

Cryptanalysis of Reduced-Round SipHash

Le He and Hongbo Yu

Tsinghua University, Beijing, China

he-117@mails.tsinghua.edu.cn, yuhongbo@mail.tsinghua.edu.cn

Abstract. SipHash is a family of ARX-based MAC algorithms optimized for short inputs. Already, a lot of implementations and applications for SipHash have been proposed, whereas the cryptanalysis of SipHash still lacks behind. In this paper, we study the property of truncated differential in SipHash and find out the output bits with the most imbalanced differential biases. Based on these results, we construct distinguishers with practical complexity 2^{10} for SipHash-2-1 and 2^{36} for SipHash-2-2. We further reveal the relations between the value of output bias and the difference after first modular addition step, which is directly determined by corresponding key bits. Making use of these relations, we propose a key recovery method for SipHash-2-1 with success rate increased from 2^{-128} to 2^{-41} .

Keywords: SipHash · Distinguish attack · Key recovery · Truncated differential cryptanalysis.

1 Introduction

In cryptography, a message authentication code (MAC) is a secret-key primitive that ensures the integrity of data. A MAC algorithm accepts a secret-key K and an arbitrary-length message M as input, and outputs a MAC value (a fixed size tag T). The secret-key K must be confidentially shared between two parties prior to the communication. The MAC value protects both message's integrity and authenticity, by allowing verifiers to detect any changes of the message content: the sender produces a tag T for a message M through the secret-key K , and sends the pair (M, T) to the receiver. The receiver who possesses the secret-key K can verify the authenticity of the message M by recalculating the tag T and comparing it with the sent one. If two tags are the same, the origin and the integrity of message are ensured. Otherwise, the message, or the tag, or both of them must have been modified or forged during the transmission.

SipHash [1] is an Add-Rotate-XOR (ARX) based family of MAC algorithms which was proposed by Jean Philippe Aumasson and Daniel J. Bernstein in 2012. SipHash computes a 64-bit MAC from a variable-length message M and a 128-bit secret key K . It is specifically suitable for short inputs, with performance comparable to non-cryptographic hash functions, such as CityHash [20] and SpookyHash [10]. Thus it can be used in hash tables to prevent DoS collision attack (hash flooding) or to authenticate network packets. Algorithms in

SipHash family are denoted as SipHash- c - d , where c is the number of compression rounds per message block and d is the number of finalization rounds after compression steps. The recommended parameters are SipHash-2-4 for the best performance, and SipHash-4-8 for conservative security.

Most existing results on SipHash concentrate on implementations and applications [3, 18, 19, 21], whereas there has been little progress in mounting an attack so far even in simplified versions of SipHash. In [1], the designers provide the differential cryptanalysis of SipHash and in [9], Christoph Dobraunig and Florian Mendel et al provide the first external security analysis regarding differential cryptanalysis. All the differential cryptanalysis results can reach a limited number of rounds because standard differential trails diverge quickly in SipHash and it is hard to keep the number of active bits at a low level.

However, truncated differential cryptanalysis is a powerful technique to help solve the problem. Truncated differential cryptanalysis [11] is a generalization of differential cryptanalysis against block ciphers, which was developed by Lars Knudsen in 1994. Unlike ordinary differential cryptanalysis [6] that analyzes full differences between two texts, the truncated variant only considers differences partially determined. That means the attack merely makes prediction of some specific bits instead of the full output.

This technique has been applied in Salsa [2], E2 [16], Skipjack [13], Chaskey [15], SAFER [12], Twofish [17] and even the stream cipher Salsa20 [8]. In [2] and [15] the authors set the input difference as 1-bit to detect output biases, which gives us great inspiration. As a result, they get differential trails with more rounds than ordinary differential cryptanalysis. In [7], it is shown that the data complexity of truncated differential cryptanalysis is $\Omega(\varepsilon^{-2})$ (ε denotes the imbalance of truncated differential). In Section 3, the relation between data complexity and truncated differential bias is further discussed.

