Conditional Differential-Neural Cryptanalysis

Zhenzhen Bao[1], Jian Guo[1], Meicheng Liu[2], Li Ma[2], and Yi Tu[1]

[1] Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore.
{zzbao,guojian}@ntu.edu.sg, tuyi0002@e.ntu.edu.sg

[2] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, China.
{meicheng.liu,skloismary}@gmail.com

Abstract. Although it has been a long-standing question that whether computers can learn to perform cryptanalytic tasks, positive answers made by breakthrough machine-learning-based cryptanalysis are still rare. In CRYPTO 2019, a remarkable work made by Gohr shed light on a positive answer. It shows that well-trained neural networks can perform cryptanalytic distinguishing tasks at a superior level to traditional differential-based distinguishers. Additionally, a non-traditional key-recovery procedure was devised, integrating with the Upper Confidence Bounds and Bayesian optimization. Combining the neural distinguishers with a classical differential, integrating the advanced key-recovery procedure, an 11-round key-recovery attack on Speck32/64, a small-sized modern cipher designed by researchers from NSA, was achieved, which has a competitive performance compared with the state-of-the-art result. However, it turns out to be still difficult for the community to achieve a comparable performance increase on longer reduced-versions of the same cipher. This difficulty calls into a question: to what extent is the advantage of machine-learning approaches over traditional ones, and whether the advantage generally exists on modern ciphers?

To answer these questions, we devised the first practical 13-round and improved 12-round neural-distinguisher-based key-recovery attacks on Speck32/64 and 16-round key-recovery attacks on Simon32/64. The results confirm the advantages of using machine-learning approaches in cryptanalysis. However, the main reason lies in the enhancement made on the classical components. The crucial technical element for the improved attacks is the concept of conditional (simultaneous) neutral bits/bit-sets, which is derived from the concept of neutral bit with a long history in cryptanalysis. This fact indicates an outcome: a strengthened combination between the classical cryptanalysis and machine learning approaches is one way for machine-learning-based cryptanalysis to maximize its advantage.

Apart from best attacks, we exhibit substantial details of the key-recovery phase that is missing a theoretical model to analyze its complexity and success probability. Some observations on important statistics could serve as a rule of thumb on tuning parameters and making trade-offs.

To answer whether the advantage of machine learning approaches shown in the cryptanalysis of Speck32/64 can also be obtained on other prim-

itives, we produce various neural distinguishers and traditional DDT-based distinguisher on SIMON32/64. The answer is slightly negative. The same approaches for SPECK32/64 indeed apply to SIMON32/64. However, the advantage over the pure differential-based approach seems to be limited.

Keywords: Neural Distinguisher, Key Recovery Attack, Differential Cryptanalysis, SIMON, SPECK, Generalized Neutral Bits, Bayesian Search

1 Introduction

Cryptography and machine learning share many common 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" [23]. Artificial intelligence (AI) and machine learning (ML) have made rapid progress in application domains ranging from machine translation, visual recognition, autonomous vehicles to playing various board games at superhuman levels [24]. In addition, ML has also been utilized to construct new types of cryptographic schemes [1] or crack ancient ciphers [15].

However, it is still an unforeseen future for machine learning models learning from scratch and then completely breaking modern ciphers. Yet, one can still look forward to that machine learning approaches and deep learning models be a strong positive addition to the existing cryptanalysis toolkit, which has already been true in side-channel cryptanalysis [22].

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

- Is it possible for ML models discovering new features with/without prior human cryptanalysis?
- Is it possible for ML approaches to provide a more accurate and efficient measurement of known features?
- Is it possible for compositions of various ML approaches and cryptanalysis techniques to perform cryptanalysis tasks at a superior level to orthodox cryptanalysis, then be dissected and interpreted, and in turn help to develop innovative and general cryptanalytic techniques?
- Is it possible for ML approaches to be used in developing efficient and intelligent tools to explore a vast space of attack vectors?

These questions, together with plenty of exciting achievements of ML in various fields, motivate the community to an unorthodox explore, although breakthrough cryptanalysis using machine learning is still rare.

In CRYPTO 2019, a remarkable work by Gohr [14] 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, and showed that for SPECK, deep neural-network distinguisher could exploit features that strong classical distinguishers fail to capture. In [14], neural networks were

trained with principles of differential cryptanalysis in mind. They show a remarkable capability in distinguishing attacks. More importantly, when combining them with classical differentials and using a highly selective key search policy, the formed differential-neural cryptanalysis can be a powerful key recovery attack. Specifically, using the obtained neural distinguishers as the main engines, prepending them with a classical differential, applying basic reinforcement learning mechanisms, *i.e.*, the Upper Confidence Bounds (UCB) and Bayesian optimization, results in an 11-round key-recovery attack on SPECK32/64 with an unparalleled speed. However, attacking more rounds, either the classical component, i.e., the prepended differential, be extended, or the neural distinguisher be. Both are facing obstacles that have not been overcome since [14].

In EUROCRYPT 2021, Ghor's neural distinguishers got 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 [14], has not been fully interpreted and theorized, thus still missing necessary guidance on how to tuning various parameters and sound theoretical models on the relationship between various parameters with the data/time complexity and the success probability.

Note that one of the main difficulties in evaluating and characterizing the scope of applicability of machine learning algorithms is the lack of a formally specified theoretical model. Strong theoretical models for machine-learning-based cryptanalysis are vital for generalizing the technique used in practical attacks on small-size ciphers to theoretical attacks on large ones. However, in parallel or even before our community manages to achieve sound theoretical models (which is inevitable very difficult), devising a sufficient number of successful attacks as positive examples in this new setting is essential. Without providing the best successful 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 substantial experimental data to support the first steps towards a realistic theoretical model for machine-learning-based cryptanalysis.

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

- Practical attacks and illustrations
  - The first practical neural-distinguisher-based 13-round key-recovery attack and an improved 12-round key-recovery attack on SPECK32/64 were devised, which have a considerable advantage in time complexity over attacks devised using orthodox cryptanalysis. In addition, the first practical neural-distinguisher-based 16-round key-recovery attack on SIMON32/64 was devised, which has a considerable advantage in data complexity over the attack devised using orthodox cryptanalysis. These results are summarized in Table 1.
  - Substantial illustrations on previously hidden details of the key-recovery phase are displayed. Important observations (especially Observation 1)

together with these exhibitions of details provide some rule of thumb on tuning important parameters and making better trade-offs.

– Enhanced cryptanalytic technique
  The improved attacks were achieved by enhancing the classical components in the differential-neural attack scheme in [14]. Specifically, we deeply explored more generalized neutral bits of differentials, which we call conditional (simultaneous) neutral bit/bit-sets. Note that the concept of *neutral bit* has a long history and heroic role in various cryptanalysis on various primitives [4, 6, 8, 14]. We name the attacks equipped with this generalized concept of neutral bit the *conditional differential-neural cryptanalysis*. This might be of independent interest for cryptanalysis in the classical setting.
  - Firstly, in addition to the single neutral bits used in [14], a more sophisticated combination of bits, which allows generating additional conforming pairs by flipping the set simultaneously, was found and exploited. We call them "simultaneous neutral bit-sets" (SNBS's).
  - Secondly, in addition to neutral bits that generate the additional conforming pairs with probability one (named as complete neutral bits), those with lower probabilities are also of interest (named as probabilistic neutral bits). We find that some probabilistic neutral bits can be turned to complete by imposing conditions on the chosen inputs. We call them "conditional neutral bits" (CNB's), or "conditional simultaneous neutral bit-sets" (CSNBS's).
  - We find *all* SNBS's (flipping simultaneously up to 4 bits) in a brute-force search or an algebraic method basing on Gröbner basis. For finding CNB's, the manual analysis supported by experiments was applied.

– New observations
  - We note the output difference of differential path matters to neural distinguisher (ND), but not the input difference. Hence, more than one differential can be prepended to ND, as long as they share the same output difference. Surprisingly, some neutral bits can be shared by multiple such differentials. Using such differentials might enable data reuse, thus slightly reduce 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 [9,12,25]. Thus, the attacks only work for a subspace of the keys, *i.e.*, weak keys up to half of the keyspace. Interestingly, within these weak keys and knowledge of these constraints, the attacks can generally be improved in time and data complexities.

– Various neural distinguishers and DDT-based distinguishers for Simon32/64
  - Besides the Residual Network (ResNet) [17] considered by Gohr in [14], other neural networks that have shown advantages on ResNet in specific tasks, including Dense Network (DenseNet) [19] and the Squeeze-and-Excitation Network (SENet) [18], are investigated. Additionally, various training schemes, including direct training, key-averaging, and staged training, were attempted. This effort results in various neural distinguishers covering up to 11-round Simon32/64.

- Full distribution of differences for SIMON32/64 induced by the input difference `0x0000/0040` up to 11 rounds are computed. These pure differential-based distinguishers (DD) provide solid baselines for neural-network-based distinguishers (ND). The results show that r-round ND achieves similar but weaker classification accuracy than $(r-1)$-round DD (see Table 5). Besides, in a key ranking task on 11-round SIMON32/64, ND can perform well but still cannot do better than its counterpart DD (see Table 6). In this sense, r-round ND 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. Results by these efforts support Benamira *et al.*'s interpretation, but meanwhile indicate a difference about the advantage of neural-distinguishers over classical differential distinguishers between SPECK and SIMON.

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

Target	#R	Time (GPU h)	Time (CPU h)	Time (#Enc)	Data	Succ. Rate	Weak keys	Dist.	Conf.	Ref.
SPECK32/64	11	-	-	2^{46}	2^{14}	-	2^{64}	DD	1+6+4	[12]
		-	0.139	2^{38}⋆	$2^{13.6}$	0.52	2^{64}	ND	1+2+7+1	[14]
	12	-	-	2^{51}	2^{19}	-	2^{64}	DD	1+7+4	[12]
		<1	12	$2^{43.40}$⋆	$2^{22.97}$	0.40	2^{64}	ND	1+2+8+1	[14]
		-	33.7	$2^{44.89}$⋆	$2^{22.00}$	0.86	2^{64}	ND	1+2+8+1	Sect. 4.3 Fig. 6
		-	11.6	$2^{43.35}$⋆	$2^{18.58}$	0.81	2^{63}	ND	1+3+7+1	Sect. 4.3 Fig. 7
	13	-	-	2^{57}	2^{25}	-	2^{64}	DD	1+8+4	[12]
		1043.2	-	$2^{49.84}$⋆$+r$	2^{30}	0.75	2^{63}	ND	1+3+8+1	Sect. 4.2 Fig. 4
		652.8	-	$2^{49.16}$⋆$+r$	2^{29}	0.61	2^{63}	ND	1+3+8+1	Sect. 4.2 Fig. 5
	14	-	-	2^{62}‡	2^{30}‡	-	2^{63}‡	DD	1+9+4	[12]
		-	-	$2^{61.47}$‡	$2^{29.47}$‡	-	2^{63}‡	DD	1+9+4	[25]
SIMON32/64	16	-	-	$2^{26.48}$	$2^{29.48}$	0.62	2^{64}	DD	2+12+2	[3]
		4	-	$2^{41.81}$⋆$+r$	2^{21}	0.49	2^{64}	ND	1+3+11+1	Sect. D.4
	18	-	-	$2^{46.00}$	$2^{31.2}$	0.63	2^{64}	DD	1+13+4	[2]
	19	-	-	$2^{34.00}$	$2^{31.5}$	-	2^{64}	DD	2+13+4	[9]
	21	-	-	$2^{55.25}$	$2^{31.0}$	-	2^{64}	DD	4+13+4	[27]

- 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 is the CPU time and gpu is the GPU time running an attack. 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 seconds on CPU and 1265 seconds on GPU).
‡ Revised complexity due to newly found constraints on subkeys (Sect. C)

ORGANIZATION. The rest of the paper is organized as follows. Section 2 gives the preliminary on machine-learning-based differential cryptanalysis and introduces the design of SIMON and SPECK. Section 3 introduces concepts of generalized neutral bits and some new notice on differential trails of SPECK32/64. The framework of the conditional-differential neural cryptanalysis and its applications to

SPECK32/64 and SIMON32/64 are presented in Section 4 and D. Section 5 exhibits details of important statistics during the key-recovery phase. In addition, rules of thumb are provided for tuning various parameters for the attacks. Section 6 presents various of ND's and 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, $\cdot$ or & the bit-wise AND, $x^{\lll s}$ or $x \lll s$ the bit-wise left rotation by s positions, $x^{\ggg s}$ or $x \ggg s$ 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 in the lightweight block cipher families SPECK and SIMON [5] designed by researchers from the National Security Agency (NSA) of the USA. Both SPECK32/64 and SIMON32/64 are of Feistel constructions, has 32-bit block and 64-bit key. 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 SIMON32/64

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

Algorithm 2: Encryption of SPECK32/64

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

2.2 Differential-based Neural Distinguishers

The work in [14] shows that neural network could be trained to capture the non-randomness of the distribution of values of output pairs when the input pairs to round-reduced SPECK32/64 are of specific difference, and thus play the role of distinguisher in cryptanalysis. This differential-based neural distinguisher is the first known machine learning model that successfully performed cryptanalysis task on modern ciphers (beyond the applications on side-channel attacks).

In the following, the way of training the differential-based neural distinguisher introduced in [14] is briefly recalled.

The Training Data and Input Representation. For a target cipher, the neural network is to be 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.*, $(\texttt{0x0040}, \texttt{0000})$, 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 fed into the neural network with an input layer consisting of 64 units. Among the set of training data and verification data, half are positive examples labelled by 1, and the other half are negative examples labelled by 0.

Training Schemes. The neural network structure used in [14] is a deep residual network. There are three training schemes proposed in [14]. 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 KEYAVERAGING algorithm used with an $(r-1)$-round distinguisher. Using the second scheme, the best neural distinguisher on 7-round SPECK32/64 was achieved in [14]. 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 neural distinguisher 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.*, an automatic exploitation versus exploration trade-off based on upper confidence bounds) was proposed in [14].

The improved key-recovery attack employs an r-round main and an $(r-1)$-round helper neural distinguisher 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 neural distinguishers (refer to Fig. 1 for an illustration of the components of the key-recovery attack.) About $c \cdot 2^p$ (denoted by n_{cts}) data pairs with difference $\Delta_{I'}$ 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 resulted n_{cts} structures of data pairs are decrypted by one round with 0 as the subkey 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 neural distinguisher (and latter of the second to last subkey by the $(r-1)$-round helper neural distinguisher) 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 bitwise 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 with minimizing the weighted Euclidean distance as the criteria in an BayesianKeySearch Algorithm 3. 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 the visited times of them.

An important detail in the BayesianKeySearch is that the responses $v_{i,k}$ from the neural distinguisher 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 Algorithm 3). 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 it 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 neural distinguisher has a low accuracy.

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

3 Deep Exploring of Neutral Bits

3.1 Motivation of Neutral Bits

In general, the more rounds the neural distinguisher covers, the lower the accuracy is. Theoretically, a neural distinguisher with an accuracy that is higher

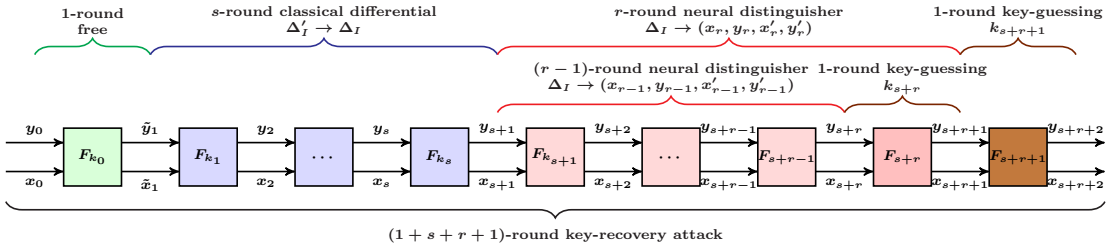


Fig. 1: Components of the key-recovery attacks

than 0.5 should have some distinguishing advantage over a random distinguisher. However, when the accuracy being marginally higher than 0.5, it is hard to be used in practical key recovery attack [14]. Thus, Gohr in [14] used the combined response (as in Formula 1) of the neural distinguisher over large number of samples of same distribution as a distinguisher (named as combined-response-distinguisher). By doing so, the signal from the neural distinguisher is amplified and the distinguishability is increased. For a combined-response-distinguisher built on top of a weak neural distinguisher to reach its most potential with respect to distinguishability, the number of samples of the same distribution should be sufficiently large.

For the hybrid differential distinguisher used in the key-recovery attack in [14], it is not straightforward to aggregate enough number of samples of same distribution fed to the neural distinguisher due to the prepended classical differential. To overcome this problem, Gohr in [14] used the neutral bits of the classical differential, which is a notion introduced by Biham and Chen for collision attacking SHA-0 [8] and frequently used in previous attacks of different types on various primitives [4,6]. That is, changing the values at the neutral bits of an input pair does not change the conformability for the differential. The more the neutral bits of the prepended differential, the larger the number of samples of same distribution could be generated and fed into the neural distinguisher. However, in general, the longer the classical differential, the lesser the number of neutral bits.

Finding enough neutral bits for prepending a long differential over a long but weak neural distinguisher becomes a difficult problem for devising a key-recovery to cover more rounds.

Thus, the first part of this work focuses on finding new types of neutral bits.

3.2 Neutral Bits and Generalized Neutral Bits

Notations. Let $\Delta_{in} \to \Delta_{out}$ be a differential with input difference Δ_{in} and output difference Δ_{out} of an r-round encryption F^r. Let (P, P') be the input pair and $(C, C' \mid C = F^r(P), C' = F^r(P'))$ be the output pair, where $P \oplus P' = \Delta_{in}$. If $C \oplus C' = \Delta_{out}$, (P, P') is said to be *conforming* the differential $\Delta_{in} \to \Delta_{out}$.

The primary notion of neutral bits can be interpreted as follows.

Definition 1 (Neutral bits of a differential, NB's [8]). *Let $e_0, e_1, \ldots, e_{n-1}$ 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 $\Delta_{in} \to \Delta_{out}$*, if for any conforming pair* (P, P')*,* $(P \oplus e_i, P' \oplus e_i)$ *is also a conforming pair.*

Let $\{i_1, i_2, \ldots, i_n\}$ be the set of neutral bits of a differential $\Delta_{in} \to \Delta_{out}$. Denote the subspace of $\mathbb{F}_2^n$ with basis $\{e_{i_1}, e_{i_2}, \ldots, e_{i_n}\}$ 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 data structure with the same conformability for the differential.

For a differential $\Delta_{in} \to \Delta_{out}$ of F^r, in the view of 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 [14], because of the lack of neutral bits for the 2-round prepended differential of Speck32/64, probabilistic neutral bits (PNB's 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, PNB's [4]). *Let* i *be an index of a bit. The* i*-th bit is a* p*-probabilistic neutral bit of the differential* $\Delta_{in} \to \Delta_{out}$*, if for any conforming pair* (P, P')*,* $(P \oplus e_i, P' \oplus e_i)$ *conform the differential at the same time with a probability* p*.*

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, the neutral bits are said to be *complete neutral bits* when $p = 1$.

In this work, two types of generalized neutral bits are considered beyond the (probabilistic) neutral bits considered in [14]. The first type, named as simultaneous-neutral bit-set (SNBS's for short), has already been introduced together with the notion of neutral bit in [8]. 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, SNBS's [8]). *Let* $I_s = \{i_1, i_2, \ldots, i_s\}$ *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 for the differential* $\Delta_{in} \to \Delta_{out}$*, 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.*

The second type, which is a natural generalization, is named in this work, as conditional (simultaneous-) neutral bit(-set)s (CSNBS's 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, CSNBS's). *let* $I_s = \{i_1, i_2, \ldots, i_s\}$ *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* $\mathcal{P}_\mathcal{C}$ *be the set of inputs that fulfill the constraints* $\mathcal{C}$. *The bit-set* I_s *is a conditional simultaneous-neutral bit-set for the differential* $\Delta_{in} \to \Delta_{out}$, *if for any conforming pair* $(P, P' \mid P \in \mathcal{P}_\mathcal{C})$, $(P \oplus f_{I_s}, P' \oplus f_{I_s})$ *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 more involved system of linear or non-linear equations, and correspondingly named as *linear-conditional (simultaneous-) neutral bit(-set)s* (LCSNBS's for short) and *nonlinear-conditional (simultaneous-) neutral bit(-set)s* (NCSNBS's for short).

Specifically, in this work for SIMON32/64 and SPECK32/64, conditional neutral bits are slightly different in the following ways:

- for SPECK32/64, a set of bits is neutral only when the value of some specific bits are fixed to a particular value. Thus, one chooses particular data instead of random one to form ciphertext-structure, but always uses the same set of neutral bit-sets (refer to Sect. 4.1).
- for SIMON32/64, depending on the value of specific bits, one can always obtain a neutral bit-sets by grouping different bits. Thus, one randomly generates a data pair, then selects different neutral bit-sets depending on the values of specific bits of the random pair to generate a ciphertext-structure (refer to Sect. D.5).

An observation on neutral bits of reduced-round differential of SPECK32/64 is that, the lower the Hamming weight of a differential, the more the number of neutral bits. Thus, for some of the sequel key-recovery attacks, differentials with sub-optimal probability but with low Hamming weight are of interests and eventually used.

Remark 1. The neutrality of CSNBS's depends on values of some particular bits. The selected data is at intermediate round in our attacks in this work, although the difference does not depend on the round-key, the values do. Thus, using CSNBS's, the attack requires to guess some key bits of the first round.

Remark 2. In the sequel attacks on SIMON32/64, with involved analysis, conditions on the neutral bits are explicit, and thus, under the known conditions, all used (generalized) neutral bits are complete. For SPECK32/64, it is difficult to fully capture all explicit conditions for some bits to be neutral because of the complicated modular addition. Thus, some used (generalized) neutral bits are not completely neutral but neutral with high probability. That means, besides those explicit conditions observed, there are still some hidden conditions that can be fulfilled with high probability.

3.3 Exploiting Multiple Differentials Sharing Same Neutral Bits

For the prepended classical differential, the goal is to propagate more rounds with as less plaintext requirement as possible while leaving enough positive samples to the neural distinguisher.

From the connecting difference between the classical differential and the neural distinguisher propagating upward, there might be multiple similar differentials with equally good probability. The observation is that, 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, let $D_1 = \Delta_{in_1} \rightarrow \Delta_{out}$ and $D_2 = \Delta_{in_2} \rightarrow \Delta_{out}$ be two differentials with input differences satisfying $\Delta_{in_1} \oplus \Delta_{in_2} = \Delta_{nb_i}$ and with the same output difference. Suppose nb_i is a neutral bit for both D_1 and D_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 D_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 D_2. Thus, by re-pairing the corresponding ciphertext pairs, the number of ciphertext structures is doubled. Such a pair of differentials are said to be matched differentials.

This 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 other neutral bits to be used in key-recovery attacks.

An example can be found in Sect. 4.1. One useful differential might match with many useful differentials in this sense. The more matched differentials found, the lower the final data complexity will be.

Remark 3. There is an implicit relation between neutral bits of a differential and high-order differential. A simultaneous-neutral bit-set 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 [16]. 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 sections shows that the neural distinguishers have not reached their full potential in the key-recovery attacks in [14]. They could be harnessed to cooperate with classical cryptanalytic tools and perform key-recovery attacks that are more competitive to the attacks devised purely by classical cryptanalysis techniques.

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 [14].

The first 13-round attack and an improved 12-round attack that use neural distinguishers on SPECK32/64 were obtained.

The improved attacks follow the same framework of the improved key-recovery attacks on SPECK32/64 in [14]. An r-round main and an $(r-1)$-round helper neural distinguishers are employed and an s-round classical differential 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, Bayesian key search employing wrong key response profile is to be used.

The prepended classical differentials to be used in the improved attacks include the same 2-round differential used in the attack in [14] and four new 3-round differentials. The preliminary is to find enough NB's of these differentials to obtain enough samples of same distribution, so that to use the combined response from the neural distinguishers. In the following, the simultaneous neutral bit-sets and CNB's introduced in Sect. 3 are to be found.

