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_8R}$, and $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_9R}$ on 9-round examples



(a) Distribution of scores from distinguishers $\mathcal{DD}_{\text{AD}}^{\text{SIMON}_8R}$, $\mathcal{DD}_{\text{DD}}^{\text{SIMON}_9R}$, $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_9R}$, and $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{10R}}$ on 10-round positive examples



(b) Distribution of scores from distinguishers $\mathcal{DD}_{\text{AD}}^{\text{SIMON}_8R}$, $\mathcal{DD}_{\text{DD}}^{\text{SIMON}_9R}$, $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_9R}$, and $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{10R}}$ on 10-round negative examples

Fig. 23: Distribution of scores from distinguishers $\mathcal{DD}_{\text{AD}}^{\text{SIMON}_8R}$, $\mathcal{DD}_{\text{DD}}^{\text{SIMON}_9R}$, $\mathcal{ND}_{\text{VD}}^{\text{SIMON}_9R}$, and $\mathcal{ND}_{\text{VV}}^{\text{SIMON}_{10R}}$ on 10-round examples