Table 1. Best cryptanalytic results of SipHash

Variant	Type of Attack	Probability	Reference
SipHash-2-1	Distinguisher	2^{-56}	[1]
SipHash-2-2	Distinguisher	2^{-159}	[1]
SipHash-2-x	Internal collision	$2^{-236.3}$	[9]
SipHash-1-x	Internal collision	2^{-167}	[9]
4-Round Finalization	Distinguisher	2^{-35}	[9]
SipHash-2-1	Distinguisher	2^{-10}	This Paper
SipHash-2-2	Distinguisher	2^{-36}	This Paper
SipHash-2-1	Key Recovery	2^{-41}	This Paper

Relative Work. In [1], the designers provide their differential trail up to SipHash-2-4, with success rates 2^{-56} for 3 rounds and 2^{-159} for 4 rounds. This directly constructs a kind of distinguishers for SipHash-2-1 and SipHash-2-2. In [9], the authors propose a method to construct internal collisions. Using the results, they further present a distinguisher for 4-round finalization (the finalization part

of Siphash2-4). Although the complexity is very close to our distinguisher for SipHash-2-2, the techniques involved are quite different.

Our Contributions. This paper studies the output biases under specific input differences in reduced-round SipHash. Inspired by truncated differential cryptanalysis above, we introduce 1-bit difference to the input and detect if biased bits exist in the output of reduced-round SipHash. We exhaustively search a variety of 1-bit input differences (from bit 0 to 63) and obtain corresponding biased output bits with great imbalances. Using these results, we construct distinguishers for SipHash-2-1 and SipHash-2-2 with practical complexity (see in **Table 1**). We further reveal the relations between the value of differential bias and corresponding key bits. As a main application, we propose a key recovery method that can recover the 128-bit key with success rate over 2^{-41} .

Organization of the Paper. The paper starts with a description of SipHash in Section 2. In Section 3, some basic knowledge about probability theory and analysis techniques related to this work are given. The following Section 4 introduces the results of differential cryptanalysis and Section 5 constructs distinguishers for reduced-round SipHash. Key recovery is discussed in Section 6 and the conclusion is presented in Section 7.

2 Description of SipHash

SipHash is an ARX primitive operated on 64-bit words. Different SipHash versions are denoted as SipHash- $c-d$, where c is the number of compression rounds processing each message block and d is the number of finalization rounds. SipHash possesses a 256-bit internal state, uses a 128-bit key and outputs a 64-bit tag.

The 64-bit tag is computed as follows.

Initialization. The internal state of SipHash consists of four 64-bit words v_0, v_1, v_2 and v_3 . The 128-bit key can be expressed as $k = k_1 \parallel k_0$. Four 64-bit words of internal state are initialized as:

$$\begin{aligned} v_0 &= k_0 \oplus h_0 = k_0 \oplus 736f6d6570736575 \\ v_1 &= k_1 \oplus h_1 = k_1 \oplus 646f72616e646f6d \\ v_2 &= k_0 \oplus h_2 = k_0 \oplus 6c7967656e657261 \\ v_3 &= k_1 \oplus h_3 = k_1 \oplus 7465646279746573 \end{aligned}$$

Parsing. The ω -byte input m is parsed into w 64-bit little-endian words $m_0 \dots m_{w-1}$ before compression where $w = 1 + \text{int}(\omega/8)$. The word m_{w-1} includes the last $\omega \bmod 8$ bytes, filled with bytes 00 if needed and followed by a byte encoding the positive integer $\omega \bmod 256$ in the end. For example, 1-byte input $m = ab$ will be parsed as $m_0 = 0x01000000000000ab$.

Compression. For each message block m_i in sequence, SipHash- $c-d$ will compress it and update four internal state words through three steps. The first step is $v_3 = v_3 \oplus m_i$, and then c rounds of SipRound are iteratively executed, followed by the final step $v_0 = v_0 \oplus m_i$.

Finalization. After all message blocks have been processed, the constant $0xff$ (255) is xored to v_2 . Then d iterations of SipRound are performed and SipHash- $c-d$ returns the 64-bit MAC $v_0 \oplus v_1 \oplus v_2 \oplus v_3$ at last.

The function SipRound updates the internal state by (also see in **Fig.1**):

$$\begin{array}{ll}
 v_0 + = v_1 & v_2 + = v_3 \\
 v_1 \lll = 13 & v_3 \lll = 16 \\
 v_1 \oplus = v_0 & v_3 \oplus = v_2 \\
 v_0 \lll = 32 & \\
 v_2 + = v_1 & v_0 + = v_3 \\
 v_1 \lll = 17 & v_3 \lll = 21 \\
 v_1 \oplus = v_2 & v_3 \oplus = v_0 \\
 v_2 \lll = 32 &
 \end{array}$$

Symbols +, \lll and \oplus separately represent modular addition, left rotation and xor. Each operation is operated on 64-bit words.