4.1 Finding CSNBS's for SPECK32/64

For finding NB's 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 SNBS's for 2-round Differential. For the prepended 2-round differential on top of the neural distinguishers, one can experimentally obtain 3 complete NB's and 2 SNBS's (simultaneously complementing up to 4 bits) using exhaustive search. Besides, bits and bit-sets that are (simultaneous-)neutral with high probabilities are also detected. Concretely, for the 2-round differential $(\texttt{0x0211}, \texttt{0x0a04}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$, bits and bit-sets that are (probabilistic) (simultaneous-)neutral are summarized in Table 2.

Table 2: (Probabilistic) SNBS's for 2-round differential $(\texttt{0x0211}, \texttt{0x0a04}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ of SPECK32/64

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

Finding SNBS's for 3-round Differential. The 2-round differential $(\texttt{0x0211}, \texttt{0x0a04}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ can be extended to two optimal (prob. $\approx 2^{-11}$) 3-round differentials, *i.e.*,

$$(\texttt{0x0a20}, \texttt{0x4205}) \rightarrow (\texttt{0x0040}, \texttt{0x0000}), (\texttt{0x0a60}, \texttt{0x4205}) \rightarrow (\texttt{0x0040}, \texttt{0x0000}).$$

However, the NB's/SNBS's 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. C for more details), *i.e.*,

$$(\texttt{0x8020}, \texttt{0x4101}) \to (\texttt{0x0040}, \texttt{0x0000}), (\texttt{0x8060}, \texttt{0x4101}) \to (\texttt{0x0040}, \texttt{0x0000}),$$
$$(\texttt{0x8021}, \texttt{0x4101}) \to (\texttt{0x0040}, \texttt{0x0000}), (\texttt{0x8061}, \texttt{0x4101}) \to (\texttt{0x0040}, \texttt{0x0000}).$$

For these sub-optimal 3-round differentials, the hamming weights of the input differences are low, and they have more SNBS's. Still, the numbers of SNBS's are not enough for appending a weak neural network distinguisher. Thus, conditional ones were searched. The concrete approach for finding CNB's/CNBS's is empirical.

At a high-level, the empirical approach is as follows. First, the sufficient conditions for a bit or a set of bits to be neutral are observed. Next, the necessity of the sufficient conditions is tested. Concretely, let $(\tilde{x}, \tilde{y})$ be the chosen data for the 3-round differential. Because the 3-round differential will be neutrally extended one round to the backward in the key-recovery attack, in the real encryption, $(x, y) = (\tilde{x} \oplus k_0, \tilde{y} \oplus k_0)$ is the real input to the 3-round differential (refer to Fig. 3). The considered sufficient conditions are on the values of each bit of the following four variables, *i.e.*, x, y, $(x \ggg 7) \oplus y$, $(x \ggg 7) \cdot y$. All bits of these variables are examined to see if any of them keeps as a constant 0 or 1 among all correct pairs in the structure generated by each candidate CNBS. Concerning values of x and y is for examining the conditions on the values of the inputs; concerning the values of the later two is for examining the conditions on the values that will be involved in the modular addition. We observed that for some bits/bits-sets that are neutral with relatively high probabilities, some bits p_i's of $(x \ggg 7) \oplus y$ for the correct pairs are always b ($b \in \{0, 1\}$), from which we obtained the sufficient conditions for the bits/bits-sets to be neutral. We then fixed the corresponding bits p_i's to be b, and examined the probabilities for the bits/bits-sets to be neutral. Exploited experimental results are summarized in Table 4. Besides, we observed that for each of the four sub-optimal differentials, there are three sufficient (linear) conditions for a pair $((x, y), (x', y'))$ to conform the 3-round differentials, as listed in Table 3.

Table 3: Sufficient conditions to conform the 3-round differentials

$(\texttt{0x8020}, \texttt{0x4101})$ $\downarrow$ $(\texttt{0x0040}, \texttt{0x0000})$	$(\texttt{0x8060}, \texttt{0x4101})$ $\downarrow$ $(\texttt{0x0040}, \texttt{0x0000})$	$(\texttt{0x8021}, \texttt{0x4101})$ $\downarrow$ $(\texttt{0x0040}, \texttt{0x0000})$	$(\texttt{0x8061}, \texttt{0x4101})$ $\downarrow$ $(\texttt{0x0040}, \texttt{0x0000})$
$\begin{cases} x[7] = 0, \\ x[5] \oplus y[14] = 1, \\ x[15] \oplus y[8] = 0. \end{cases}$	$\begin{cases} x[7] = 0, \\ x[5] \oplus y[14] = 0, \\ x[15] \oplus y[8] = 0. \end{cases}$	$\begin{cases} x[7] = 0, \\ x[5] \oplus y[14] = 1, \\ x[15] \oplus y[8] = 1. \end{cases}$	$\begin{cases} x[7] = 0, \\ x[5] \oplus y[14] = 0, \\ x[15] \oplus y[8] = 1. \end{cases}$

Note that the first condition $x[7] = 0$ are shared among the four differentials, while for the other two conditions, they are complementary. Since the condi-

tions are linear, once a condition be fulfilled, the probability of the differential increases by a factor of 2^1. However, in the key-recovery attacks, because of the extended one round on top of these 3-round differentials, these conditions cannot be fulfilled by chosen data without guessing corresponding bits of k_0.

Table 4: (Probabilistic) (simultaneous-)neutral bit/bit-sets for 3-round differential $(\texttt{0x8020}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$, $(\texttt{0x8060}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$, $(\texttt{0x8021}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$, and $(\texttt{0x8061}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ of SPECK32/64

	(8020, 4101)		(8060, 4101)		(8021, 4101)		(8061, 4101)		
Bit-set	Pre.	Post.	Pre.	Post.	Pre.	Post.	Pre.	Post.	Condition
[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	$y[1] = 1$
[6, 11, 12, 18]	0.445	0.903	0.456	0.906	0.333	0.701	0.382	0.726	$x[2] \oplus y[11] = 0$
[4, 27, 29]	0.672	0.916	0.648	0.905	0.535	0.736	0.536	0.718	$x[11] \oplus y[4] = 1$

Pre.: probability obtained using 1000 correct pairs without fulfilling the conditions.
Post.: probability obtained using with 1000 correct pairs and fulfilling all the four conditions in the last column.
□: Neutral bit(-set)s used in the 13-round attack $\mathcal{A}^{\text{SPECK}13R}$ on SPECK32/64.
□: Neutral bit(-set)s used in the 12-round attack $\mathcal{A}^{\text{SPECK}12R}$ on SPECK32/64.

Exploiting Multiple Differentials. The four differentials share most of the high-probabilistic NB's and the conditions on the NB's (except for the [30], [0, 8, 31]). Besides, the neutral bit [22] makes $(\texttt{0x8020}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ and $(\texttt{0x8060}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ matched differentials, and $(\texttt{0x8021}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ and $(\texttt{0x8061}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ also matched differentials as introduced in Sect 3.3. More specifically, take $(\texttt{0x8020}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ and $(\texttt{0x8060}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ for example, they share neutral bit [22] and all other useful NB. Since $(\texttt{0x8020}, \texttt{0x4101}) \oplus (\texttt{0x8060}, \texttt{0x4101}) = (\texttt{0x0040}, \texttt{0000})$, while the neutral bit [22] corresponds to difference $\Delta_{22} = (\texttt{0x0040}, \texttt{0000})$, ciphertext structures for $(\texttt{0x8060}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ can be directly obtained from that of $(\texttt{0x8020}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ (refer to Sect. 3.3). Thus, using a pair of matched 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.

For the ease of notation, let us denote $(\texttt{0x8020}, \texttt{0x4101})$ as example difference Δ_E^1, and $(\texttt{0x8021}, \texttt{0x4101})$ as Δ_E^2. Six queries of a plaintext 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, equivalent with the saving by a factor of $2^{1.42}$.

Note that, to use these CNBS's, one has to guess the value corresponding to the conditions, *i.e.*, some key bits or their linear combinations. For example, guessing 4 linear combinations of key bits, one can additionally get 4 more NBS's of high probability; The more the guessed bits, the larger each cipher-structure one can expand to, thus the higher the success probability of the key-recovery attack. However, more guessed key bits also result in higher time and data complexities. Thus, one has to determine the trade-off between success rate and attack complexity through the number of guessed key bits of k_0.

4.2 Key Recovery Attack on 13-round SPECK32/64

Employing two classical differentials that have identical CNB's that have been identified using the above method, 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 classical differentials sharing the same output difference $(\texttt{0x8020}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ and $(\texttt{0x8060}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ (refer to the rounds colored in blue in Fig. 3), and the set of their 12 NBS's, *i.e.*, $\mathcal{NB}$: {[22], [13], [20], [5, 28], [15, 24], [12, 19], [6, 29], [6, 12, 11, 18], [4, 27, 29], [14, 21], [0, 8, 31], [30]} (refer to the columns framed by blue lines in Table 4);
2. An 8-round neural distinguisher $\mathcal{ND}^{\text{SPECK}_{8R}}$ trained with difference $(\texttt{0x0040}, \texttt{0x0000})$ and its wrong key response profiles $\mathcal{ND}^{\text{SPECK}_{8R}}.\mu$ and $\mathcal{ND}^{\text{SPECK}_{8R}}.\sigma$;
3. A 7-round neural distinguisher $\mathcal{ND}^{\text{SPECK}_{7R}}$ trained with difference $(\texttt{0x0040}, \texttt{0x0000})$ 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 bits of k_0, on which the conditions depend.
2. n_{cts}: the number of ciphertext structures.
3. n_b: the number of ciphertext pairs in each ciphertext structure, *i.e.*, $2^{|\mathcal{NB}|}$.
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 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 for guessing each of the last and the second last subkeys, respectively.

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

1. Initialize variables $Gbest_{\text{key}} \leftarrow (\text{None}, \text{None})$, $Gbest_{\text{score}} \leftarrow -\infty$.
2. For each of the n_{kg} values of the 6 key bits $k_0[7]$, $k_0[15] \oplus k_0[8]$, $k_0[12] \oplus k_0[5]$, $k_0[1]$, $k_0[2] \oplus k_0[11]$, $k_0[11] \oplus k_0[4]$,
   (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 $(\texttt{0x8020}, \texttt{0x4101})$, and satisfying the conditions for being conforming pairs, *i.e.*, $\begin{cases} \tilde{x}_1[7] = k_0[7], \\ \tilde{x}_1[15] \oplus \tilde{y}_1[8] = k_0[15] \oplus k_0[8], \end{cases}$ (refer to Table 3), and the conditions for increasing the neutrality probability of four bits *i.e.*, $\begin{cases} \tilde{x}_1[12] \oplus \tilde{y}_1[5] \oplus 1 = k_0[12] \oplus k_0[5], \\ \tilde{y}_1[1] \oplus 1 = k_0[1], \\ \tilde{x}_1[2] \oplus \tilde{y}_1[11] = k_0[2] \oplus k_0[11], \\ \tilde{x}_1[11] \oplus \tilde{y}_1[4] \oplus 1 = k_0[11] \oplus k_0[4], \end{cases}$ (refer to Table 4).
   (b) From the $n_{cts}/2$ random data pairs, generate $n_{cts}/2$ structures using the NBS's in $\mathcal{NB}$, marking the correspondence between old pairs and new pairs that are generated using the NB [22].
   (c) Decrypt one round using zero as the subkey for all data in the structures obtained above (*i.e.*, the round in green depicted in Fig. 3) and obtain $n_{cts}/2$ plaintext structures;
   (d) Query for the ciphertexts under 13-round SPECK32/64 of the $n_{cts}/2 \times n_b \times 2$ plaintexts, thus obtain $n_{cts}/2$ ciphertext structures.
   (e) 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 $(\tilde{x}_1||\tilde{y}_1, \tilde{x}_1||\tilde{y}_1 \oplus (\texttt{0x8020}, \texttt{0x4101}))$ and $(\tilde{x}_1||\tilde{y}_1 \oplus (\texttt{0x0040}, \texttt{0000}), \tilde{x}_1||\tilde{y}_1 \oplus (\texttt{0x8020}, \texttt{0x4101}) \oplus (\texttt{0x0040}, \texttt{0000}))$, 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 are corresponding to couples of plaintext pairs for the second differential $(\texttt{0x8060}, \texttt{0x4101})$ 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 $\{\mathcal{C}_1, \ldots, \mathcal{C}_{n_{cts}}\}$, are obtained.
   (f) Initialize an array $w_{\max}$ and an array n_{visit} to record the highest distinguisher score obtained so far and the number of visits have received in the last subkey search for the ciphertext structures.
   (g) Initialize variables $best_{\text{score}} \leftarrow -\infty$, $best_{\text{key}} \leftarrow (\text{None}, \text{None})$, $best_{\text{pos}} \leftarrow$ 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.
   (h) 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 $i \in \{1, \ldots, n_{cts}\}$, 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 [14].

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 Algorithm 3 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.*, $L_1 = \{(g_{1_i}, v_{1_i}) : i \in \{1, \ldots, n_{byit1} \times n_{cand1}\}\}$.

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 of the recommended last subkey $g_{1_i} \in L_1$, if the score $v_{1i} > c_1$,

A. Decrypt the ciphertexts in $\mathcal{C}$ using the g_{1_i} by one round and obtain the ciphertext structure $\mathcal{C}'$ of 12-round Speck32/64.

B. Run BayesianKeySearch Algorithm 3 with $\mathcal{C}'$, the neural distinguisher $\mathcal{ND}^{\text{Speck}_{7R}}$ and its wrong key response profile $\mathcal{ND}^{\text{Speck}_{7R}}.\mu$ and $\mathcal{ND}^{\text{Speck}_{7R}}.\sigma$, 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.*, $L_2 = \{(g_{2_i}, v_{2_i}) : i \in \{1, \ldots, n_{byit2} \times n_{cand2}\}\}$.

C. Find the maximum among v_{2i} and the corresponding g_{2_i} 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 2i.

(i) Make a final improvement using VerifierSearch [13] 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 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 4. In Gohr's implementations of the attack [13], two bits of g_1 are randomly assigned instead of being recommended by minimizing the weighted euclidean distance. This is based on observation on 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 last subkey, which is done using the stronger helper ND. The wrong key response profile with respect to the helper ND $\mathcal{ND}^{\text{Speck}_{7R}}.\mu$ and $\mathcal{ND}^{\text{Speck}_{7R}}.\sigma$ is 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 $\mathcal{A}^{\text{SPECK}_{13R}}$, the 8-round and 7-round neural distinguishers provided in [13] were used. The accuracy of $\mathcal{ND}^{\text{SPECK}_{8R}}$ is about 0.514, and that of $\mathcal{ND}^{\text{SPECK}_{7R}}$ is about 0.616. Concrete parameters and the complexity of $\mathcal{A}^{\text{SPECK}_{13R}}$ are as follows (see Figure 5).

$$n_{kg} = 2^6, \quad n_b = 2^{12}, \quad n_{cts} = 2^{11}, \quad n_{it} = 4 \times n_{cts}$$
$$c_1 = 20, \quad c_2 = -500, \quad n_{byit1} = n_{byit2} = 5, \quad n_{cand1} = 2 \times n_{cand2} = 64$$

The data complexity is $n_{kg} \times n_b \times n_{cts}$, that is, $2^{6+11+12}$, *i.e.*, 2^{29} plaintexts (because of the using 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 six conditions being fulfilled only. That is, tested whether a particular one of 2^6 loops in Step 2 can successfully recover the last two subkeys. In that particular loop, the trialed value of the 6 bits of k_0 is real. In other loops, the trialed values deviate from the real 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, because the prepended classical differentials are valid to keys fulfilling $k_2[12] \neq k_2[11]$, thus we tested for these valid keys only, and the presented attack works for 2^{63} keys (refer to Sect. C).

The core of the attack was examined in 80 trials. Among the 80 trials, 2 trials failed because of no correct ciphertext structure as for the prepended differential. We count a key guess as successful if the sum of the Hamming weights of the differences between the returned last two subkeys and the real two subkeys is at most two. Within the remaining 78 trials in which the neural distinguishers are called, there are 49 succeeded trials. Thus, the success rate is 49/80, which is 0.6125.

The trials were executed using a server with 8 GPUs[3]. The maximum execution time among the 80 runs is 10.2 hours (which runs all the n_{it}, *i.e.*, 8192 iterations). For 2^6 loops in Step 2, the worst situation is that within each loop, all n_{it} iterations are executed. Accordingly, the full attack requires about $2^6 \times 10.2$, *i.e.*, 652.8 GPU hours, which is equivalent to $2^{49.16+r}$ executions of SPECK32/64 [4].

Trade-offs. Using double the amount of data (*i.e.*, $n_{cts} = 2^{12}$), reducing n_{cand1} by half (*i.e.*, $n_{cand1} = 32$) and lower cutoff c_1 to 18, the success rate can be increased to 0.75 without doubling the time (equivalent to $2^{49.84+r}$ executions of SPECK32/64) (see Fig. 4).

4.3 Key Recovery Attack on 12-round Speck32/64

To devise key-recovery attack on 12-round SPECK32/64, Gohr in [14] used the 2-round classical differential $(\texttt{0x0211}, \texttt{0x0a04}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ combined with

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

[4] 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, $r = 2.4$

the 8-round and 7-round neural distinguishers. For amplifying the weak signal from the 8-round neural distinguisher, 13 single-bit NB's of the prepended 2-round classical differential were exploited. However, many of the 13 NB's are neutral with probabilities that are not high (refer to App. Table 10). 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, it can be seen that there are SNBS's that are completely neutral or are neutral with high probability. Using 13 SNBS's from Table 2, cutting the required data by nearly half, and using the following parameters, our experiments show that the success rate of the resulted attack can be increased to 0.86 using fewer data (see Fig. 6).

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

However, the data complexity is still bounded by the weakness of the 8-round distinguisher. To further reduce the data requirement, we considered combining the 3-round classical differential and the stronger 7-round (and 6-round) neural distinguisher. In this case, unconditional SNBS's are enough for the 7-round neural distinguisher. Thus, those conditional ones can be dismissed in such a 12-round attack. Therefore, all the four 3-round differentials sharing the many NB's can be employed, which makes it possible to obtain one plaintext pair with 3/4 instead of 2 queries (*i.e.*, by obtaining 8 ciphertext pairs with 6 queries as introduced in Sect. 4.1).

Concretely, the components of the 12-round key-recovery attack on SPECK32/64, denoted by $\mathcal{A}^{\text{SPECK}_{12R}}$, are as follows.

1. Four 3-round classical differentials $(\texttt{0x8020}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$, $(\texttt{0x8060}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$, $(\texttt{0x8021}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$, $(\texttt{0x8061}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ and the set of their 6 neutral bit(-set)s, *i.e.*, $\mathcal{NB}$: {[22], [13], [20], [12, 19], [14, 21], [6, 29]} (refer to the rows framed by green lines in Table 4);
2. A 7-round neural distinguisher $\mathcal{ND}^{\text{SPECK}_{7R}}$ trained with difference $(\texttt{0x0040}, \texttt{0x0000})$ and its wrong key response profiles $\mathcal{ND}^{\text{SPECK}_{7R}}.\mu$ and $\mathcal{ND}^{\text{SPECK}_{7R}}.\sigma$;
3. A 6-round neural distinguisher $\mathcal{ND}^{\text{SPECK}_{6R}}$ trained with difference $(\texttt{0x0040}, \texttt{0x0000})$ 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 $\mathcal{A}^{\text{SPECK}_{13R}}$ (refer to Fig. 2). The difference is that, at the beginning, we only guess one key bit of k_0, that is $k_0[7]$, because for all four 3-round differentials, the common condition for correct pairs is $x_1[7] = 0$ (refer to Table 3). Thus, n_{kg} is 2, and there are only 2 outermost loops.

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

$$\begin{array}{llll} n_{kg}=2^1, & n_b=2^6, & n_{cts}=2^{12}, & n_{it}=2^{13} \\ c_1=7, & c_2=10, & n_{byit1}=n_{byit2}=5, & n_{cand1}=2\times n_{cand2}=64 \end{array}$$

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 CPU. Concretely, 160 trials were done with 32 threads in a CPU server[5]. Within the 160 trials, 3 trials have no correct ciphertext structures. In the remaining 157 trials, there are 130 successful trials (the returned last two subkeys have a Hamming distance to the real subkeys at most two). The success rate is 130/160, *i.e.*, 0.8125.

The maximum execution time among the trials is 5.8 hours (which runs all the n_{it}, *i.e.*, 8192 iterations). For 2^1 loops in Step 2, the maximum run time should be about 11.6 CPU hours, which is equivalent to $2^{43.35}$ executions of SPECK32/64.

Trade-off. If accepting a success rate of 0.5625, the data complexity can be further reduced to $2^{17.58}$ (by setting $n_{cts} = 2^{11}$) (see Fig. 8).

5 Turning Parameters for the Key Recovery Attacks

The key-recovery attack with Upper Confidence Bounds and BAYESIANKEY-SEARCH has shown its effectiveness in guessing keys in [14] and this work. However, the turning 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 important observations to bring some light on how to tune important parameters and make 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 BAYESIANKEY-SEARCH. 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 of 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 4 to 8).

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

– $\mathcal{D}_w^{v_{1\max}}$, $\mathcal{D}_r^{v_{1\max}}$, $\mathcal{D}_s^{v_{1\max}}$: 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 (the score of the subkey before final improvement);
– 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 $\mathcal{D}_r^{v_{2\max}}$, $\mathcal{D}_w^{v_{2\max}}$, $\mathcal{D}_s^{v_{2\max}}$)[6];
– distribution of Hamming distances between the returned subkeys and the real subkeys;
– distribution of the used number of iterations in successful attacks.

5.2 Some rules of thumb

Apart from substantial illustrations on previously hidden details of the key-recovery phase, the following observations are made to provide some rules of thumb on 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

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

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

[6] Some $v_{2\max}$'s corresponding to success cases are lower than cutoff c_2, that is due to the final improvement.

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 20 so that q is 0.32; then, to have a success probability of 0.61, the required number of ciphertext structures, *i.e.*, n_{cts} should satisfy $n_{cts} \geq \log 2(1-0.61)/\log 2(1-2^{-9}\cdot 0.32) = 1506$. On the other hand, suppose one selects n_{cts} to be 2^{11}, and aims P_s to be 0.61; since p is 2^{-9}, this requires $c_1 \leq Q(1-(1-(1-0.61)^{2^{-11}})/2^{-9}) = Q(1-0.24) = 22$.

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.

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 hard to produce good recommendations for the second last subkeys. On the other hand, too small cutoff c_1 results in a high percentage of v_1 from wrong ciphertext structures passing it. As a consequence, a lot of running time will be wasted on the wrong ciphertext structures. Thus, 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 3 bits Hamming distance towards the real subkey) from both correct and wrong ciphertext structures 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 $\mathcal{D}_w^{v_{1\max}}$.

5.3 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 analyze the values of $v_{1\max}$ statistically. 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 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 $\mathcal{D}_w^{v_{1\max}}$ are 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 14).

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 dis-*

tribution [7] *(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 distribution 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 $KL(\texttt{real}||\texttt{rand})$ in 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 $KL(\texttt{real}||\texttt{rand})$ increases considerably. Increasing from 2^6 to 2^7 (Fig. 9c and 9e), the mean increase approximately 2 times, however, the $KL(\texttt{real}||\texttt{rand})$ 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 wrong ciphertext structures pass, and 30% from correct ciphertext structures pass; whereas when $n_b = 2^6$, 9% from wrong ciphertext structures pass and 35% from 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, turning n_{cand} could achieve better trade-offs between time complexity and success rate than turning 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.

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

6 Neural Distinguishers on Round-Reduced SIMON32/64

This section presents the neural distinguishers on SIMON32/64 obtained in this work, using which, key-recovery attacks covering 16 rounds are devised and presented in Sect. D. The advantage in data complexity further confirms that machine learning can produce powerful cryptographic distinguishers that can be used to devise efficient key-recovery attacks. Besides, DDT-based DD's are computed and provide baselines for ND's. Comparisons between DD's and 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) [17] used in [14] were conducted. Specifically, Dense Network (DenseNet) [19] shows advantages in parameter efficiency, implicit deep supervision, and feature reuse. Squeeze-and-Excitation Network (SENet) [18] won the first place in the *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC 2017) for classification task. SENet can also be combined with existing deep architectures to boost performance at minimal additional computational cost. 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 [14]. All three schemes are attempted. For short rounds, the basic training scheme already works well. For longer rounds, 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 to do key-recovery attacks. Thus, we trained distinguishers accepting data composed with partial values and partial differences.

Training using the basic scheme. Using the basic training scheme and adopting SENet, neural distinguishers to recognize output pairs of 7-, 8-, 9-round SIMON32/64 with the input difference $(\texttt{0x0000}, \texttt{0x0040})$ 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 $(\texttt{0x0000}, \texttt{0x0040})$ 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. Summaries are presented in Table 5 for detailed accuracy and in Fig. 19 for the wrong key response profile.

Training to simulate KeyAverageing algorithm. A successful training of the 10-round distinguisher is achieved by adopting the training scheme of simulating a KeyAverageing Algorithm [14] 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 $(\texttt{0x0000}, \texttt{0x0040})$ 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 [14], is performed from a randomly initialized network state. This training procedure results in a 10-round distinguisher, named as $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{10R}}$ with accuracy 0.5551, as summarized in Table 5 for detailed accuracy and Fig. 19 for the wrong key response profile.

Training using the Staged Training Method. The best 11-round distinguisher that is successfully used in a practical key-recovery attack, is trained using the staged training method, which was the same method used to train the 8-round distinguisher of SPECK32/64 in [14]. Concretely, in the first stage, the best 9-round distinguisher $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{9R}}$ is retained to recognize 8-round SIMON32/64 with the input difference $(\texttt{0x0440}, \texttt{0x0100})$. Note that, the most likely difference to appear three rounds after the input difference $(\texttt{0x0000}, \texttt{0x0040})$ is $(\texttt{0x0440}, \texttt{0x0100})$, 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 $(\texttt{0x0000}, \texttt{0x0040})$. For this training, 2^{30} examples are freshly generated and fed, and 2^{28} examples are for verification. One epoch with learning rate 10^{-4} is done. In the last stage, the resulted 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 resulted distinguisher $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{11R}}$ achieves an accuracy 0.5173 (refer to Table 5 for detailed accuracy and Fig. 19 for the wrong key response profile.)

Table 5: Summary of neural distinguishers on SIMON32/64

#R	Name	Network	Accuracy	True Positive Rate	True Negative Rate
6	$\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}6R}$	DDT	0.9918	0.9995	0.9841
7		ResNet	$0.9823 \pm 1.2 \times 10^{-4}$	$0.9996 \pm 2.7 \times 10^{-5}$	$0.9650 \pm 2.3 \times 10^{-4}$
	$\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}7R}$	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	$\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}7R}$	DDT	0.8465	0.8641	0.8288
8	$\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}8R}$	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	$\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}8R}$	DDT	0.6628	0.5781	0.7476
8	$\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}8R}$	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	$\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}9R}$	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	$\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}9R}$	DDT	0.5683	0.4691	0.6674
9	$\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}9R}$	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	$\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}10R}$ +	SENet	$0.5610 \pm 4.5 \times 10^{-4}$	$0.4761 \pm 6.0 \times 10^{-4}$	$0.6460 \pm 7.2 \times 10^{-4}$
	$\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}10R}$ *	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	$\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}10R}$	DDT	0.5203	0.5002	0.5404
11	$\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}11R}$	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	$\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}11R}$	DDT	0.5044	0.4852	0.5236

The network structure and parameters for the ResNet follow exactly that used in [13] for training the 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 ND's on SIMON32/64.
* This neural distinguisher is trained using the KEYAVERAGING algorithm.
+ This neural distinguisher is trained using the staged training method.

Table 6: Comparing ND and 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, 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 [14]: 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	$\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}8R}$	11.8 ± 3.1	1.0	0.238
	$\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}8R}$	43.9 ± 21.4	2.0	0.188
64×2	$\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}8R}$	0.9 ± 0.2	1.0	0.415
	$\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}8R}$	1.3 ± 0.2	1.0	0.335

Training directly 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 $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{(r-1)R}}$ and the latter by $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{rR}}$.

The results show that $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{(r-1)R}}$ indeed could achieve slightly better accuracy than $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{rR}}$ (refer to Table 5). Besides, from Fig. 19, the wrong key response profiles of $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{8R}}$ (Fig. 19b) and that of $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{9R}}$ (Fig. 19c) share observable pattern and symmetry. For key values that have little different from the real value, responses from $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{8R}}$ are higher than responses from $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{9R}}$. Similar observations can be derived from a comparison between that of $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{9R}}$ (Fig. 19d) and that of $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{10R}}$ (Fig. 19e). Thus, these $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{(r-1)R}}$ trained using data of form $(x, x', y \oplus y')$ were used in the key-recovery attacks presented in Sect. D.

6.3 Computing DD's and Further Interpretations

To provide baselines for ND's, we calculate the full distribution of differences for SIMON32/64 induced by the input difference `0x0000/0040` up to 11 rounds (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 [20]. Note that, the fed data to r-round ND are values of ciphertexts, from which, for SIMON32/64, one can directly compute the differences on $(r-1)$-round outputs without knowing the subkey. Thus, $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{rR}}$ or $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{r-1R}}$ should be compared with $\mathcal{ND}_{\mathbf{DD}}^{\text{SIMON}_{(r-1)R}}$.

The results show that $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{rR}}$ and $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{(r-1)R}}$ achieve similar but weaker classification accuracy than $\mathcal{ND}_{\mathbf{DD}}^{\text{SIMON}_{(r-1)R}}$. To further evaluate the gaps between the advantage of DD over ND, we devised a key ranking task, as done by Gohr for comparing ND's and DD's on SPECK32/64 in [14]. Specifically, a simple key ranking procedure to recover the last subkey on 11-round SIMON32/64 can be performed both by $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{8R}}$ or $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{8R}}$ in a configuration of 1+8+2. Table 6 shows the performance of $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{8R}}$ and $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{8R}}$ 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 rate of around 20%. However, $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{8R}}$ is slightly inferior to $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{8R}}$. To achieving the same success rate, $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{8R}}$ requires more data than $\mathcal{DD}_{\mathbf{DD}}^{\text{SIMON}_{8R}}$, but the difference is less than twice.

These comparisons suggest that r-round ND can "decrypt" one un-keyed 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 distribution of differences.

Remark 5. 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 ND is the r-round DD. From [14], r-round ND learns additional features beyond differences and has better classification accuracy than r-round DD. We conjecture that the mean reason is that, for SPECK, pure XOR-difference DD's cannot provide the best baselines for 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 [11, 21] 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 ND's on SPECK32/64 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 ND's can "learn to decrypt up to the values not messed up by outer subkey" might due to the design by Gohr as explained in [14] as "the use of the initial width-1 convolutional layer is intended to make the learning of simple bit-sliced functions such as bitwise addition easier". Remarkably, for SIMON32/64, the ND's seems to have also successfully peeled off the nonlinear bitwise AND layer in the last round.

These experiments on SIMON and comparisons with SPECK suggest that differential-based neural-distinguishers might work well in general on modern ciphers. Still, they might not always be superior to their classical counterparts. Their advantages might be easier to show on ARX ciphers, for which differential propagation has not been accurately evaluated using existing tools.

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. 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.
6. C. Beierle, G. Leander, and Y. Todo. Improved differential-linear attacks with applications to ARX ciphers. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 329–358. Springer, Heidelberg, Aug. 2020.
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. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. A. Gohr. Implementation of the Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning. GitHub Repository. https://github.com/agohr/deep_speck, 2019.
14. 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.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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.

20. 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.
21. 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.
22. J. Rijsdijk, L. Wu, G. Perin, and S. Picek. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *IACR TCHES*, 2021(3):677–707, 2021. https://tches.iacr.org/index.php/TCHES/article/view/8989.
23. 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.
24. 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.
25. 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.
26. Stefan Kölbl. CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives. https://github.com/kste/cryptosmt.
27. 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.

Supplementary Material

A Illustrations for Attack Procedures and for Turning Parameters for the Key Recovery Attacks

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

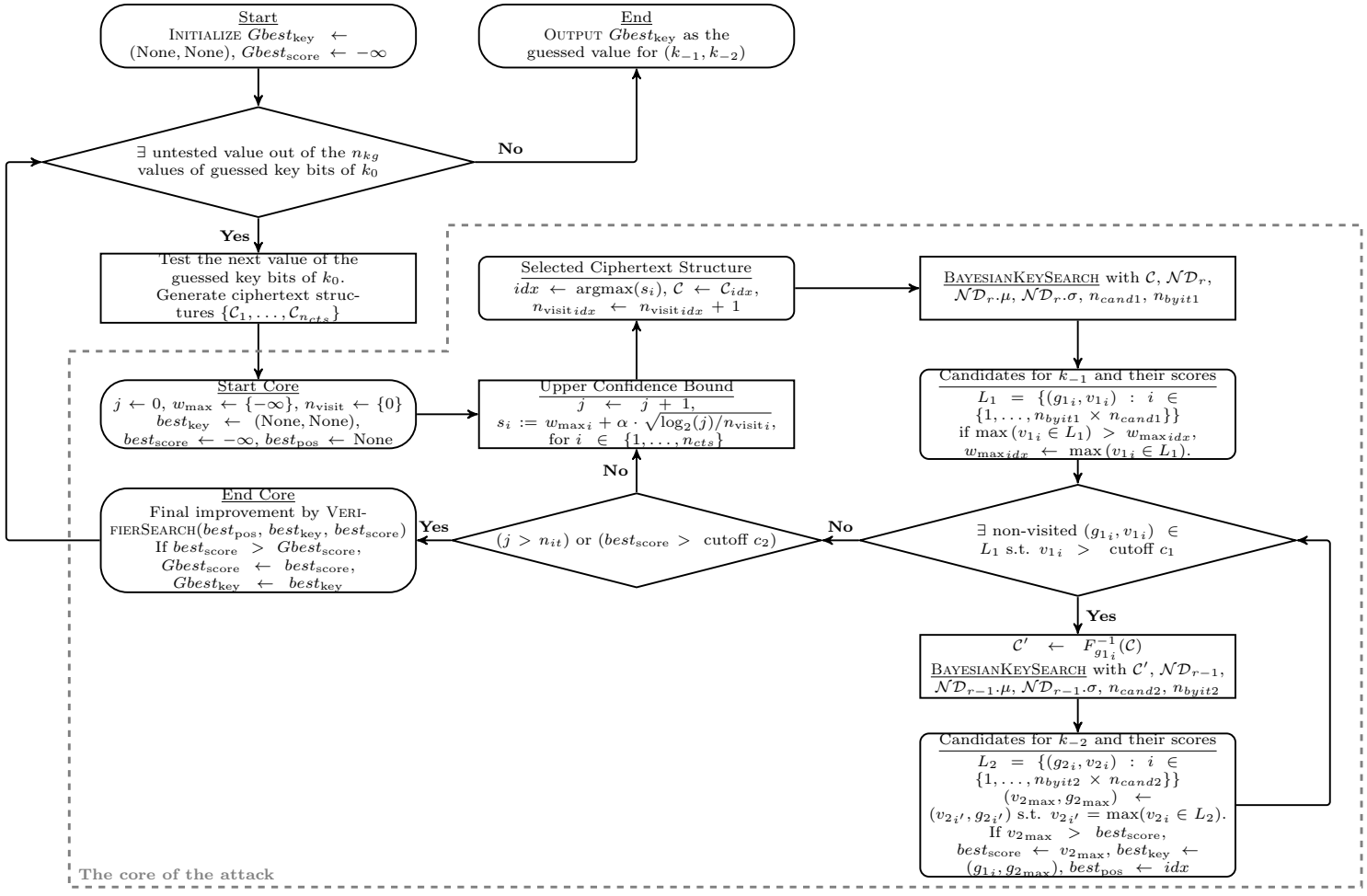


Fig. 2: Framework of the key-recovery attacks

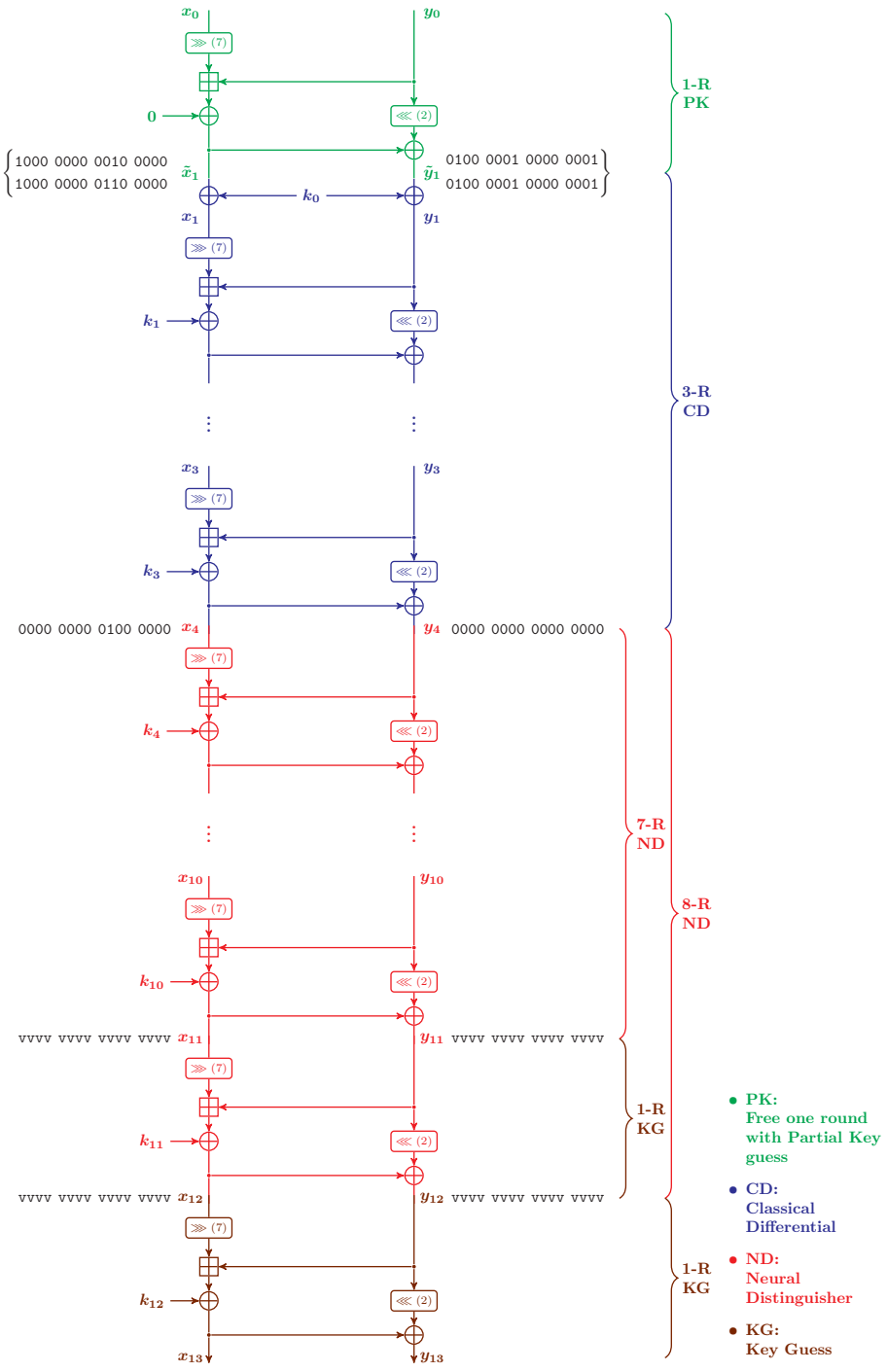
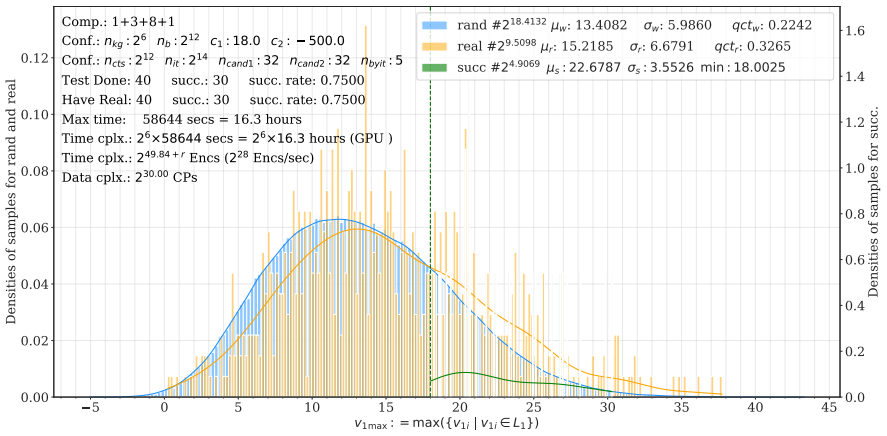
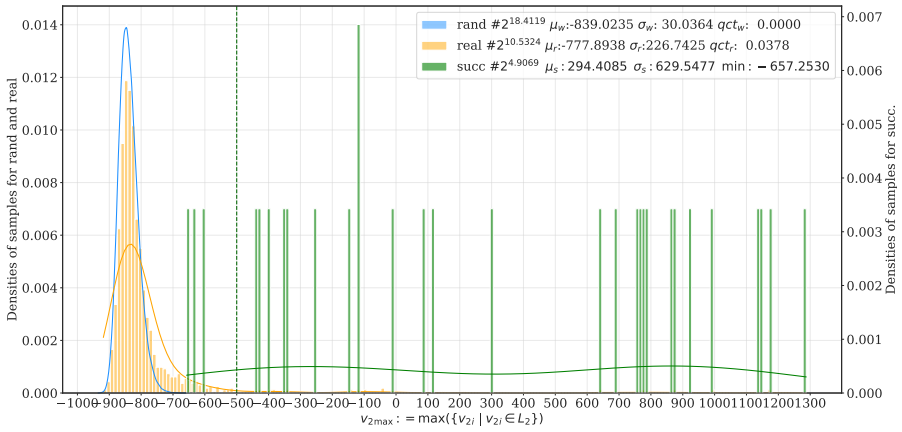


Fig. 3: Components for key-recovery attack on 13-round SPECK32/64

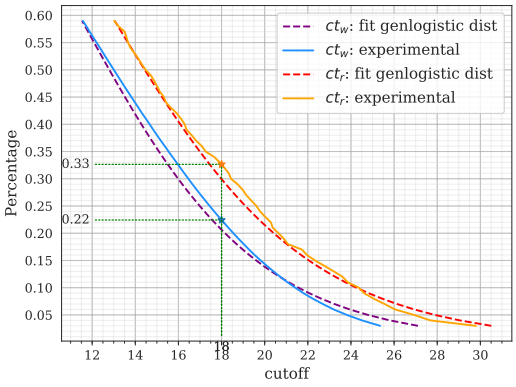
A.2 Visualizing the distributions of important statistics for various attacks



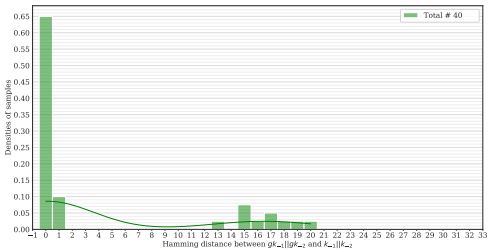
(a) Attack information and distributions of $v_{1\max}$



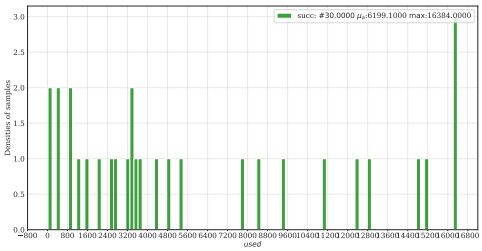
(b) Distributions of $v_{2\max}$ during the attack and those when successfully recovered the key



(c) Percentage of samples passing various cutoffs in 4a



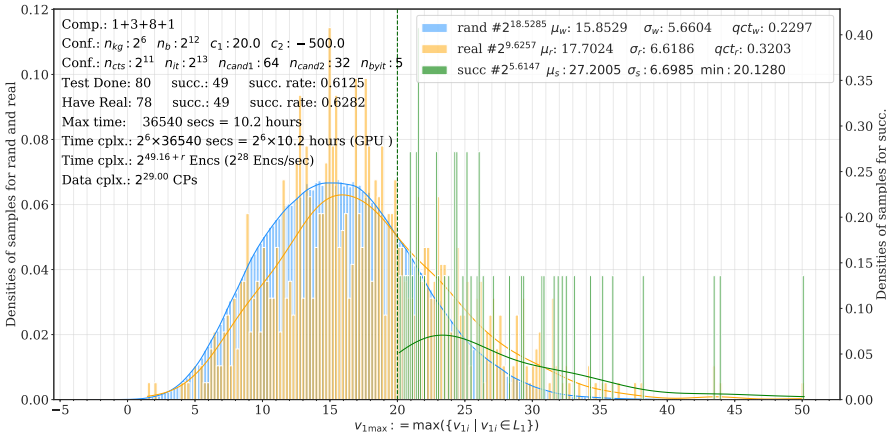
(d) Hamming distances between $gk_{-1}||gk_{-2}$ and $k_{-1}||k_{-2}$



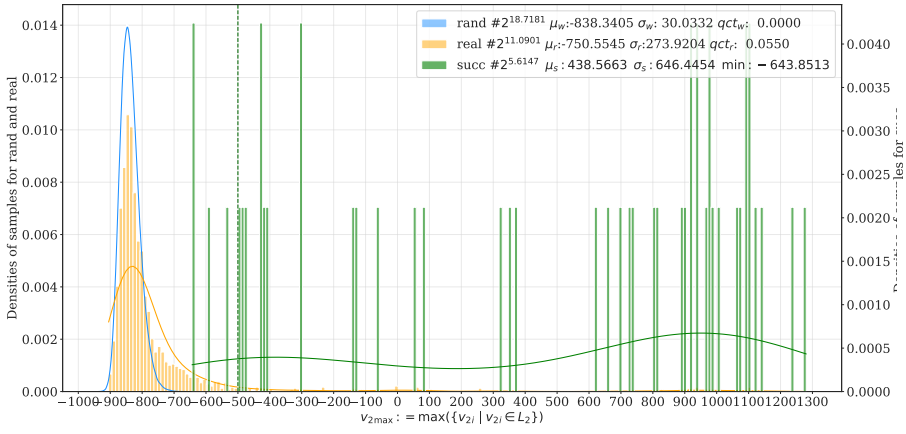
(e) Used number of iterations before return

Fig. 4: Detailed information for attack $\mathcal{A}_I^{\text{SPECK}_{13R}}$

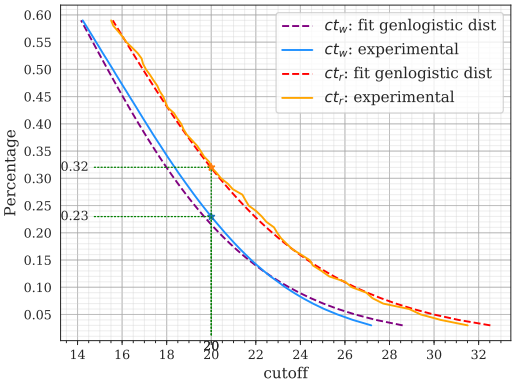
A.3 Illustrations for Investigations on $\mathcal{D}_{\boldsymbol{r}}^{v_{1\max}}$ and $\mathcal{D}_{\boldsymbol{w}}^{v_{1\max}}$



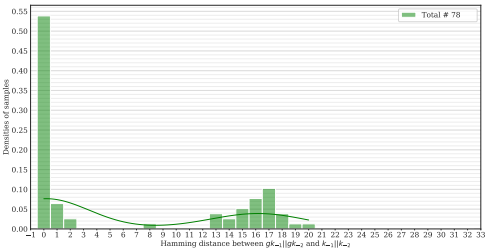
(a) Attack information and distributions of $v_{1\max}$



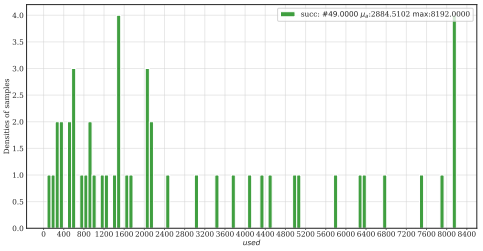
(b) Distributions of $v_{2\max}$ during the attack and those when successfully recovered the key