Fig.1 shows the structure of the major function SipRound.

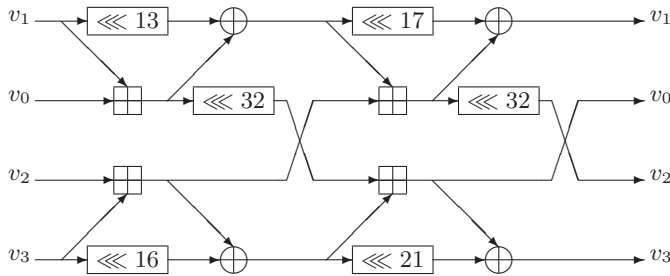


Fig. 1. ARX network of SipRound [1]

Fig.2 shows the procedure of hashing a 15-byte message by SipHash-2-4.

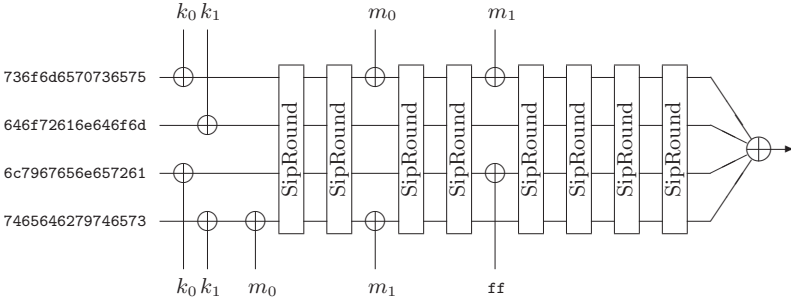


Fig. 2. SipHash-2-4 processing a 15-byte message [1]

In this paper, distinguisher and key recovery are restricted to one-block attacks, which means the most significant byte of m_0 must be 07. So input differences on such byte are actually impractical. However, it is still assumed that attackers have the ability to alter the most significant byte. In other words, the parsing restriction of SipHash is neglected in this paper. When considering the parsing restriction, the complexities of distinguisher and key recovery will be a bit higher.

3 Knowledge about Probability Theory

In this section, we discuss some knowledge about probability theory which is related to our work. The first part reveals the relationship between the value of output bias and the needed amount of input data according to Central Limit Theorem (CLT). The second part introduces the method to convert prior probabilities into posterior probabilities through Bayes' Formula.

3.1 Application of CLT in Bias Test

Suppose $\{x_1, x_2, \dots, x_n\}$ is a group of independent samples of random variable X , which has $EX = \mu$ and $DX = \sigma^2$. Central Limit Theorem tells us that $X^* = \sum_{i=1}^n (X_i - \mu) / \sqrt{n}\sigma$ can be well approximated by standard normal distribution $N(0, 1)$ as long as n is large enough. This theorem holds even if X is not normal.

In our bias tests of reduced-round SipHash, X can be regarded as a Bernoulli random variable. Under a pair of random message blocks (m, m') holding $m \oplus m' = 1 \ll k$ (1-bit difference), set $X = 1$ if the difference of a certain output bit $output_j \oplus output'_j = 1$ and $X = 0$ if not, where (k, j) is a pair of fix numbers chosen from 0 to 63 before the tests.

Let b denote the output bias (b can be negative), which means $P(X = 1) = 0.5 + b$ and $P(X = 0) = 0.5 - b$. We can estimate b by sampling a huge number of X_i and calculating $\bar{X} = \sum_{i=1}^n X_i / n - 0.5$. Compare \bar{X} and X^* : since $|b|$ is much smaller than 0.5, we approximately have $\sigma^2 = 0.25 - b^2 \approx 0.25$. Then we get $\bar{X} \sim N(b, 1/4n)$ or $b \sim N(\bar{X}, 1/4n)$. It is seen that the standard deviation $\sigma = \sqrt{1/4n}$ should be in the same order of magnitude as $|b|$. Otherwise, higher standard deviation may lead to great disturbance when counting result \bar{X} . Thus the amount of input data n must be $\Omega(b^{-2})$.