(c) Percentage of samples passing various cutoffs in 5a

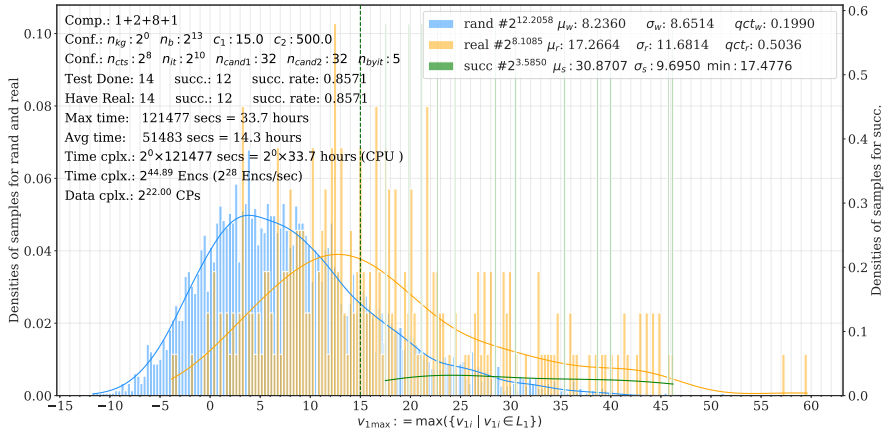


(d) Hamming distances between $gk_{-1}||gk_{-2}$ and $k_{-1}||k_{-2}$

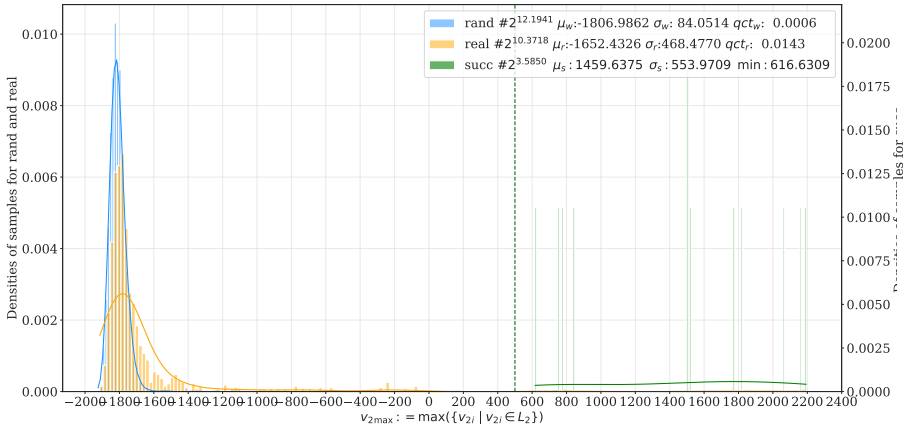


(e) Used number of iterations before return

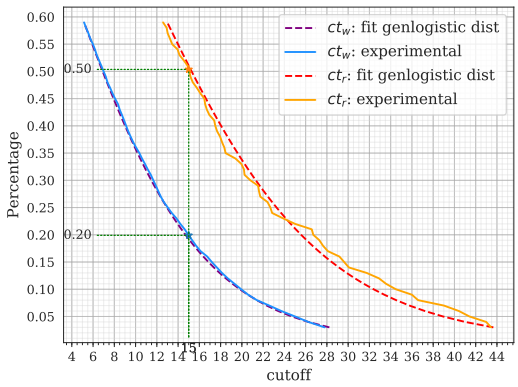
Fig. 5: Detailed information for attack $\mathcal{A}_{II}^{\text{SPECK}_{13R}}$



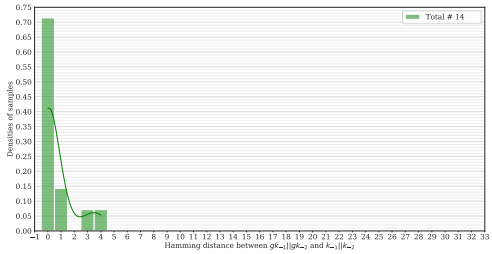
(a) Attack information and distributions of $v_{1\max}$



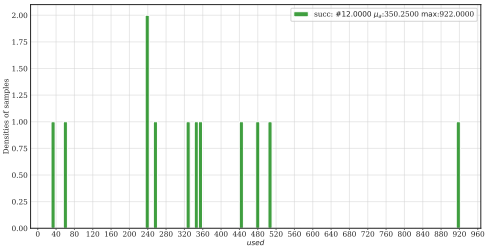
(b) Distributions of $v_{2\max}$ during the attack and those when successfully recovered the key



(c) Percentage of samples passing various cutoffs in 6a

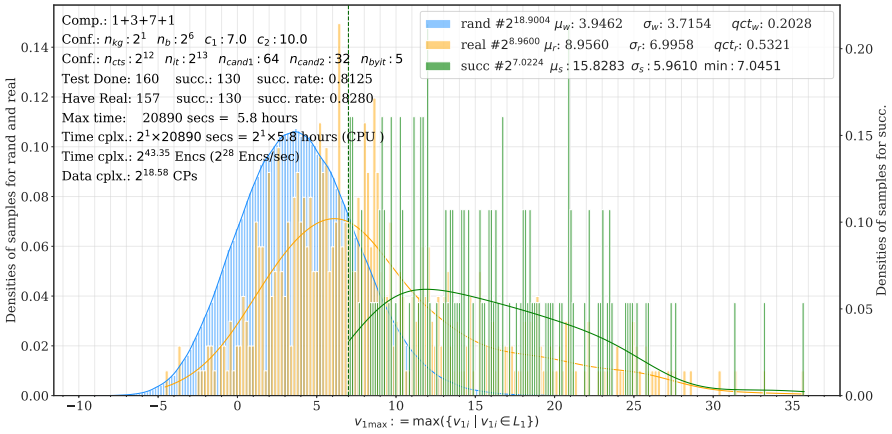


(d) Hamming distances between $gk_{-1}||gk_{-2}$ and $k_{-1}||k_{-2}$

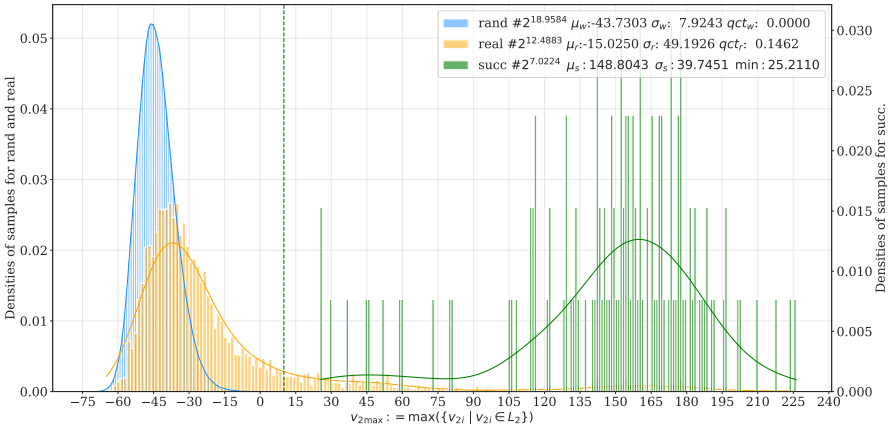


(e) Used number of iterations before return

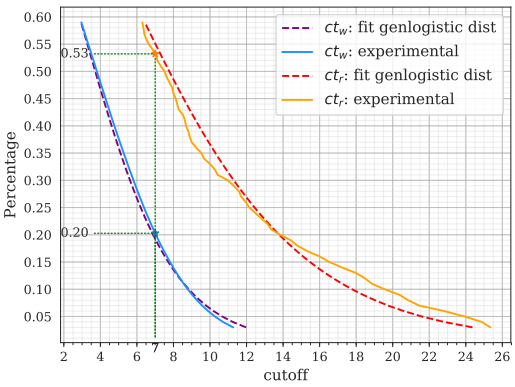
Fig. 6: Detailed information for attack $\mathcal{A}_I^{\text{SPECK}12R}$



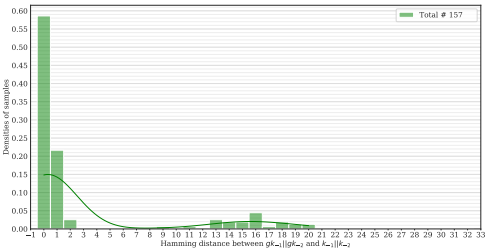
(a) Attack information and distributions of $v_{1\max}$



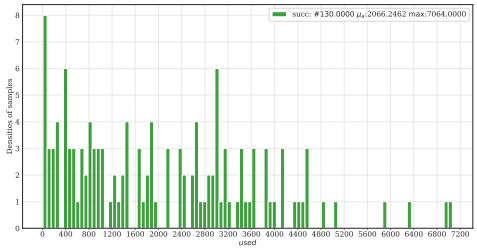
(b) Distributions of $v_{2\max}$ during the attack and those when successfully recovered the key



(c) Percentage of samples passing various cutoffs in 7a

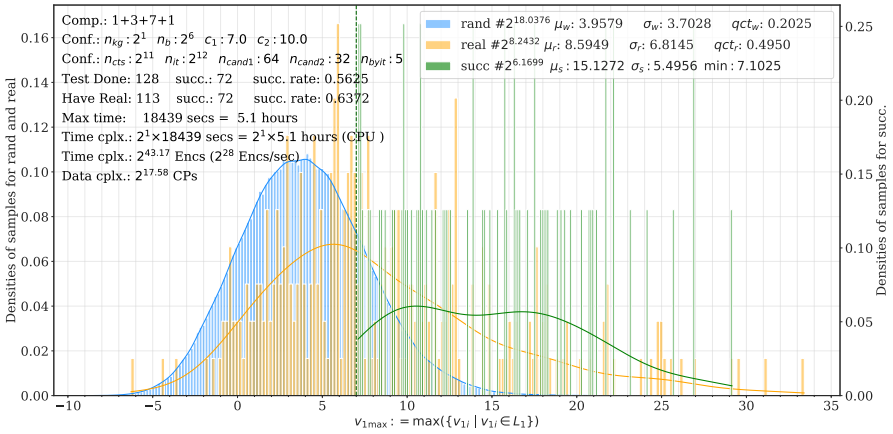


(d) Hamming distances between $gk_{-1}||gk_{-2}$ and $k_{-1}||k_{-2}$

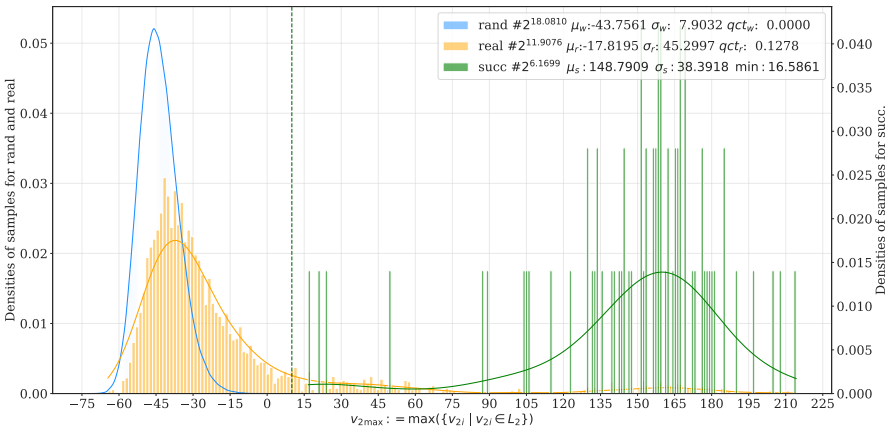


(e) Used number of iterations before return

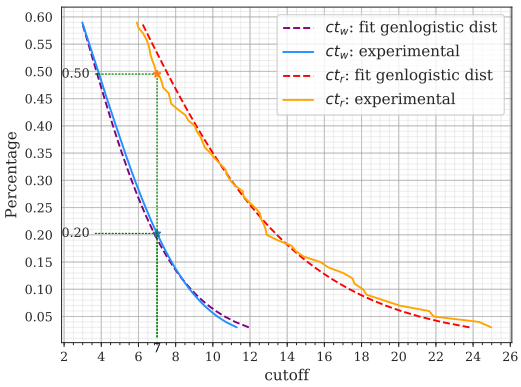
Fig. 7: Detailed information for attack $\mathcal{A}_{II}^{\text{SPECK}12R}$



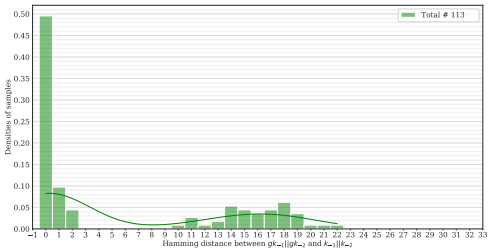
(a) Attack information and distributions of $v_{1\max}$



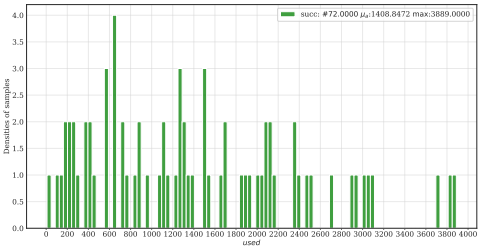
(b) Distributions of $v_{2\max}$ during the attack and those when successfully recovered the key



(c) Percentage of samples passing various cutoffs in 8a

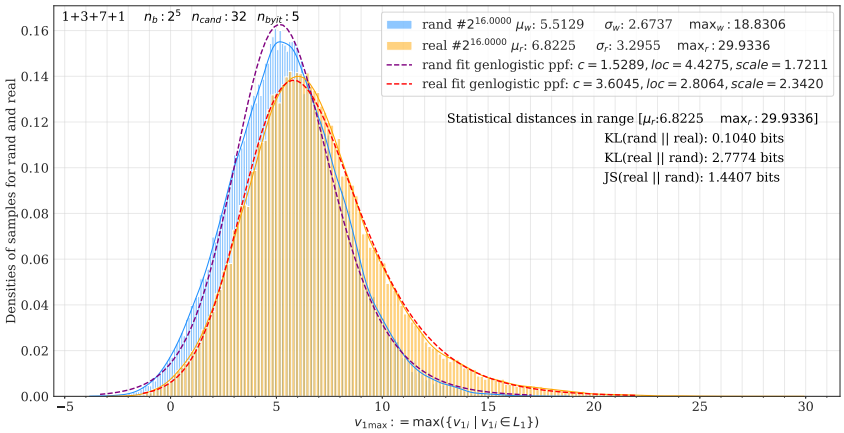


(d) Hamming distances between $gk_{-1}||gk_{-2}$ and $k_{-1}||k_{-2}$

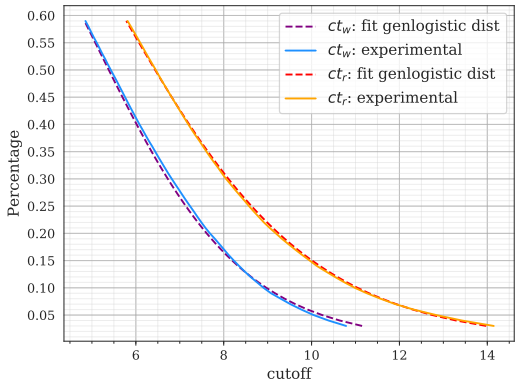


(e) Used number of iterations before return

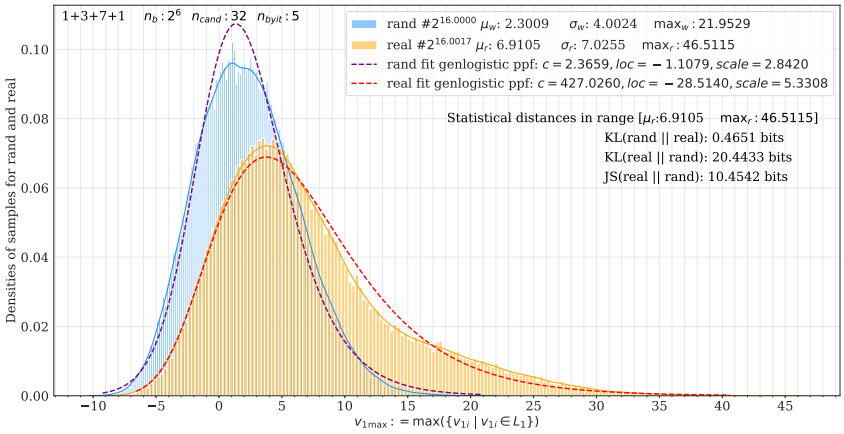
Fig. 8: Detailed information for attack $\mathcal{A}_{III}^{\text{SPECK}12R}$



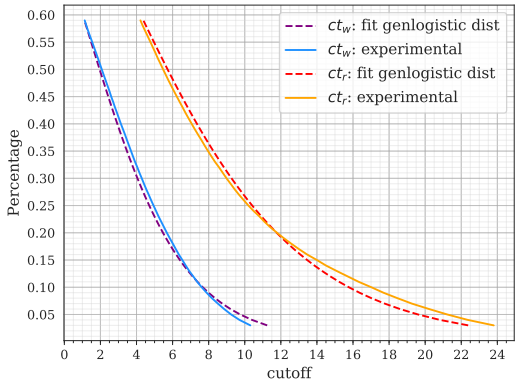
(a) Distributions $\mathcal{D}_r^{v1\max}$ and $\mathcal{D}_w^{v1\max}$ when using 5 NBs



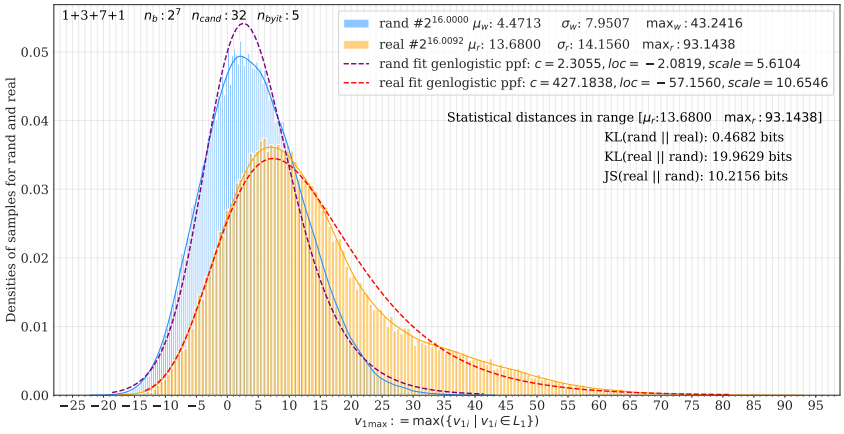
(b) Percentage of samples passing various cutoffs in 9a



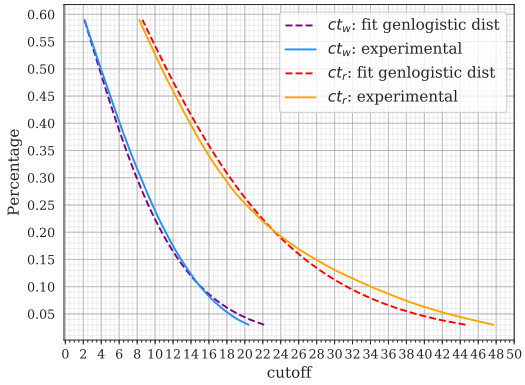
(c) Distributions $\mathcal{D}_r^{v1\max}$ and $\mathcal{D}_w^{v1\max}$ when using 6 NBs



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

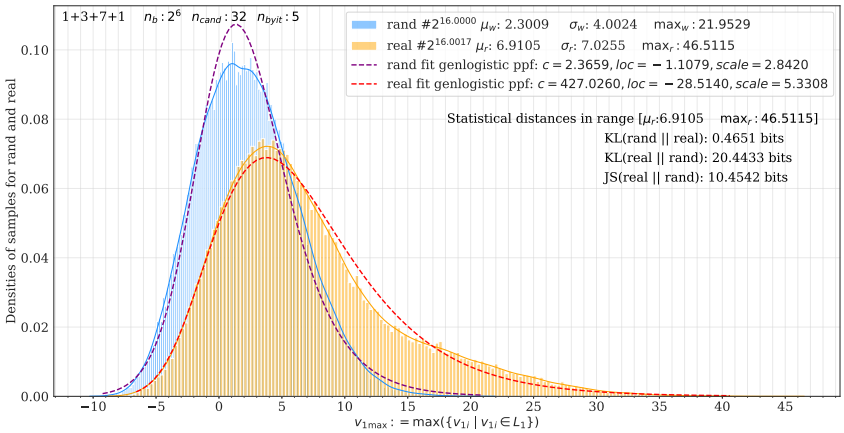


(e) Distributions $\mathcal{D}_r^{v1\max}$ and $\mathcal{D}_w^{v1\max}$ when using 7 NBs

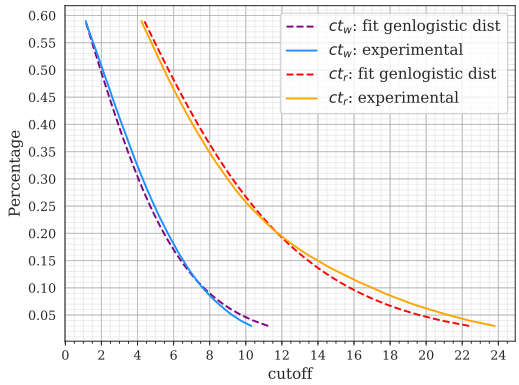


(f) Percentage of samples passing various cutoffs in 9e

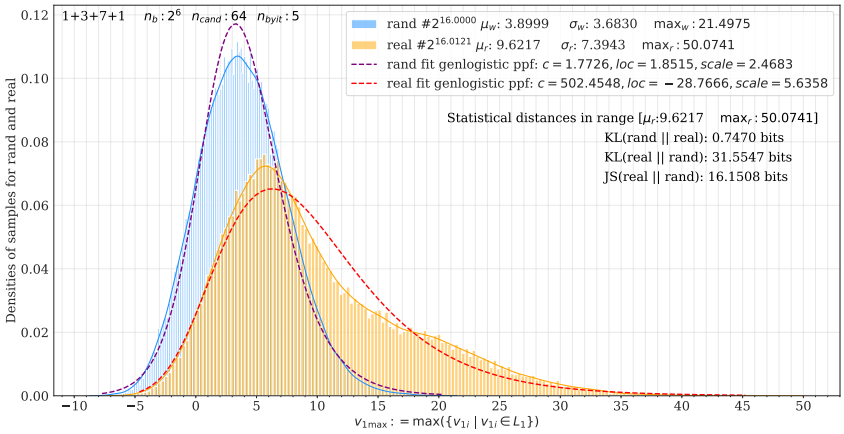
Fig. 9: Distributions of $v_{1\max}$ from correct ciphertext structures (real) and from wrong ciphertext structures (rand) of size n_b for $n_b \in \{2^5, 2^6, 2^7\}$



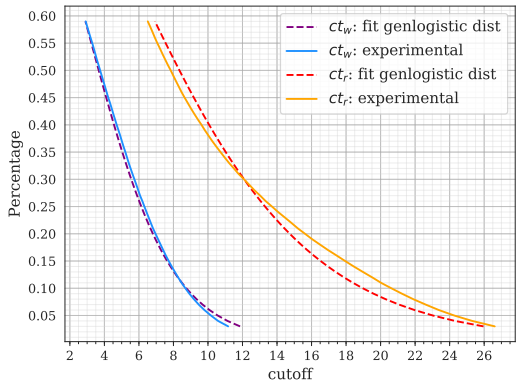
(a) Distributions $\mathcal{D}_r^{v_1\max}$ and $\mathcal{D}_w^{v_1\max}$ when $n_b = 2^6$, $n_{cand} = 32$, $n_{byit} = 5$