Next, we concretely discuss the constant coefficient, which also plays a role in data complexity. This refers to Pauta criterion or 3σ -principle is the same, which tells us that a random sample of $N(\mu, \sigma^2)$ will locate in $(\mu - 3\sigma, \mu + 3\sigma)$ with probability over 99.73%. In this paper, we use 4σ -principle to further promise the correctness with probability over 99.99%. According to 4σ -principle, if we find $|\bar{X}| = 2^{-t}$, $2^{-t} > 4\sigma = \sqrt{4/n}$ is required to ensure the output bias, which infers $n = 2^{2t+2}$. That is the relationship between the bias value and the data amount. Take $n = 2^{20}$ with $4\sigma = 2^{-9}$ as examples. If $|\bar{X}| > 2^{-9}$ is found, the corresponding output bias $|b| > 0$ almost definitely exists with probability over 99.99%. If $|\bar{X}|$ is found to be 2^{-10} or 2^{-11} , the probability will drop to 95.45% or 68.27%.

3.2 Prior Probability and Posterior Probability

Sometimes we can easily get the prior probability $P(B|A)$, but cannot directly test the posterior probability $P(A|B)$. For example, if the 128-bit key of SipHash is fixed, we can easily test any property of the output $P(output|key)$ by giving random inputs and calculating corresponding outputs. However, we cannot test $P(key|output)$ because it is impossible to recover the 128-bit key from a 64-bit output. In this situation, we can convert prior probabilities into posterior probabilities through Bayes' Formula:

$$P(A_k|B) = \frac{P(A_k)P(B|A_k)}{\sum P(A_i)P(B|A_i)}$$

Among the formula above, A_1, A_2, \dots, A_m are m incompatible events which means $A_i \cap A_j = \emptyset$ and $\sum A_i = \Omega$. In this paper, the number of incompatible events m is always 2 where A_1 represents a certain *keybit* is 0 and A_2 represents a certain *keybit* is 1. B may be a property or a combination of output biases. Take **Table 2** as an example.

Table 2. Determination with two events

Event	$B_1 : \bar{X}_j \leq fixbound$	$B_2 : \bar{X}_j > fixbound$
$A_1 : keybit = 0$	p_1	$1 - p_1$
$A_2 : keybit = 1$	p_2	$1 - p_2$

Suppose we need to determine some *keybit* by testing the output bias \bar{X}_j under a certain 1-bit input difference k . First, randomly generate different keys that fit *keybit* = 0 or *keybit* = 1. Second, calculate output bias \bar{X}_j under those keys separately. Finally, count the number of keys that meet $|\bar{X}_j| \leq fixbound$ and get the table, in which $P(B_1|A_1) = p_1$ and $P(B_1|A_2) = p_2$.

As for the *keybit* to be determined, we also calculate corresponding bias \bar{X}_j and judge whether $|\bar{X}_j| \leq fixbound$ holds. According to Bayes' Formula and $P(A_1) = P(A_2) = 0.5$, we have:

$$\begin{aligned} P(A_1|B_1) &= \frac{p_1}{p_1 + p_2} \\ P(A_2|B_1) &= \frac{p_2}{p_1 + p_2} \\ P(A_1|B_2) &= \frac{1 - p_1}{2 - p_1 - p_2} \\ P(A_2|B_2) &= \frac{1 - p_2}{2 - p_1 - p_2} \end{aligned}$$

Then we can determine it by choosing the larger of $P(A_1|B_1)$ and $P(A_2|B_1)$ or the larger of the other two if B_1 or B_2 happens. The expected success rate is

$P(B_1) * \max(P(A_1|B_1), P(A_2|B_1)) + P(B_2) * \max(P(A_1|B_2), P(A_2|B_2))$, where $P(B_1) = p_1P(A_1) + p_2P(A_2)$ and $P(B_2) = (1 - p_1)P(A_1) + (1 - p_2)P(A_2)$.