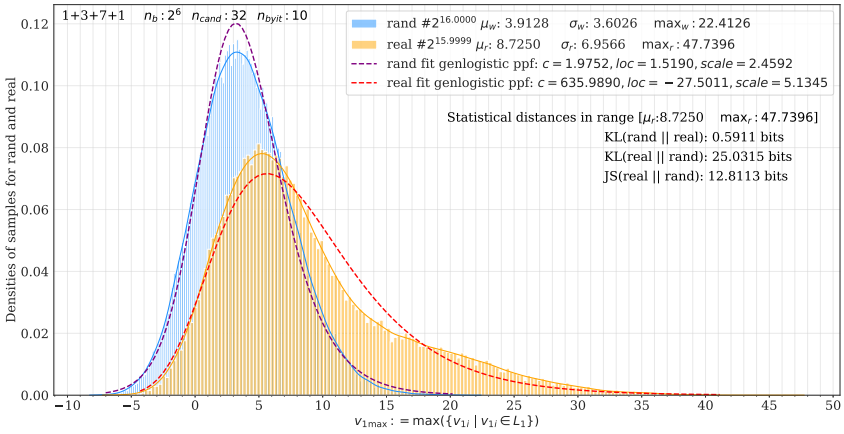
(b) Percentage of samples passing various cutoffs in 10a



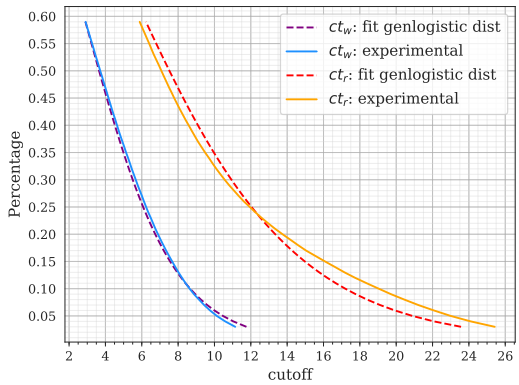
(c) Distributions $\mathcal{D}_r^{v_1\max}$ and $\mathcal{D}_w^{v_1\max}$ when $n_b = 2^6$, $n_{cand} = 64$, $n_{byit} = 5$



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

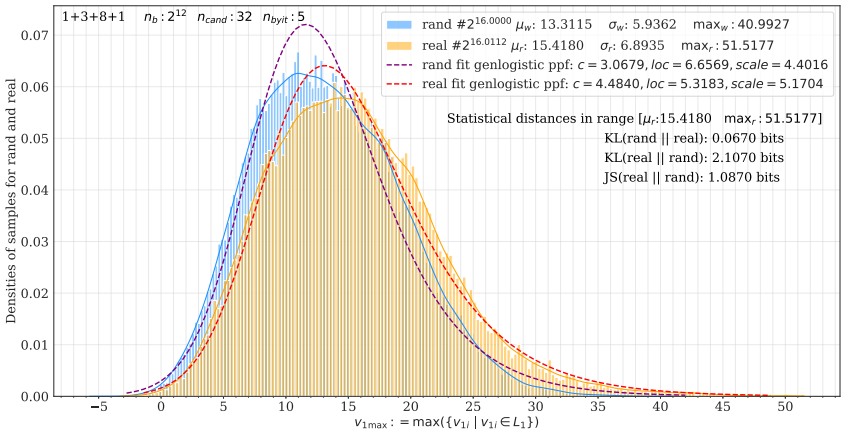


(e) Distributions $\mathcal{D}_r^{v_1\max}$ and $\mathcal{D}_w^{v_1\max}$ when $n_b = 2^6$, $n_{cand} = 32$, $n_{byit} = 10$

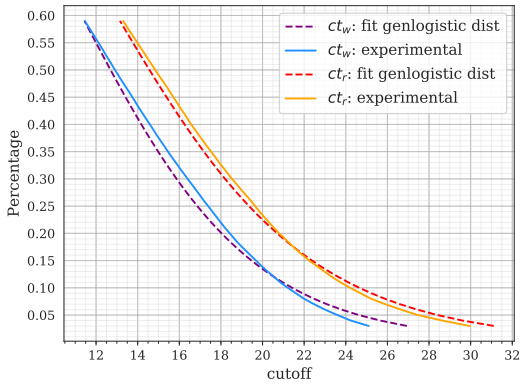


(f) Percentage of samples passing various cutoffs in 10e

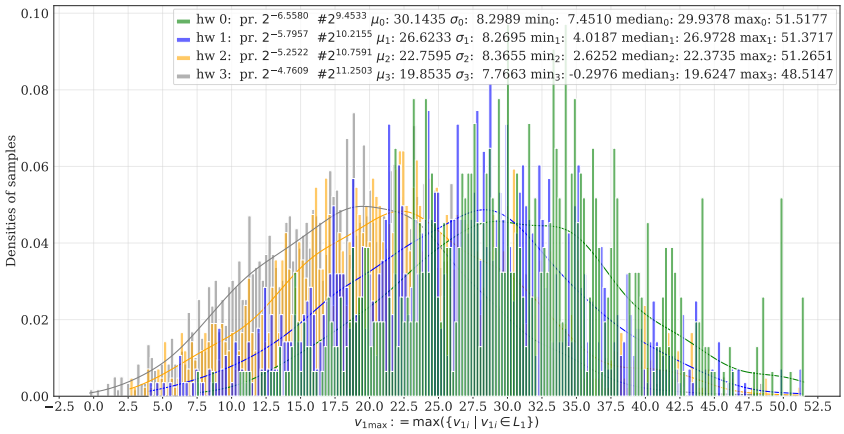
Fig. 10: Distributions of $v_{1\max}$ from correct ciphertext structures (`real`) and from wrong ciphertext structures (`rand`) with n_b fix to 2^6, $n_{cand} \in \{32, 64\}$, $n_b \in \{5, 10\}$



(a) Sampling 2^{16} correct/wrong ciphertext structures to study the distributions $\mathcal{D}_r^{v_{1\max}}$ and $\mathcal{D}_w^{v_{1\max}}$ involved in attack $\mathcal{A}^{\text{SPECK}13R}$

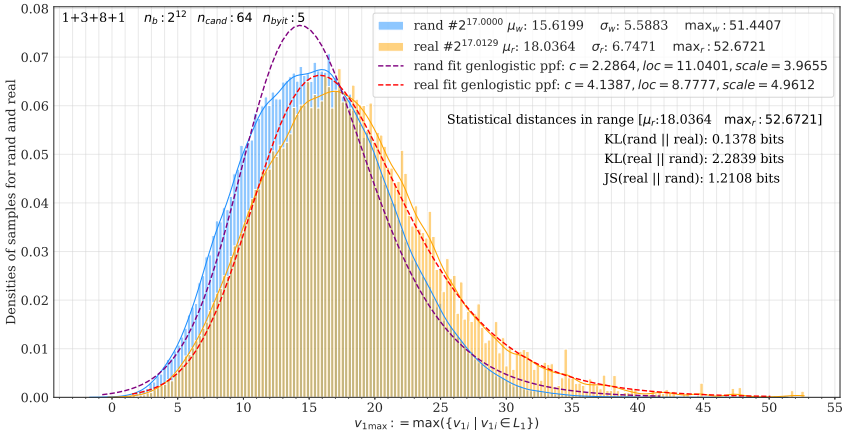


(b) Percentage of samples passing various cutoffs in 11a

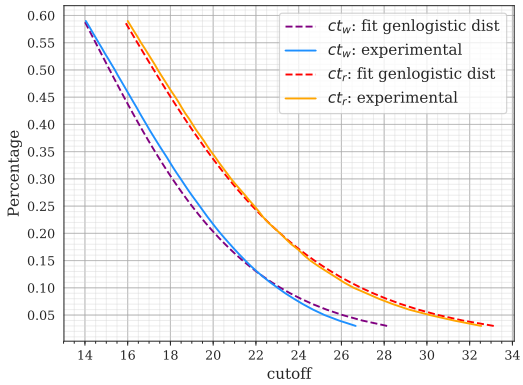


(c) Distribution of combined responses using correct ciphertext structures when the corresponding recommended subkey has Hamming distance hw with the real subkey

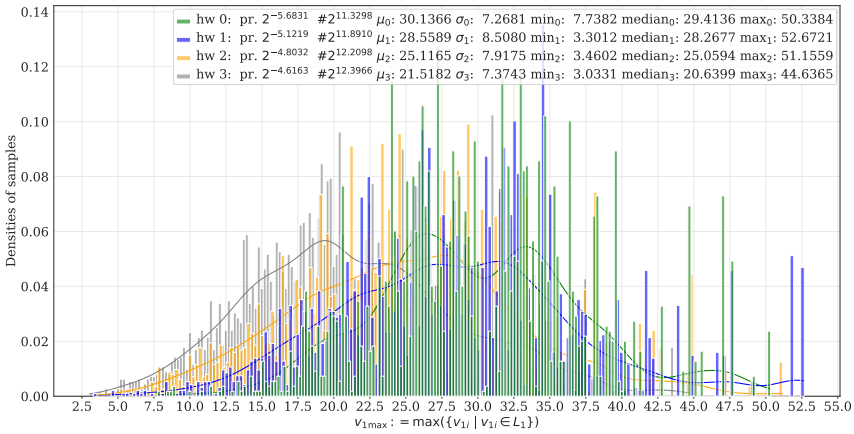
Fig. 11: Distribution of the combined responses on outputs of decrypting one round using recommended subkeys from correct ciphertext structures (`real`) and from wrong ciphertext structures (`rand`)



(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 $\mathcal{A}^{\text{SPECK}13R}$

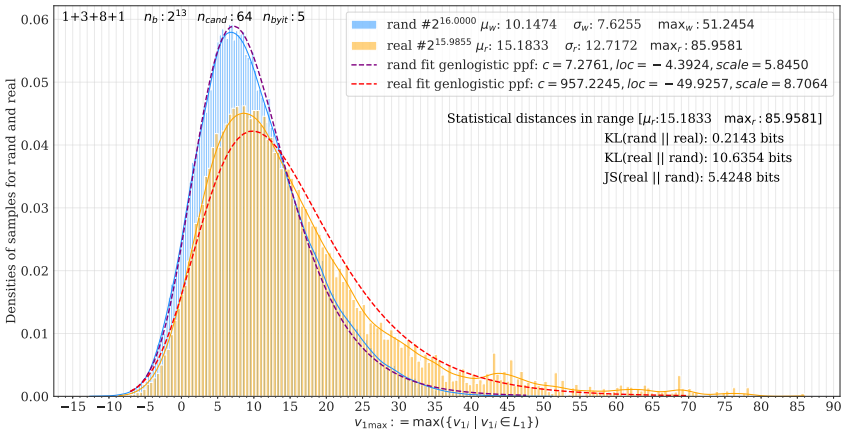


(b) Percentage of samples passing various cutoffs in 12a

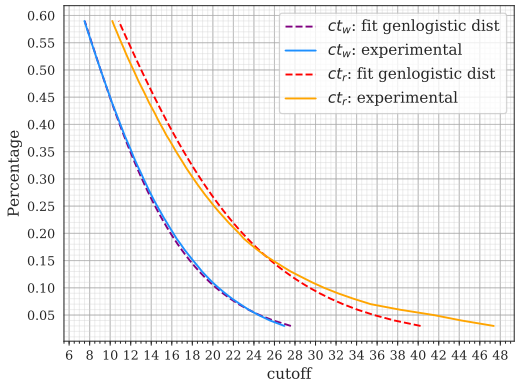


(c) Distribution of combined responses using correct ciphertext structures when the corresponding recommended subkey has Hamming distance hw with the real subkey

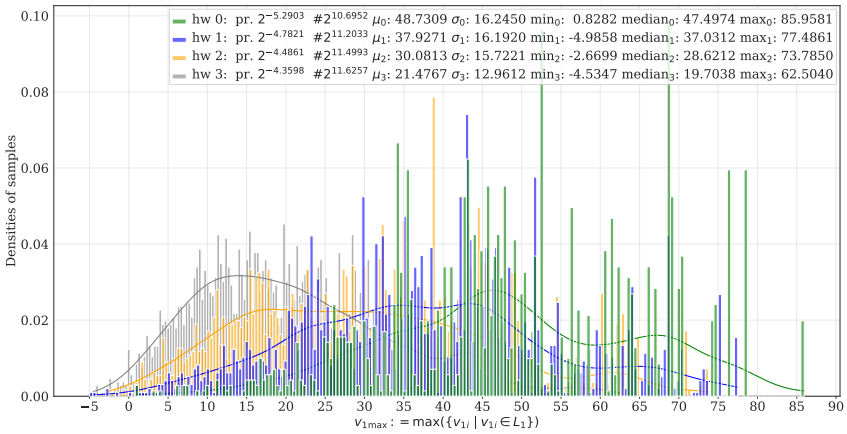
Fig. 12: Distribution of the combined responses on outputs of decrypting one round using recommonded subkeys from correct ciphertext structures (`real`) and from wrong ciphertext structures (`rand`)



(a) Sampling 2^{16} correct/wrong ciphertext structures to study the distributions $\mathcal{D}_r^{v_{1\max}}$ and $\mathcal{D}_w^{v_{1\max}}$ involved in attack $\mathcal{A}^{\text{SPECK}13R}$

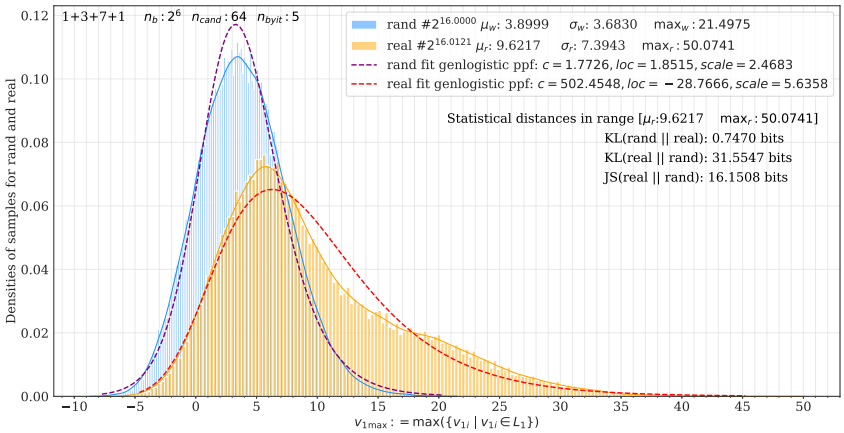


(b) Percentage of samples passing various cutoffs in 13a

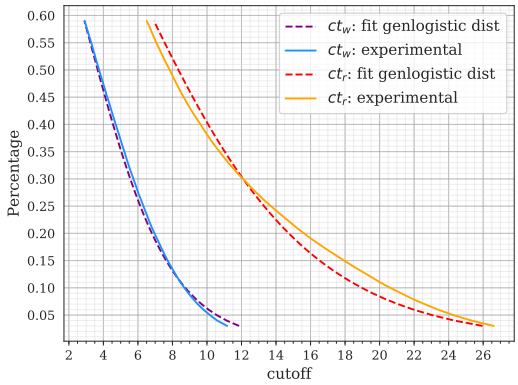


(c) Distribution of combined responses using correct ciphertext structures when the corresponding recommended subkey has Hamming distance hw with the real subkey

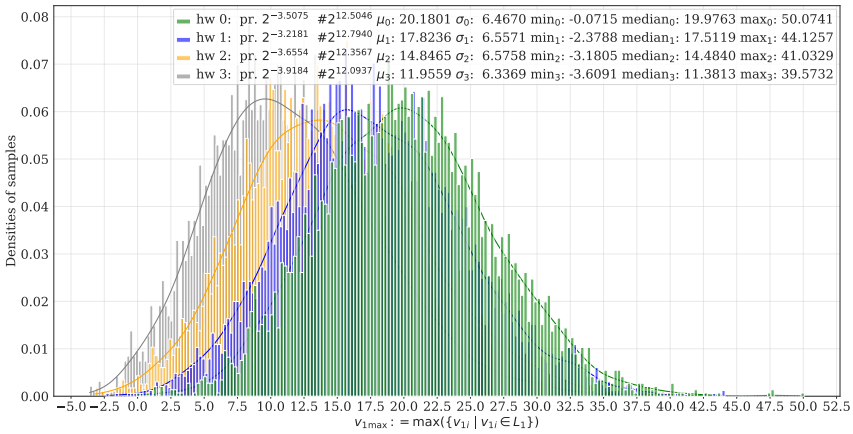
Fig. 13: Distribution of the combined responses on outputs of decrypting one round using recommended subkeys from correct ciphertext structures (`real`) and from wrong ciphertext structures (`rand`)



(a) Sampling 2^{16} correct/wrong ciphertext structures to study the distributions $\mathcal{D}_r^{v_{1\max}}$ and $\mathcal{D}_w^{v_{1\max}}$ involved in attack $\mathcal{A}^{\text{SPECK}12R}$



(b) Percentage of samples passing various cutoffs in 14a



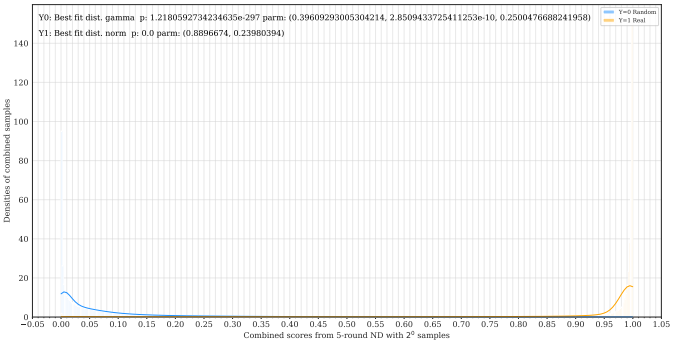
(c) Distribution of combined responses using correct ciphertext structures when the corresponding recommended subkey has Hamming distance hw with the real subkey

Fig. 14: 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`)

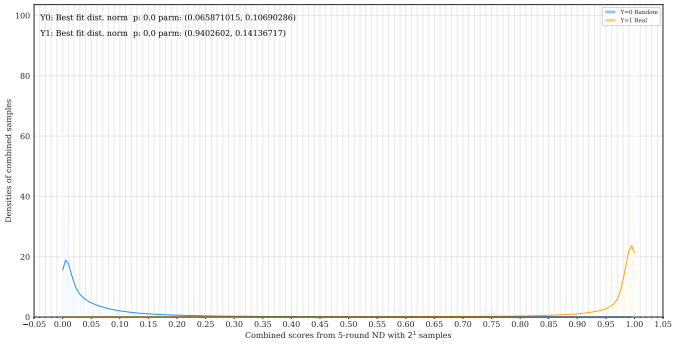
B Distributions of the Combined-response with Various Number of Blocks

To investigate how many samples from the same distribution should be combined to achieve a good combine-response distinguisher, that is how many neutral bits are necessary, the distributions of the combine-responses were experimentally investigated. The resulted distributions are illustrated using histogram plots. Besides, parameters of the best fitting distributions among {`Normal, Chi-squared, Generalized logistic, Logistic, Gamma`} are provided.

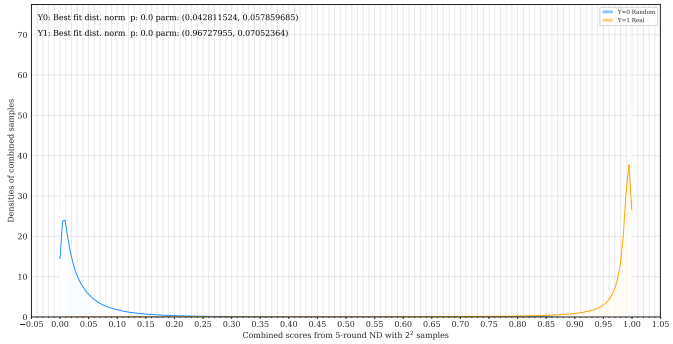
From Figures 15, 16, 17, 18, for Gohr's lightweight 5-, 6-, 7-, 8-round neural distinguishers (with accuracy listed in Table 11), the distributions corresponding to wrong and correct ciphertext structures can be separated when combining 2^0, $2^3 \sim 2^4$, $2^6 \sim 2^7$, and $2^{12} \sim 2^{13}$ samples, respectively. Thus, for using these distinguishers to do key-recovery and when there are only several correct ciphertext structures could occur, one should exploit 0, $3 \sim 4$, $6 \sim 7$, and $12 \sim 13$ neutral bits.



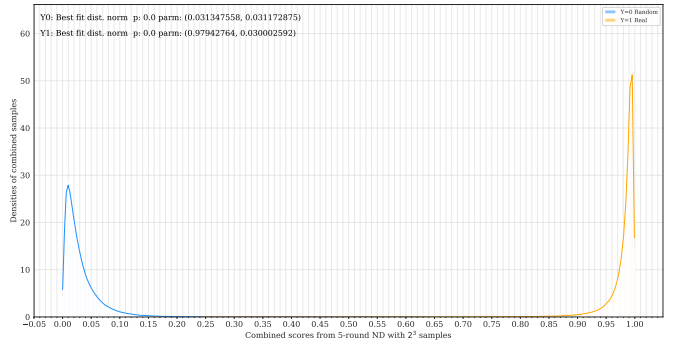
(a) Combining responses of 2^0 samples from $\mathcal{ND}^{\text{SPECK}}5R$



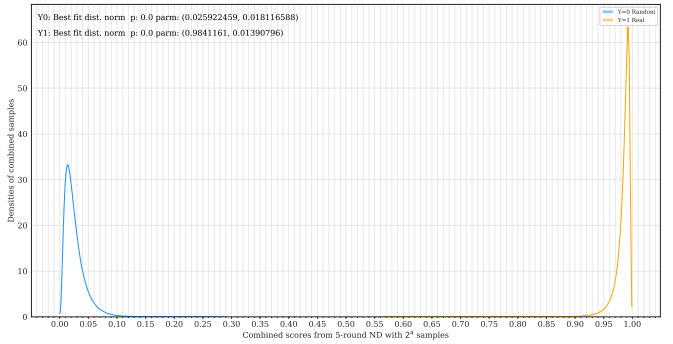
(b) Combining responses of 2^1 samples from $\mathcal{ND}^{\text{SPECK}}5R$



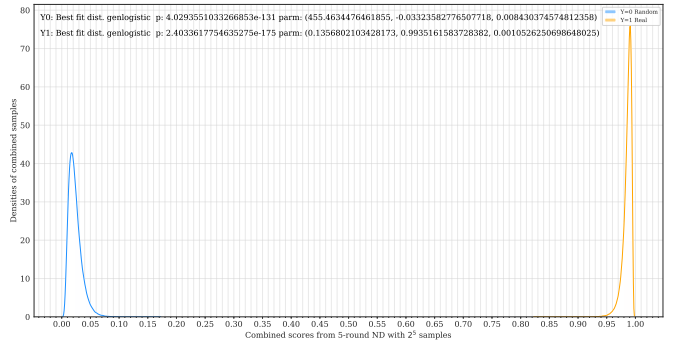
(c) Combining responses of 2^2 samples from $\mathcal{ND}^{\text{SPECK}}5R$



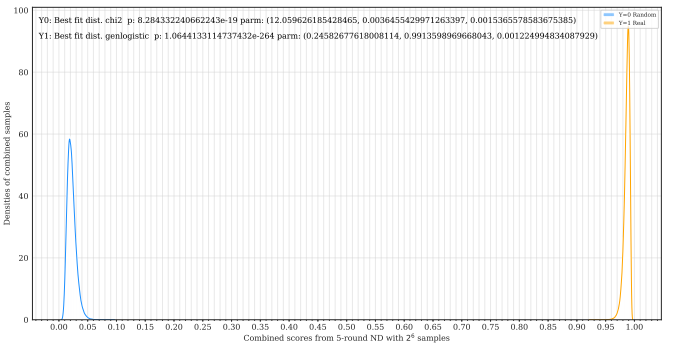
(d) Combining responses of 2^3 samples from $\mathcal{ND}^{\text{SPECK}}5R$



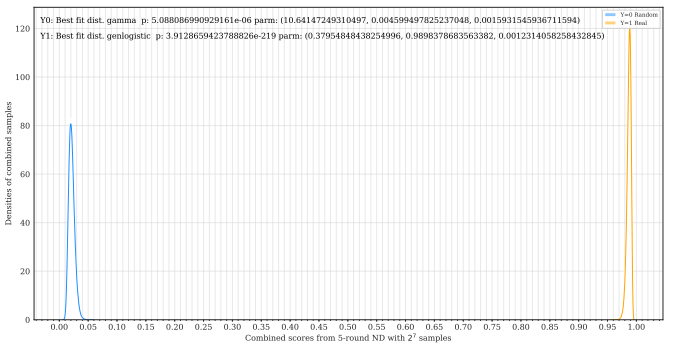
(e) Combining responses of 2^4 samples from $\mathcal{ND}^{\text{SPECK}}5R$



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

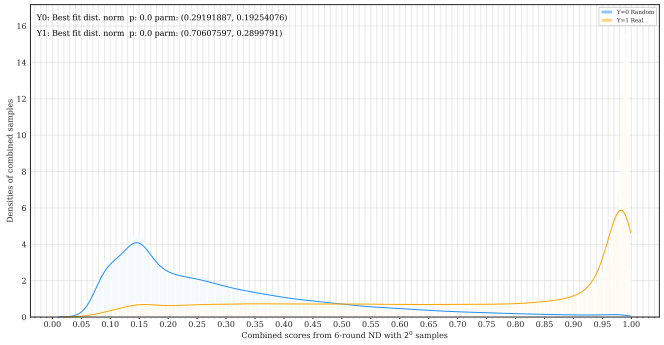


(g) Combining responses of 2^6 samples from $\mathcal{ND}^{\text{SPECK}}5R$

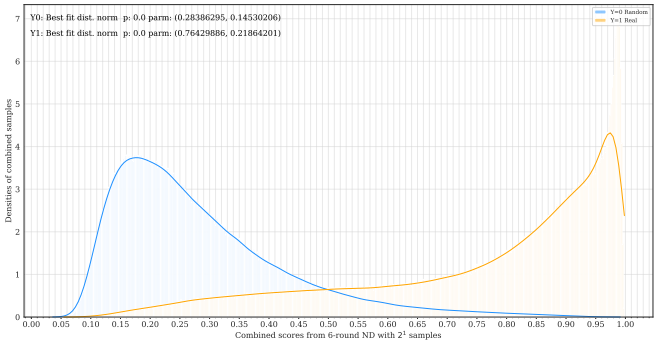


(h) Combining responses of 2^7 samples from $\mathcal{ND}^{\text{SPECK}}5R$

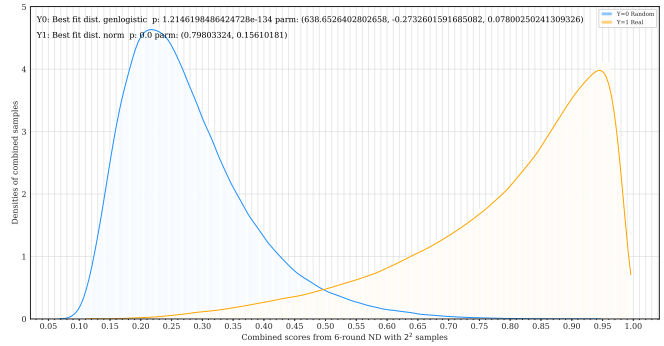
Fig. 15: The distribution of combined responses from 5-round ND (sampled with 2^{20} combined ciphertext-pairs)



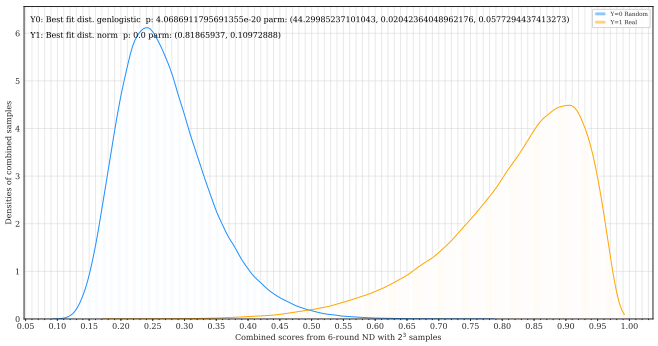
(a) Combining responses of 2^0 samples from $\mathcal{ND}^{\text{SPECK}}6R$



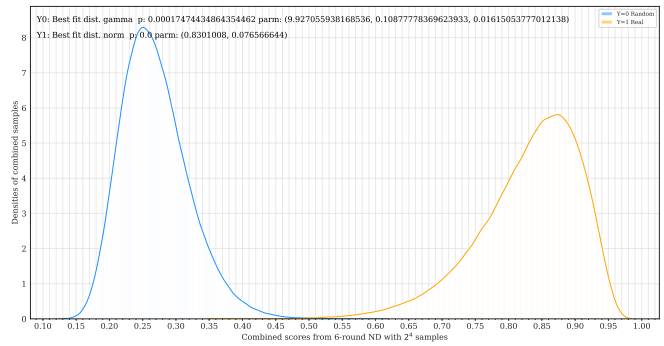
(b) Combining responses of 2^1 samples from $\mathcal{ND}^{\text{SPECK}}6R$



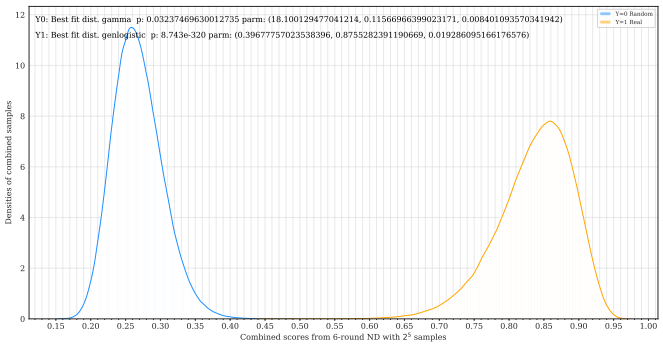
(c) Combining responses of 2^2 samples from $\mathcal{ND}^{\text{SPECK}}6R$



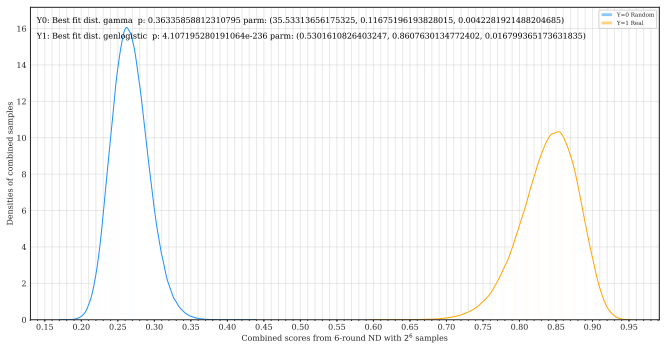
(d) Combining responses of 2^3 samples from $\mathcal{ND}^{\text{SPECK}}6R$



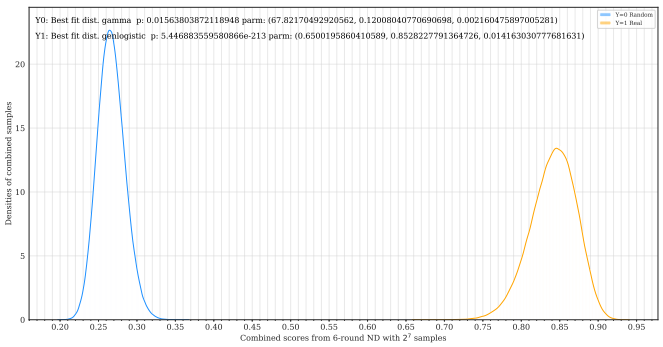
(e) Combining responses of 2^4 samples from $\mathcal{ND}^{\text{SPECK}}6R$



(f) Combining responses of 2^5 samples from $\mathcal{ND}^{\text{SPECK}}6R$

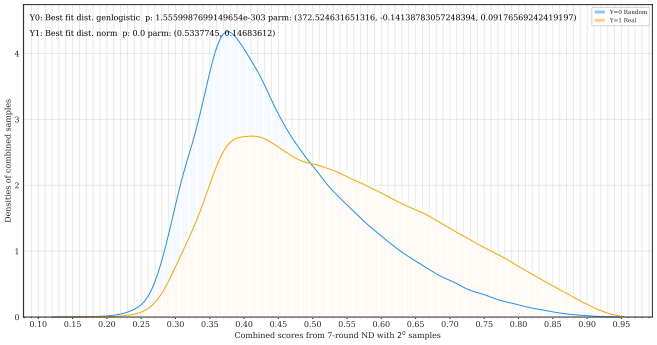


(g) Combining responses of 2^6 samples from $\mathcal{ND}^{\text{SPECK}}6R$

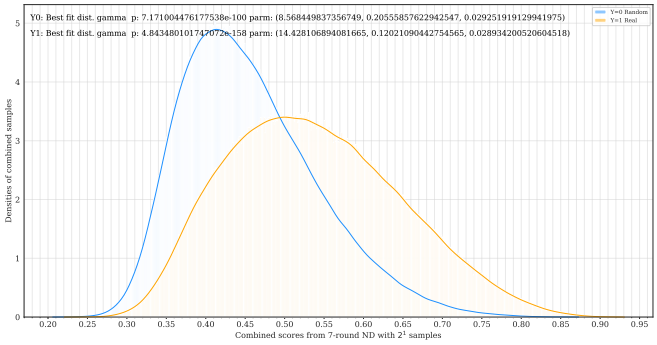


(h) Combining responses of 2^7 samples from $\mathcal{ND}^{\text{SPECK}}6R$

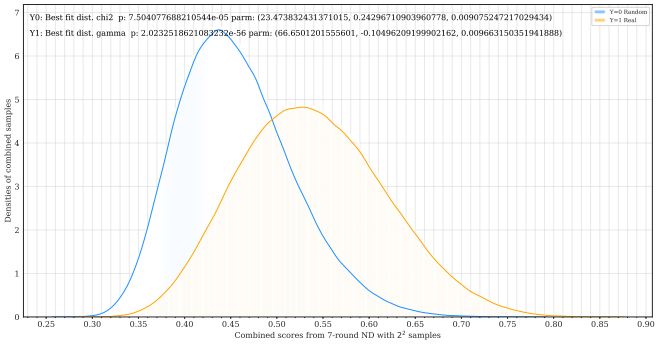
Fig. 16: The distribution of combined responses from 6-round ND (sampled with 2^{20} combined ciphertext-pairs)



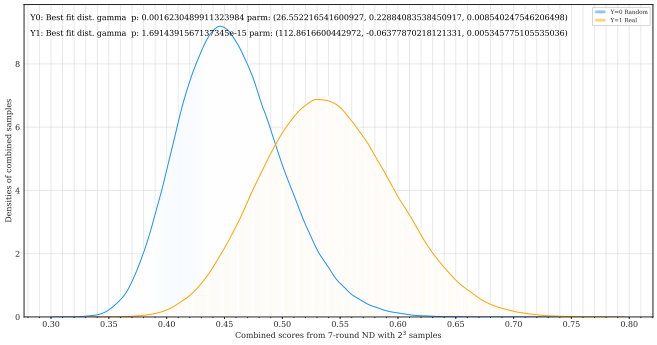
(a) Combining responses of 2^0 samples from $\mathcal{ND}^{\text{SPECK}}7R$



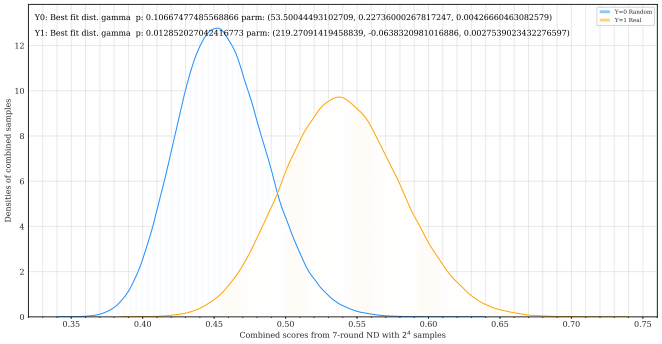
(b) Combining responses of 2^1 samples from $\mathcal{ND}^{\text{SPECK}}7R$



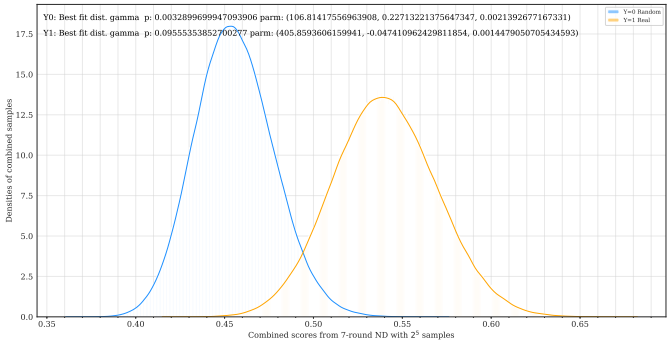
(c) Combining responses of 2^2 samples from $\mathcal{ND}^{\text{SPECK}}7R$



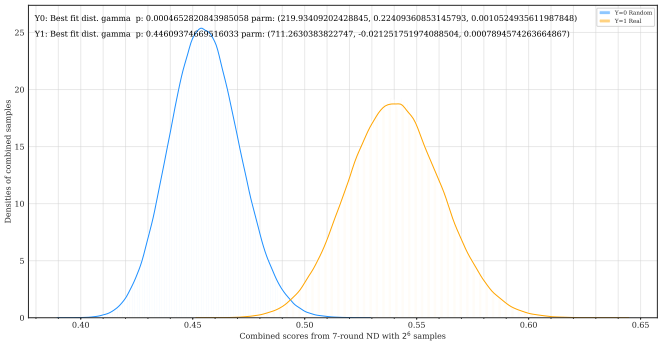
(d) Combining responses of 2^3 samples from $\mathcal{ND}^{\text{SPECK}}7R$



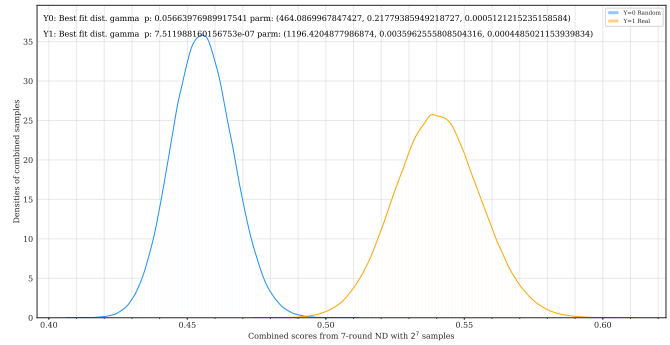
(e) Combining responses of 2^4 samples from $\mathcal{ND}^{\text{SPECK}}7R$



(f) Combining responses of 2^5 samples from $\mathcal{ND}^{\text{SPECK}}7R$

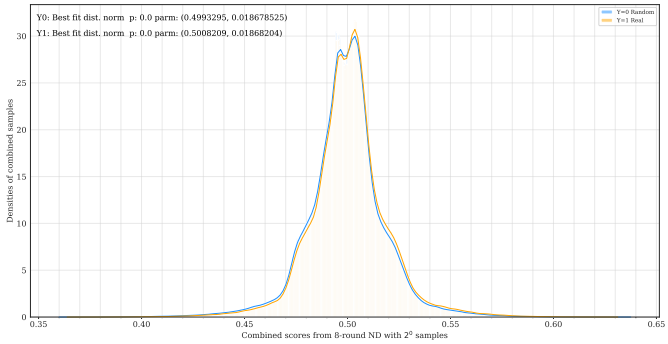


(g) Combining responses of 2^6 samples from $\mathcal{ND}^{\text{SPECK}}7R$

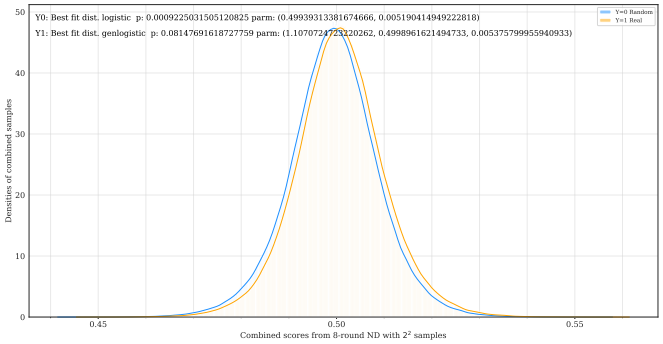


(h) Combining responses of 2^7 samples from $\mathcal{ND}^{\text{SPECK}}7R$

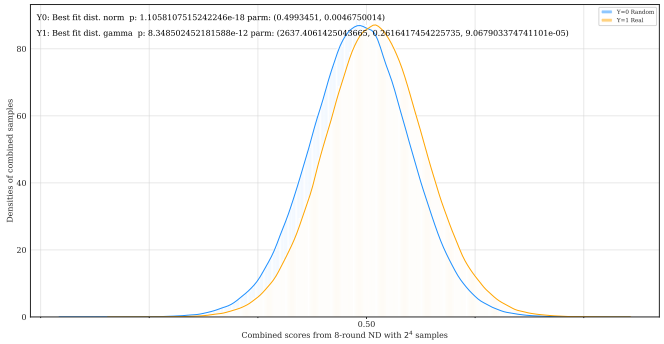
Fig. 17: The distribution of combined responses from 7-round ND (sampled with 2^{20} combined ciphertext-pairs)



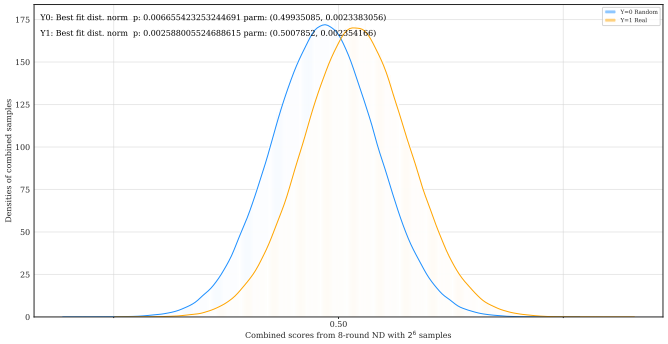
(a) Combining responses of 2^0 samples from $\mathcal{ND}^{\text{SPECK}8R}$



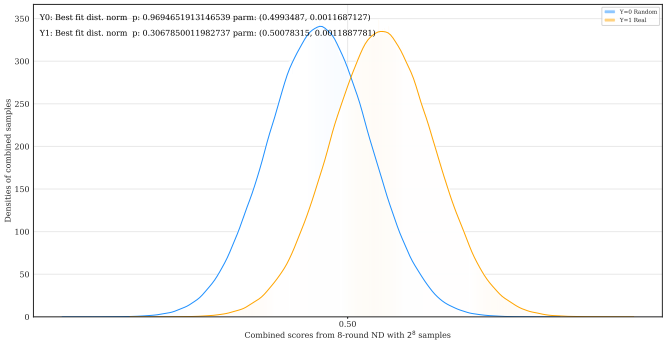
(b) Combining responses of 2^2 samples from $\mathcal{ND}^{\text{SPECK}8R}$



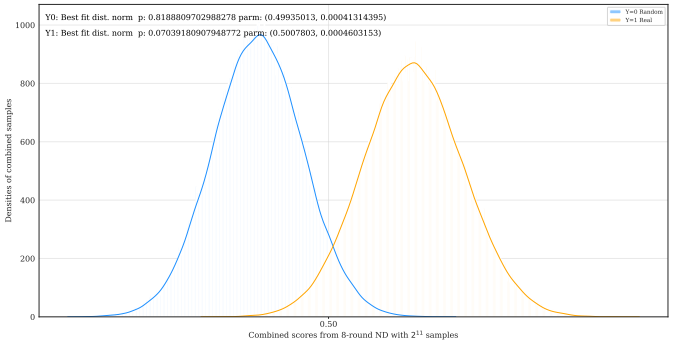
(c) Combining responses of 2^4 samples from $\mathcal{ND}^{\text{SPECK}8R}$



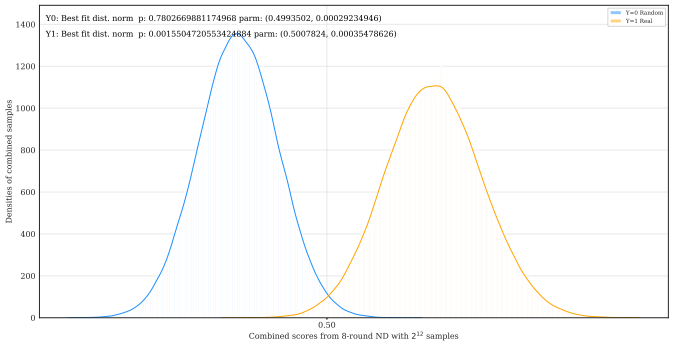
(d) Combining responses of 2^6 samples from $\mathcal{ND}^{\text{SPECK}8R}$



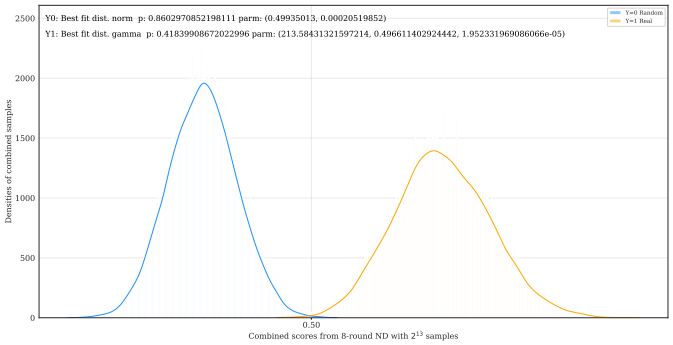
(e) Combining responses of 2^8 samples from $\mathcal{ND}^{\text{SPECK}8R}$



(f) Combining responses of 2^{11} samples from $\mathcal{ND}^{\text{SPECK}8R}$



(g) Combining responses of 2^{12} samples from $\mathcal{ND}^{\text{SPECK}8R}$



(h) Combining responses of 2^{13} samples from $\mathcal{ND}^{\text{SPECK}8R}$

Fig. 18: The distribution of combined responses from 8-round ND (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)

C 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 $(\texttt{0x8020}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$, $(\texttt{0x8060}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$, $(\texttt{0x8021}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$, and $(\texttt{0x8061}, \texttt{0x4101}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$. Note that this also happens to previous best differential attacks on SPECK32/64, including those covering the most rounds (14-round) in [9,12,25], 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 [9,12,25]. These constraints can be obtained using the existing tool ARXToolkit from [21].

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 [9] and the 14-round attacks in [12,25], the best 9-round trail was used. In this trail, there is 1 bit constraint on one subkey as shown in Table 8. In [25], 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 is 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 is previously estimated as 2^{-30} used in [9,12,25] 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 [25] should be about 2^{-32} for 2^{61} keys and 0 for other keys [8].

Additionally, we found that for one of the best differential trails with input difference $\texttt{0x0040/0000}$ 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 accuracy 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.

[8] Thus, the complexity of previous attacks are revised according to these estimations in Table 1.

Table 7: Generalized differences of the 3-round, 4-round, and 8-round differential trails used/involved in the attacks (obtained using tools in [21, 26])

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---	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-$^{c}_{0}$--x			z_0	x---------------
		k_0	----------------			k_0	----------------			k_0	----------------
		x_1	------x--------x			x_1	--------0-x----x			x_1	x---------------
1	0201 0604	y_1	-----<x------x--	1	0021 4001	y_1	-x-----!-------x	1	8000 8000	y_1	x---------------
		z_1	---->x--------$^{c}_{0}$--			z_1	-----=-----------			z_1	x-----=x--------
		k_1	---!------------			k_1	-----=-----------			k_1	------=---------
		x_2	---<x------------			x_2	-----=x--------x			x_2	x-----=x--------
2	1800 0010	y_2	---=--------x----	2	0601 0604	y_2	-----<x------x--	2	8300 8302	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	----$^{4}_{0}$----------
3	0040 0000	x_3	---------x------			x_3	---<x------------			x_3	x---$^{0}_{3}$xx$^{f}_{1}$--------
		y_3	----------------	3	1800 0010	y_3	---=--------x----	3	8e00 820a	y_3	x-----x-----x$^{1}_{0}$x-
						z_3	---------x------			z_3	x-----xx--x-xxx-
						k_3	----------------			k_3	----------------
				4	0040 0000	x_4	---------x------			x_4	x-----xx--$^{0}_{8}$-$^{0}_{3}$x$^{0}_{8}$-
						y_4	----------------	4	832e 8b04	y_4	x---x-<x------x$^{f}_{1}$-
										z_4	--x-x---------$^{2}_{0}$x-
										k_4	----------------
										x_5	--x-x---------x-
								5	2802 0410	y_5	-----x-----x----
										z_5	---------x------
										k_5	----------------
										x_6	---------x------
								6	0040 1000	y_6	---x------------
										z_6	x--x------------
										k_6	----------------
										x_7	x--x------$^{4}_{0}$-----
								7	9000 d000	y_7	xx-x------------
										z_7	-x-x---x--x-----
										k_7	----------------
								8	5120 1123	x_8	-x-x---x--x-----
										y_8	---x---x--x---xx

$Pr_a = 2^{-12}$

$Pr_r = \begin{cases} 2^{-11} & \text{for } 2^{63} \text{ keys} \\ 0 & \text{for others} \end{cases}$

$Pr_a = 2^{-17}$

$Pr_r = \begin{cases} 2^{-15} & \text{for } 2^{62} \text{ keys} \\ 0 & \text{for others} \end{cases}$

$Pr_a = 2^{-30}$

$Pr_r = \begin{cases} 2^{-26.58} & \text{for } 2^{60.58} \text{ keys} \\ 0 & \text{for others} \end{cases}$

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

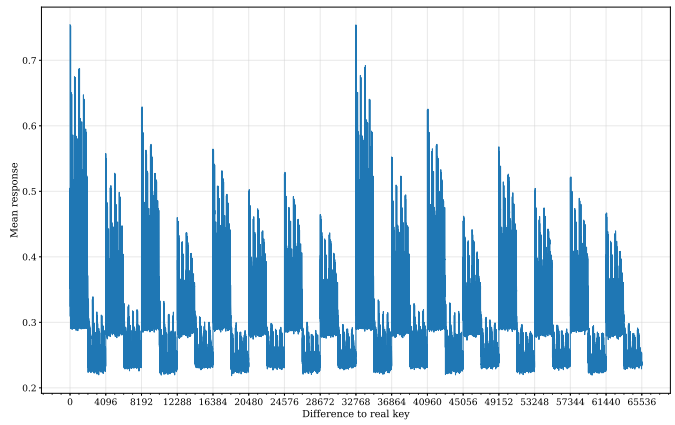
$^{c}_{0}$: uncommon constraint "c20000c3"
$^{c}_{1}$: uncommon constraint "c30000c2"
$^{c}_{2}$: uncommon constraint "c2000043"
$^{4}_{0}$: uncommon constraint "430000c2"
$^{f}_{0}$: uncommon constraint "ff0000f0"
$^{f}_{1}$: uncommon constraint "ff00000f"

$^{0}_{8}$: uncommon constraint "00824100"
$^{0}_{3}$: uncommon constraint "00381c00"
$^{0}_{4}$: uncommon constraint "00418200"
$^{1}_{0}$: uncommon constraint "1400003c"
$^{2}_{0}$: uncommon constraint "2800003c"
Constraints in red are on sub-keys.

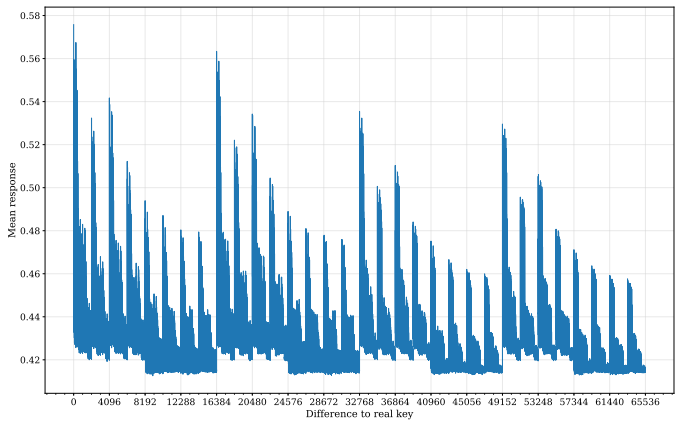
Table 8: Generalized differences of the 9-round and 10-round differential trails used in [9, 25] (obtained using tools in [21])

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	----------------
1	0000 a402	x_1	-------$^{f}_{1}$-------	1	8000 8100	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	----------------
2	a402 3408	x_2	$^{0}_{4}$-x--x--------x-	2	8000 8402	x_2	x------$^{f}_{1}$------!-
		y_2	--<x-x------x---			y_2	x----x!-------x-
		z_2	-$^{0}_{4}$-x-$^{c}_{2}$--<x------			z_2	x--->x-x------x-
		k_2	----------------			k_2	----=-----------
3	50c0 80e0	x_3	-x-x----xx------	3	8d02 9d08	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	-!----------------
4	0181 0203	x_4	-------xx------x	4	6002 1420	x_4	->x----------x-
		y_4	------x-------xx			y_4	-!-x-x----x-----
		z_4	------------>x01			z_4	---x-----xx-----
		k_4	------------!---			k_4	----------------
5	000c 0800	x_5	------------<x--	5	1060 40e0	x_5	-!-x-----xx-----
		y_5	----x-----------			y_5	-x------x>x-----
		z_5	--x---------------			z_5	------><x-------
		k_5	----------------			k_5	-------!--------
6	2000 0000	x_6	--x---------------	6	0380 0001	x_6	------x>x-------
		y_6	----------------			y_6	-------=-----=-x
		z_6	---------x------			z_6	------------x-0
		k_6	----------------			k_6	----------------
7	0040 0040	x_7	---------x------	7	0004 0000	x_7	------------x--
		y_7	---------x------			y_7	----------------
		z_7	x--------x------			z_7	----x-----------
		k_7	----------------			k_7	----------------
8	8040 8140	x_8	x--------x------	8	0800 0800	x_8	----x-----------
		y_8	x------x-x------			y_8	----x-----------
		z_8	---------x------			z_8	----x------x----
		k_8	----------------			k_8	----------------
9	0040 0542	x_9	---------x------	9	0810 2810	x_9	----x------x----
		y_9	-----x-x-x----x-			y_9	--x-x------x----
						z_9	----x-----------
						k_9	----------------
						x_{10}	----x-----------
						y_{10}	x-x-x----x------
$Pr_a = 2^{-30}$ $Pr_r = \begin{cases} 2^{-29} & \text{for } 2^{63} \text{ keys} \\ 0 & \text{for others} \end{cases}$				$Pr_a = 2^{-35}$ $Pr_r = \begin{cases} 2^{-32} & \text{for } 2^{61} \text{ keys} \\ 0 & \text{for others} \end{cases}$			

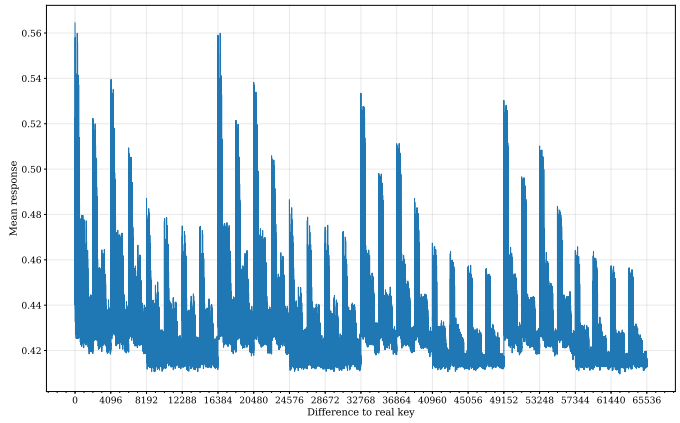
Symbols used are same as in Table 7



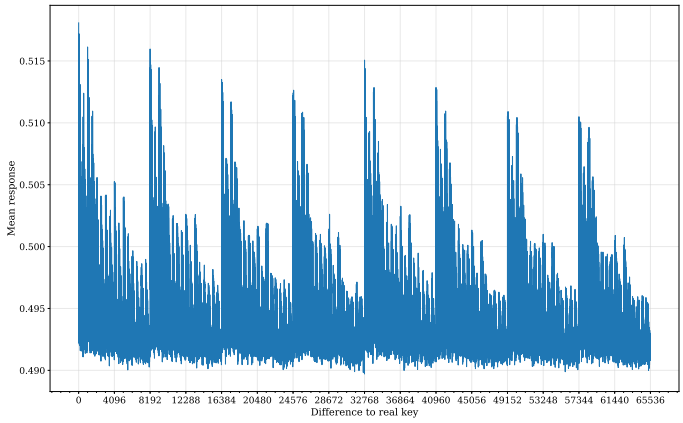
(a) $\mathcal{ND}^{\text{SIMON}8R}_{\mathbf{VV}}$: directly trained with data of the form $((x, y, x', y'))$



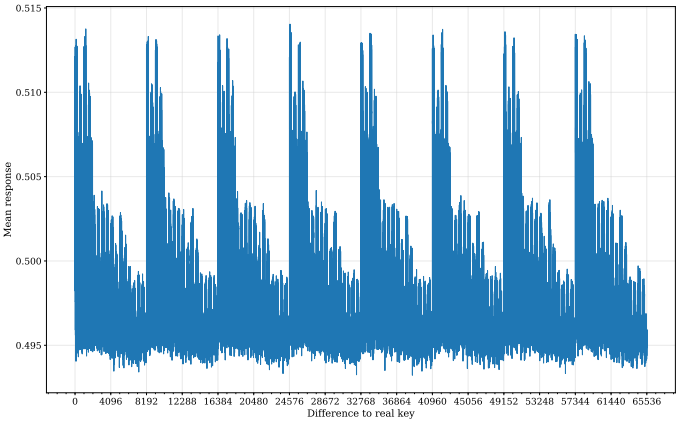
(b) $\mathcal{ND}^{\text{SIMON}8R}_{\mathbf{VD}}$: directly trained with data of the form $((x, x', y \oplus y'))$



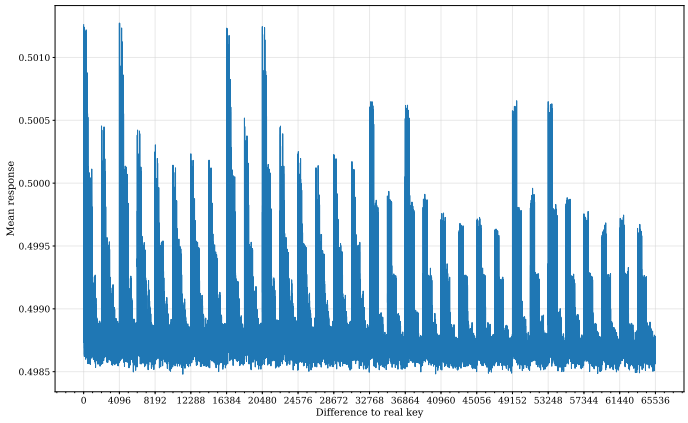
(c) $\mathcal{ND}^{\text{SIMON}9R}_{\mathbf{VV}}$: directly trained with data of the form $((x, y, x', y'))$



(d) $\mathcal{ND}^{\text{SIMON}9R}_{\mathbf{VD}}$: directly trained with data of the form $((x, x', y \oplus y'))$



(e) $\mathcal{ND}^{\text{SIMON}10R}_{\mathbf{VV}}$: trained using $\mathcal{ND}^{\text{SIMON}9R}_{\mathbf{VV}}$ and KeyAveraging algorithm



(f) $\mathcal{ND}^{\text{SIMON}11R}_{\mathbf{VV}}$: trained using $\mathcal{ND}^{\text{SIMON}9R}_{\mathbf{VV}}$ and $\mathcal{DD}^{\text{SIMON}8R}_{(\mathbf{0440},\mathbf{0100})}$ 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))

D Key-recovery Attacks on Round-Reduced Simon32/64

D.1 Wrong Key Response Profile for the Neural Distinguishers on Simon32/64

D.2 The First 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.

The attack presented in this section, named as $\mathcal{A}_I^{\text{Simon}_{16R}}$, combines the longest but weak neural distinguisher with a differential that has many SNBS.

In Appendix D.5, another attack, named as $\mathcal{A}_{II}^{\text{Simon}_{16R}}$, is presented. It combines relatively strong neural distinguishers with a differential that is one round longer but has fewer neutral bits. $\mathcal{A}_I^{\text{Simon}_{16R}}$ is superior to $\mathcal{A}_{II}^{\text{Simon}_{16R}}$, but $\mathcal{A}_{II}^{\text{Simon}_{16R}}$ achieves better data complexity than previous attacks in literature. It succeeded by using a few of the identified CSNBS of the longer classical differential. .

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

Similar to attacks on Speck32/64, to obtain decent scores from the responses of the neural distinguishers, combined response from the neural distinguisher 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 classical differential are exploited.

D.3 Finding Neutral Bits for the Classical Differentials

Finding SNBS for 3-round Differential. For the 3-round differential to be prepended to the neural distinguishers, one can obtain all neutral bits and SNBS (simultaneously complementing up to 4 bits) using the following algebraic method.

Given the input and output differences $(\texttt{0x0440}, \texttt{0x1000})$ and $(\texttt{0x0000}, \texttt{0x0040})$, 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 Gröbner basis, which can be done using the PolyBoRi library integrated in SageMath [10]. In the obtained Gröbner basis, those disappeared variables correspond to the single-bit neutral bits of the differential. To find SNBS, 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 resulted Gröbner basis, replace this set of variables with their complements simultaneously; if the Gröbner basis does not change, the set of variables corresponds to a SNBS.

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

all 3-SNBS and 4-SNBS are formed by combinations of the 9 single neutral bits and 2 2-SNBS. Thus, there are 11 independent neutral bits and SNBS in total.

From the resulted Gröbner basis (also observed by experiments), for an input pair $((x, y), (x', y'))$ to conform the 3-round differential $(\texttt{0x0440}, \texttt{0x1000}) \rightarrow (\texttt{0x0000}, \texttt{0x0040})$, one has

$$\begin{cases} x[1] = x'[1] = 0, \\ x[3] = x'[3] = 0. \end{cases} \quad (2)$$

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

The components of $\mathcal{A}_I^{\text{SIMON}_{16R}}$ are as follows (refer to Fig. 21).

1. A 3-round classical differential $(\texttt{0x0440}, \texttt{0x1000}) \rightarrow (\texttt{0x0000}, \texttt{0x0040})$ (refer to the rounds colored in blue in Fig. 21), and a set of its 11 NBS {[2], [3], [4], [6], [8], [9], [10], [18], [22], [0, 24],[12, 26] };
2. A 11-round neural distinguisher $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{11R}}$ trained using the staged approach under difference $(\texttt{0x0000}, \texttt{0x0040})$, and its wrong key response profiles $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{11R}}.\mu$ and $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{11R}}.\sigma$.
3. A 9-round neural distinguisher $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{9R}}$ trained under difference $(\texttt{0x0000}, \texttt{0x0040})$ and fed with data of type $(x, x', y \oplus y')$, and its wrong key response profiles $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{9R}}.\mu$ and $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{9R}}.\sigma$.

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 neural distinguishers $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{9R}}$ 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 $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{9R}}$.

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 $\mathcal{ND}_{\mathbf{VV}}^{\text{SIMON}_{11R}}$ is 0.5173, and that of $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{9R}}$ is 0.5629.

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

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 using 8 threads on the same GPU server testing $\mathcal{A}^{\text{SPECK}_{13R}}$. In total 99 trials are run[9]. Within the 99 trials, all trials have correct ciphertext pairs and all called the neural distinguishers. There are 49 success trials, for which the returned last two subkeys have a Hamming

[9] They are designed to run 20 trials each; but because of the walltime exceeded the required, they were killed; In total, 99 trials had completed, 7 trials terminated without finish (two almost finished and succeed).

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.

D.5 The Second Attack on 16-round SIMON32/64

Details of Find CSNBS for 4-round Differential. For the 4-round differential to be prepended to the neural distinguisher, *i.e.*, $(\texttt{0x1000}, \texttt{0x4440}) \rightarrow (\texttt{0x0000}, \texttt{0x0040})$, NB and SNBS are scarce; there are only 2 single NB and 2 2-SNBS.

However, when fixing values of some input bits, more bits become neutral. Given the 9 single NB and the 2 2-SNBS of the 3-round differential $(\texttt{0x0440}, \texttt{0x1000}) \rightarrow (\texttt{0x0000}, \texttt{0x0040})$, CSNBS for the 4-round differential can be constructed as follows. Denote the input bits to the 4-round SIMON32/64 by $\mathtt{x}_{15}$, $\mathtt{x}_{14}$, ..., $\mathtt{x}_0$, $\mathtt{y}_{15}$, $\mathtt{y}_{14}$, ..., $\mathtt{y}_0$. To deduce the conditions for these input bits to be neutral for the 4-round differential, one considers one round transformation as depicted in Fig. 20.

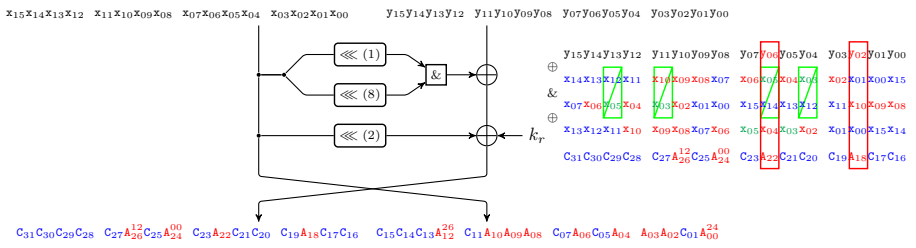


Fig. 20: Deduce conditional simultaneous-neutral bits/bit-sets for 4-round differential of SIMON32/64

Indicate the neutrality of bit i by $\mathtt{A}_i$ as neutral and $\mathtt{C}_i$ as non-neutral for the 3-round differential. Besides, indicate the neutrality of bit i under the condition that simultaneously changing bits j by $\mathtt{A}_i^j$ as neutral. Corresponding to the 9 single NB [2], [3], [4], [6], [8], [9], [10], [18], [22] and two 2-SNBS $[0, 24]$, $[12, 26]$ of the 3-round differential, we indicate the neutrality (with respect to the 3-round differential) of the output of the first round of the 4-round as follows

$$\begin{array}{l}\mathtt{C}_{31}\mathtt{C}_{30}\mathtt{C}_{29}\mathtt{C}_{28}\ \mathtt{C}_{27}\mathtt{A}_{26}^{12}\mathtt{C}_{25}\mathtt{A}_{24}^{00}\ \mathtt{C}_{23}\mathtt{A}_{22}\mathtt{C}_{21}\mathtt{C}_{20}\ \mathtt{C}_{19}\mathtt{A}_{18}\mathtt{C}_{17}\mathtt{C}_{16}\\ \mathtt{C}_{15}\mathtt{C}_{14}\mathtt{C}_{13}\mathtt{A}_{12}^{26}\ \mathtt{C}_{11}\mathtt{A}_{10}\mathtt{A}_{09}\mathtt{A}_{08}\ \mathtt{C}_{07}\mathtt{A}_{06}\mathtt{C}_{05}\mathtt{A}_{04}\ \mathtt{A}_{03}\mathtt{A}_{02}\mathtt{C}_{01}\mathtt{A}_{00}^{24}\end{array}$$

Note that, for an input pair $((x, y), (x', y'))$ to conform the 4-round differential $(\texttt{0x1000}, \texttt{0x4440}) \rightarrow (\texttt{0x0000}, \texttt{0x0040})$, one have that $\begin{cases}\mathtt{x}_5 = \mathtt{x}'_5 = 0,\\ \mathtt{x}_3 = \mathtt{x}'_3 = 0.\end{cases}$

Firstly, one can directly deduce the 2 single NB and the 2 2-SNBS for the 4-round differential from that of the 3-round differential as follows.

– From the neutrality of A_{22} and A_{18} for the 3-round differential, and because $\begin{cases} \boxed{y_{06}} \oplus x_{05}x_{14} \oplus x_{14} = A_{22}, \\ \boxed{y_{02}} \oplus x_{01}x_{10} \oplus x_{00} = A_{18}, \end{cases}$ we have that $\boxed{y_{06}}$ and $\boxed{y_{02}}$ (*i.e.*, [2] and [6]) are neutral for the 4-round differential.
– From the neutrality of A_{10} and A_{18} for the 3-round differential, let us consider the neutrality of x_{10}. The variable x_{10} is involved in the following equations in the one round transformation $\begin{cases} \boxed{x_{10}} = A_{10}, \\ y_{02} \oplus x_{01}\boxed{x_{10}} \oplus x_{00} = A_{18}, \\ \boxed{y_{12}} \oplus x_{11}x_{04} \oplus \boxed{x_{10}} = C_{28}, \\ y_{11} \oplus \boxed{x_{10}}x_{03} \oplus x_{09} = C_{27}. \end{cases}$ Because A_{10} and A_{18} are neutral for the appended 3-round differential, the first two equations do not impose conditions for $\boxed{x_{10}}$ to be neutral for the 4-round differential. In the third equation, although C_{28} is non-neutral for the appended 3-round differential, there is a free variable $\boxed{y_{12}}$; thus, complementing $\boxed{x_{10}}$ and $\boxed{y_{12}}$ simultaneously does not violate the non-neutrality of C_{28}. In the fourth equation, because x_{03} must be zero for all conforming pairs of the 4-round differential, complementing $\boxed{x_{10}}$ does not violate the non-neutrality of C_{27}. Thus, we have that $\boxed{x_{10}}$ and $\boxed{y_{12}}$ (*i.e.*, [12, 26]) form a SNBS for the 4-round differential.
– From the simultaneous-neutrality of A_{12}^{26} and A_{26}^{12} for the 3-round differential, let us consider the neutrality of x_{12} and y_{10}. Because x_{12} and y_{10} are involved in the following equations in the one round transformation $\begin{cases} \boxed{x_{12}} = A_{12}^{26}, \\ \boxed{y_{10}} \oplus x_{09}x_{02} \oplus x_{08} = A_{26}^{12}, \\ \boxed{y_{14}} \oplus x_{13}x_{06} \oplus \boxed{x_{12}} = C_{30}, \\ y_{04} \oplus x_{03}\boxed{x_{12}} \oplus x_{02} = C_{20}, \text{ where } x_{03} = 0, \end{cases}$ similar to the above analysis, we have that $\boxed{x_{12}}$, $\boxed{y_{10}}$, and $\boxed{y_{14}}$ (*i.e.*, [10, 14, 28]) form a SNBS for the 4-round differential.

Next, let consider the conditional neutrality of input bits for the 4-round differential.

– From the neutrality of A_{08} for the 3-round differential, let us consider the neutrality of x_{08}. The variable x_{08} is involved in the following equations in the one round transformation $\begin{cases} \boxed{x_{08}} = A_{08}, \\ \boxed{y_{10}} \oplus x_{09}x_{02} \oplus \boxed{x_{08}} = C_{26}, \\ \boxed{y_{09}} \oplus \boxed{x_{08}}x_{01} \oplus x_{07} = C_{25}, \\ \boxed{y_{00}} \oplus x_{15}\boxed{x_{08}} \oplus x_{14} = C_{16}. \end{cases}$ In the second equation, to not violate the non-neutrality of C_{26}, one should complement $\boxed{x_{08}}$ and $\boxed{y_{10}}$ simultaneously. In the third equation, to not violate the non-neutrality of C_{25}, when $x_{01} = 0$, one can freely complement $\boxed{x_{08}}$; when $x_{01} = 1$, one should complement $\boxed{x_{08}}$ and $\boxed{y_{09}}$ simultaneously. In the fourth

equation, to not violate the non-neutrality of $\mathtt{C}_{16}$, when $\mathtt{x}_{15} = 0$, one can freely complement $\boxed{\mathtt{x}_{08}}$; when $\mathtt{x}_{15} = 1$, one should complement $\boxed{\mathtt{x}_{08}}$ and $\boxed{\mathtt{y}_{00}}$ simultaneously. Therefore, we have a CSNBS for the 4-round differential as follows

$$\begin{cases} [\mathtt{x}_{08}, \mathtt{y}_{10}], \ \textit{i.e.}, \ [24, 10] & (\mathtt{x}_{01}, \mathtt{x}_{15}) = (0, 0); \\ [\mathtt{x}_{08}, \mathtt{y}_{10}, \mathtt{y}_{09}], \ \textit{i.e.}, \ [24, 10, 9] & (\mathtt{x}_{01}, \mathtt{x}_{15}) = (1, 0); \\ [\mathtt{x}_{08}, \mathtt{y}_{10}, \mathtt{y}_{00}], \ \textit{i.e.}, \ [24, 10, 0] & (\mathtt{x}_{01}, \mathtt{x}_{15}) = (0, 1); \\ [\mathtt{x}_{08}, \mathtt{y}_{10}, \mathtt{y}_{09}, \mathtt{y}_{00}], \ \textit{i.e.}, \ [24, 10, 9, 0] & (\mathtt{x}_{01}, \mathtt{x}_{15}) = (1, 1). \end{cases}$$

– From the neutrality of $\mathtt{A}_{06}$ for the 3-round differential, let us consider the neutrality of $\mathtt{x}_{06}$. The variable $\mathtt{x}_{06}$ is involved in the following equations in the one round transformation

$$\begin{cases} \boxed{\mathtt{x}_{06}} = \mathtt{A}_{06}, \\ \mathtt{y}_{08} \oplus \mathtt{x}_{07}\mathtt{x}_{00} \oplus \boxed{\mathtt{x}_{06}} = \mathtt{C}_{24}, \\ \mathtt{y}_{07} \oplus \boxed{\mathtt{x}_{06}}\mathtt{x}_{15} \oplus \mathtt{x}_{05} = \mathtt{C}_{23}, \\ \mathtt{y}_{14} \oplus \mathtt{x}_{13}\boxed{\mathtt{x}_{06}} \oplus \mathtt{x}_{12} = \mathtt{C}_{30}. \end{cases}$$

Similar to the above analysis, we have a CSNBS for the 4-round differential as follows

$$\begin{cases} [\mathtt{x}_{06}, \mathtt{y}_{08}], \ \textit{i.e.}, \ [22, 8] & (\mathtt{x}_{15}, \mathtt{x}_{13}) = (0, 0); \\ [\mathtt{x}_{06}, \mathtt{y}_{08}, \mathtt{y}_{07}], \ \textit{i.e.}, \ [22, 8, 7] & (\mathtt{x}_{15}, \mathtt{x}_{13}) = (1, 0); \\ [\mathtt{x}_{06}, \mathtt{y}_{08}, \mathtt{y}_{14}], \ \textit{i.e.}, \ [22, 8, 14] & (\mathtt{x}_{15}, \mathtt{x}_{13}) = (0, 1); \\ [\mathtt{x}_{06}, \mathtt{y}_{08}, \mathtt{y}_{07}, \mathtt{y}_{14}], \ \textit{i.e.}, \ [22, 8, 7, 14] & (\mathtt{x}_{15}, \mathtt{x}_{13}) = (1, 1). \end{cases}$$

– From the neutrality of $\mathtt{A}_{04}$ for the 3-round differential, let us consider the neutrality of $\mathtt{x}_{04}$. The variable $\mathtt{x}_{04}$ is involved in the following equations in the one round transformation

$$\begin{cases} \boxed{\mathtt{x}_{04}} = \mathtt{A}_{04}, \\ \mathtt{y}_{06} \oplus \mathtt{x}_{05}\mathtt{x}_{14} \oplus \boxed{\mathtt{x}_{04}} = \mathtt{A}_{22}, \\ \mathtt{y}_{05} \oplus \boxed{\mathtt{x}_{04}}\mathtt{x}_{13} \oplus \mathtt{x}_{03} = \mathtt{C}_{21}, \\ \mathtt{y}_{12} \oplus \mathtt{x}_{11}\boxed{\mathtt{x}_{04}} \oplus \mathtt{x}_{10} = \mathtt{C}_{28}. \end{cases}$$

Similar to the above analysis, we have a CSNBS for the 4-round differential as follows

$$\begin{cases} [\mathtt{x}_{04}], \ \textit{i.e.}, \ [20] & (\mathtt{x}_{13}, \mathtt{x}_{11}) = (0, 0); \\ [\mathtt{x}_{04}, \mathtt{y}_{05}], \ \textit{i.e.}, \ [20, 5] & (\mathtt{x}_{13}, \mathtt{x}_{11}) = (1, 0); \\ [\mathtt{x}_{04}, \mathtt{y}_{12}], \ \textit{i.e.}, \ [20, 12] & (\mathtt{x}_{13}, \mathtt{x}_{11}) = (0, 1); \\ [\mathtt{x}_{04}, \mathtt{y}_{05}, \mathtt{y}_{12}], \ \textit{i.e.}, \ [20, 5, 12] & (\mathtt{x}_{13}, \mathtt{x}_{11}) = (1, 1). \end{cases}$$

– From the neutrality of $\mathtt{A}_{02}$ for the 3-round differential, let us consider the neutrality of $\mathtt{x}_{02}$. The variable $\mathtt{x}_{02}$ is involved in the following equations in the one round transformation

$$\begin{cases} \boxed{\mathtt{x}_{02}} = \mathtt{A}_{02}, \\ \mathtt{y}_{04} \oplus \mathtt{x}_{03}\mathtt{x}_{12} \oplus \boxed{\mathtt{x}_{02}} = \mathtt{C}_{20}, \\ \mathtt{y}_{03} \oplus \boxed{\mathtt{x}_{02}}\mathtt{x}_{11} \oplus \mathtt{x}_{01} = \mathtt{C}_{19}, \\ \mathtt{y}_{10} \oplus \mathtt{x}_{09}\boxed{\mathtt{x}_{02}} \oplus \mathtt{x}_{08} = \mathtt{C}_{26}. \end{cases}$$

Similar to

the above analysis, we have a CSNBS for the 4-round differential as follows

$$\begin{cases} [\mathtt{x}_{02}, \mathtt{y}_{04}], \ \textit{i.e.}, \ [18, 4] & (\mathtt{x}_{11}, \mathtt{x}_{09}) = (0, 0); \\ [\mathtt{x}_{02}, \mathtt{y}_{04}, \mathtt{y}_{03}], \ \textit{i.e.}, \ [18, 4, 3] & (\mathtt{x}_{11}, \mathtt{x}_{09}) = (1, 0); \\ [\mathtt{x}_{02}, \mathtt{y}_{04}, \mathtt{y}_{10}], \ \textit{i.e.}, \ [18, 4, 10] & (\mathtt{x}_{11}, \mathtt{x}_{09}) = (0, 1); \\ [\mathtt{x}_{02}, \mathtt{y}_{04}, \mathtt{y}_{03}, \mathtt{y}_{10}], \ \textit{i.e.}, \ [18, 4, 3, 10] & (\mathtt{x}_{11}, \mathtt{x}_{09}) = (1, 1). \end{cases}$$

– From the simultaneous neutrality of $\mathtt{A}_{00}^{24}$ and $\mathtt{A}_{24}^{00}$ for the 3-round differential, let us consider the neutrality of $\mathtt{x}_{00}$ and $\mathtt{y}_{08}$. The variable $\mathtt{x}_{00}$ and $\mathtt{y}_{08}$ are involved in the following equations in the one round transformation

$$\begin{cases} \boxed{\mathtt{x}_{00}} = \mathtt{A}_{00}^{24}, \\ \mathtt{y}_{02} \oplus \mathtt{x}_{01}\mathtt{x}_{10} \oplus \boxed{\mathtt{x}_{00}} = \mathtt{A}_{18}, \\ \boxed{\mathtt{y}_{01}} \oplus \boxed{\mathtt{x}_{00}}\mathtt{x}_{09} \oplus \mathtt{x}_{15} = \mathtt{C}_{17}, \\ \boxed{\mathtt{y}_{08}} \oplus \mathtt{x}_{07}\boxed{\mathtt{x}_{00}} \oplus \mathtt{x}_{06} = \mathtt{A}_{24}^{00}. \end{cases}$$

Similar to the above analysis, we have a CSNBS for the 4-round differential as follows

$$\begin{cases} [\mathtt{x}_{00}, \mathtt{y}_{08}], \ \textit{i.e.}, \ [16, 8] & (\mathtt{x}_{09}, \mathtt{x}_{07}) = (0, 0); \\ [\mathtt{x}_{00}, \mathtt{y}_{08}, \mathtt{y}_{01}], \ \textit{i.e.}, \ [16, 8, 1] & (\mathtt{x}_{09}, \mathtt{x}_{07}) = (1, 0); \\ [\mathtt{x}_{00}], \ \textit{i.e.}, \ [16] & (\mathtt{x}_{09}, \mathtt{x}_{07}) = (0, 1); \\ [\mathtt{x}_{00}, \mathtt{y}_{01}], \ \textit{i.e.}, \ [16, 1] & (\mathtt{x}_{09}, \mathtt{x}_{07}) = (1, 1). \end{cases}$$

In summary, for the 4-round differential $(0x1000, 0x4440) \to (0x0000, 0x0040)$ of SIMON32/64, there are 9 complete NB/SNBS/CSNBS, that is

1. 2 single NB: [2], [6]
2. 2 SNBS: [12, 26], [10, 14, 28]
3. 5 CSNBS in Table 9.

Table 9: CSNBS for the 4-round differential $(\mathtt{0x1000}, \mathtt{0x4440}) \to (\mathtt{0x0000}, \mathtt{0x0040})$ of SIMON32/64

Bit-set	C.	Bit-set	C.	Bit-set	C.	Bit-set	C.	Bit-set	C.
$x[1, 15]$		$x[15, 13]$		$x[13, 11]$		$x[11, 9]$		$x[9, 7]$	
[24, 10],	00	[22, 8],	00	[20],	00	[18, 4],	00	[16, 8],	00
[24, 10, 9],	10	[22, 8, 7],	10	[20, 5],	10	[18, 4, 3],	10	[16, 8, 1]	10
[24, 10, 0],	01	[22, 8, 14],	01	[20, 12],	01	[18, 4, 10],	01	[16],	01
[24, 10, 9, 0]	11	[22, 8, 7, 14]	11	[20, 12, 5]	11	[18, 4, 3, 10]	11	[16, 1]	11

C.: Conditions on $x[i, j]$, *e.g.*, $x[i, j] = 10$ means $x[i] = 1$ and $x[j] = 0$.

□: CSNBS that are used in the 16-round attack $\mathcal{A}_{II}^{\text{SIMON}_{16R}}$ on SIMON32/64.

Note that, for an input pair $((x, y), (x', y'))$ to conform the 4-round differential $(\mathtt{0x1000}, \mathtt{0x4440}) \to (\mathtt{0x0000}, \mathtt{0x0040})$, one have that

$$\begin{cases} x[5] = x'[5] = 0, \\ x[3] = x'[3] = 0. \end{cases} \quad (3)$$

The attack $\mathcal{A}_{II}^{\text{Simon}_{16R}}$. The components of $\mathcal{A}_{II}^{\text{SIMON}_{16R}}$ are as follows (refer to Fig. 22).

1. a 4-round classical differential $(\texttt{0x1000}, \texttt{0x4440}) \rightarrow (\texttt{0x0000}, \texttt{0x0040})$ (refer to the rounds colored in blue in Fig. 22), and a set of its $4 + 3$ neutral bit(-set)s (*i.e.*, four non-conditional ones $[2]$, $[6]$, $[12, 26]$, $[10, 14, 28]$ and the three ones conditioned on $x[15, 13]$, $x[13, 11]$, $x[11, 9]$ (refer to the columns framed by green lines in Table 9));
2. a 9-round neural distinguisher $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{9R}}$ trained under difference $(\texttt{0x0000}, \texttt{0x0040})$ and fed with data of type $(x, x', y \oplus y')$, and its wrong key response profiles $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{9R}}.\mu$ and $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{9R}}.\sigma$;
3. a 8-round neural distinguisher $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{8R}}$ trained under difference $(\texttt{0x0000}, \texttt{0x0040})$ and fed with data of type $(x, x', y \oplus y')$. and its wrong key response profiles $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{8R}}.\mu$ and $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{8R}}.\sigma$.

The goal is to recover the last two subkeys k_{15} and k_{14}. A difference with the attack $\mathcal{A}^{\text{SPECK}_{13R}}$ and $\mathcal{A}_I^{\text{SIMON}_{16R}}$ is that, as the neural distinguishers accept data of type $(x, x', y \oplus y')$, after guessing k_{15} and decrypting a ciphertext pair to $(x_{15}, y_{15}), (x'_{15}, y'_{15})$, one can compute $(x_{14}, x'_{14}, y_{14} \oplus y'_{14})$ by inverse one round with 0 as the subkey, and thus can be fed to $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{9R}}$. After guessing k_{14} and decrypting a pair of $(x_{15}, y_{15}), (x'_{15}, y'_{15})$ to $(x_{14}, y_{14}), (x'_{14}, y'_{14})$, one can compute $(x_{13}, x'_{13}, y_{13} \oplus y'_{13})$ by inverse one round with 0 as the subkey, and thus can be fed to $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{8R}}$.

Another difference is that, at the beginning, we guessed two key bits of k_0, that is $k_0[3]$ and $k_0[5]$, because for the 4-round differentials, the conditions for correct pairs is $x_1[5] = x'_1[5] = 0$ and $x_1[3] = x'_1[3] = 0$ (refer to Eq. 3); and four key bits $k_0[15]$, $k_0[13]$, $k_0[11]$, $k_0[9]$ for employing three conditional neutral bits (refer to Table 9). For different values of the chosen data pairs $(\tilde{x}_1, \tilde{y}_1)$, with guessed values of the four key bits, we choose different neutral bit-sets to generate the structures.

Thus, n_{kg} is $2 + 4$, and there are 2^6 outermost loops.

Except these differences, other part of the framework of attack $\mathcal{A}_{II}^{\text{SIMON}_{16R}}$ is the same as that of the 13-round attack $\mathcal{A}^{\text{SPECK}_{13R}}$ on SPECK32/64. The concrete parameters of the attack are as follows. The accuracy of $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{9R}}$ is 0.5629, and that of $\mathcal{ND}_{\mathbf{VD}}^{\text{SIMON}_{8R}}$ is 0.6587.

$$
\begin{array}{llll}
n_{kg} = 2^{2+4}, & n_{cts} = 2^{10}, & n_b = 2^{4+3}, & n_{it} = 2^{11} \\
c_1 = 20, & c_2 = 70, & n_{byit1} = n_{byit2} = 5, & n_{cand1} = n_{cand2} = 32
\end{array}
$$

The data complexity is $n_{kg} \times n_{cts} \times n_b \times 2$, that is, 2^{24} plaintexts. To examine the performance of the attack, experiments were done using 8 threads on the same GPU server testing $\mathcal{A}^{\text{SPECK}_{13R}}$. Each thread ran 20 trails, thus, 160 trails were run in total. Within the 160 trails, all trails have correct ciphertext pairs and all called the neural distinguishers. There are 52 success trails (the returned last two subkeys have hamming distance to the real subkeys at most two). Thus, the success rate is computed as 52/160, *i.e.*, 0.325.

The 160 trails took 120.25 core hours in total. For a trail, running full 2048 iterations requires less than 1 hour (about 50 minutes). Thus, the worst case to run 2^6 outermost loops for a full attack should take roughly 64 GPU hours.

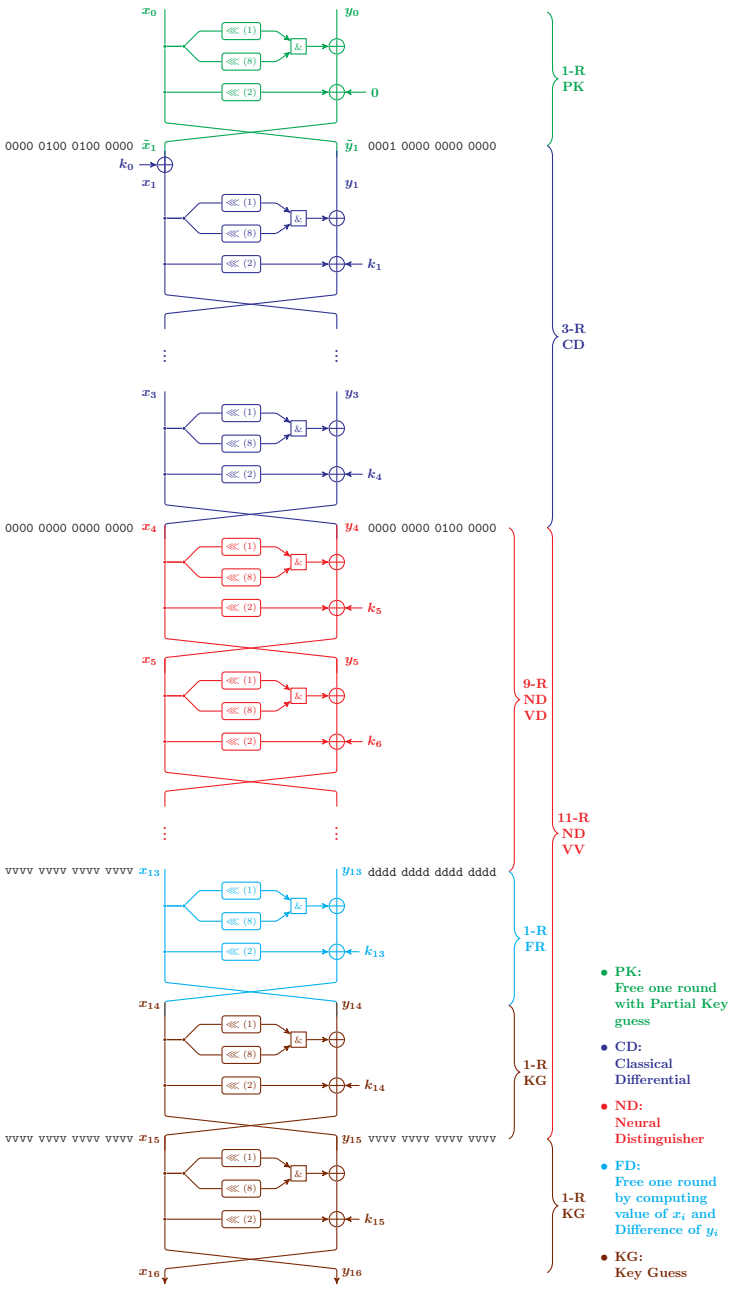


Fig. 21: Components for key-recovery attack $\mathcal{A}_I^{\text{SIMON}16R}$ on 16-round SIMON32/64

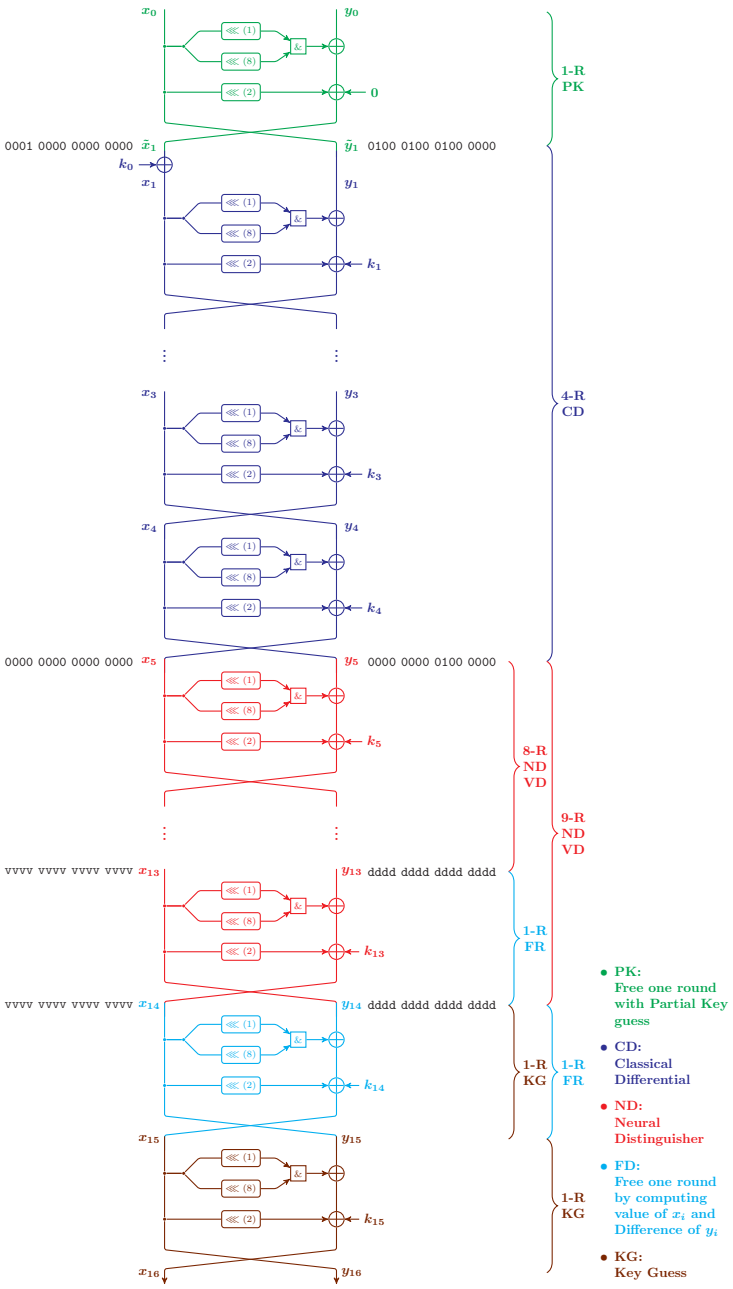


Fig. 22: Components for key-recovery attack $\mathcal{A}_{II}^{\text{SIMON}16R}$ on 16-round SIMON32/64

E Details of the Key-recovery Attack in [14]

E.1 Neutral bits Used in [14]

Table 10: (Probabilistic) single-bit neutral bit for 2-round differential $(\texttt{0x0211}, \texttt{0x0a04}) \rightarrow (\texttt{0x0040}, \texttt{0x0000})$ of SPECK32/64

NBs	Pr.	NBs	Pr.	NBs	Pr.	NBs	Pr.	NBs	Pr.	NBs	Pr.	NBs	Pr.
[20]	1	[21]	1	[22]	1	[14]	0.965	[15]	0.938	[23]	0.812	[7]	0.806
[30]	0.809	[0]	0.763	[8]	0.664	[24]	0.649	[31]	0.644	[1]	0.574		

E.2 Accuracy of the Neural Distinguishers on SPECK32/64 in [14]

Table 11: Accuracy of Gohr's neural distinguishers on SPECK32/64 [14]

#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	$\mathcal{ND}^{\text{SPECK}8R}$	$0.514 \pm 1.00 \times 10^{-3}$	$0.519 \pm 1.41 \times 10^{-3}$	$0.508 \pm 1.42 \times 10^{-3}$

E.3 BayesianKeySearch Algorithm in [14]

Algorithm 3: BayesianKeySearch Algorithm [14]

/* The description of this BayesianKeySearch Algorithm in [14] has a small typo and is inconsistent with that in the implementation codes [13], the description here corrects it according to [13]. */

Input: Ciphertext structure $\mathcal{C} := \{C_0, \cdots, C_{n_b-1}\}$, a neural distinguisher $\mathcal{ND}$, and its wrong key response profile μ and σ, 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

$S := \{k_0, k_1, \ldots, k_{n_{cand}-1}\} \leftarrow$ choose n_{cand} values at random without replacement from the set of all subkey candidates.
$L \leftarrow \{\}$
for $t = 1$ *to* n_{byit} **do**
  for $\forall k_i \in S$ **do**
    for $j = 0$ *to* $n_b - 1$ **do**
      $C'_{j,k_i} = F_{k_i}^{-1}(C_j)$
      $v_{j,k_i} = \mathcal{ND}(C'_{j,k_i})$
      $s_{j,k_i} = \log_2(v_{j,k_i}/(1 - v_{j,k_i}))$
    end
    $s_{k_i} = \sum_{j=0}^{n_b-1} s_{j,k_i}$; /* the combined score of k_i */
    $L \leftarrow L||(k_i, s_{k_i})$
    $m_{k_i} = \sum_{j=0}^{n_b-1} v_{j,k_i}/n_b$
  end
  for $k \in \{0, 1, \cdots, 2^{16} - 1\}$ **do**
    $\lambda_k = \sum_{i=0}^{n_{cand}-1} (m_{k_i} - \mu_{k_i \oplus k})^2 / \sigma^2_{k_i \oplus k}$
  end
  $S \leftarrow \text{argsort}_k(\lambda)[0 : n_{cand} - 1]$; /* Pick n_{cand} keys with the n_{cand} smallest score to form the new set of candidate keys S */
end
return L