Finally, the expected success rate can be simplified into:

$$\frac{\max(p_1, p_2) + \max(1 - p_1, 1 - p_2)}{2} = \frac{1 + |p_1 - p_2|}{2}$$

4 Differential Cryptanalysis of Reduced-Round SipHash

In this section, we start with the standard differential trail of SipHash-2-4 based on previous work [4, 5, 14, 22]. It is found that all the differential cryptanalysis results can reach a limited number of rounds where standard differential trails diverge quickly. Next, we investigate if biased differential bits exist in the output of reduced-round SipHash instead of analyzing full differences. As a result, we find out some interesting phenomena about the output bias.

4.1 Standard Differential Cryptanalysis Results

In [1], the designers provide the differential cryptanalysis of SipHash-2-4. The differential trail is listed in **Table 3**.

Table 3. Differential characteristics reported by the designers [1]

Round	Difference	Prob.
1	0000000000000000 0000000000000000 0000000000000000 8000000000000000	1
	0000000000000000 0000000000000000 8000000000000000 80000000000008000	(1)
2	80000000000008000 8000000000000000 0000000080000000 8000001000108000	13
	00008000000000000 00000000000009000 8000001080108000 8010000000100000	(14)
3	0010800000100000 80000011a0101000 8010100080000010 8010820000000200	42
	a000100080108011 8012b413a2000000 0000920080000210 8200920082008200	(56)
4	2200820002100211 e835621322010235 2200021080122613 6210c21042004203	103
	20110024ca35e013 667784530057bd22 4010c000c2126410 8200820080110600	(159)
5	a21182244a24e613 2ec144fcb80115dd c245d93226674453 e20180048a34a603	152
	f225f3ce8cd0c6d8 a44f51d8d09e5616 20445936ac53e250 a040d3020a500051	(311)
6	526520cc8680c689 27baa9d2d0e0fcd8 7ccdb446840b08ee 32246acc8cb4ce93	187
	56603a5175df891e 20e5d30249fb3ea6 4ee9de8a08bfc67d 2425523ec62cf459	(498)

As we can see from the table, the probability of differential characteristic is less than 2^{-128} for more than 3 rounds. Notice that the upper bound of distinguishing complexity is 2^{128} , because the complexity of exhaustively searching all possible keys is only 2^{128} . Therefore, the differential cryptanalysis provided in [1] cannot help in distinguishing for SipHash-2-2.

The best differential trail given in [9] is used in internal collision of SipHash-2-4 with a probability of $2^{-236.3}$. Compared with the trail in [1], it is better for constructing internal collisions. Using these results, a distinguisher for 4-round finalization is further presented. However, this trail is unsuitable for constructing distinguishers for SipHash-2-x because all the active bits of the trail occur in the first two rounds.

In summary, we cannot construct distinguishers for SipHash-2-2 based on present standard differential cryptanalysis. However, truncated differential cryptanalysis can help improve the results.

4.2 Truncated Differential Cryptanalysis Results

In this part, we give some results of our truncated differential cryptanalysis of SipHash-2-1 and SipHash-2-2. As mentioned above, we set 1-bit input difference on the k -th (0 to 63 starting from the least significant) input bit and calculate differential biases of all 64 output bits. For each test, we randomly choose 4096 keys with data complexity $n = 2^{20}$ in SipHash-2-1 and 256 keys with $n = 2^{40}$ in SipHash-2-2, which can detect output biases over 2^{-9} and 2^{-19} .

We finally find out some thought-provoking phenomena as below:

Observation 1. In SipHash-2-1, we can always find some biased bits with obvious biases, among which the greatest one varies from 2^{-3} to 2^{-7} .

Observation 2. In SipHash-2-2, for a proportion of keys we can find some bits with biases varying from 2^{-14} to 2^{-19} , while for the other proportion we can find nothing.

Observation 3. In both SipHash-2-1 and SipHash-2-2, the relations between the input differential bits and the biased output bits show a kind of rotation property, which means if output bit j has differential bias with input differential bit k , then output bit $j + i$ is much likely to have the same differential bias with input differential bit $k + i$.

Table 4 shows output bits with the greatest imbalance for each 1-bit input difference in SipHash-2-1, which also reveals the rotation property in a way. In each entry, we give at most 2 bits that may have the greatest imbalance with the highest probability. If an entry gives only 1 bit, it means those bits with the greatest imbalance are almost centralized into that position.

Table 4. Bits with the greatest imbalance in SipHash-2-1

k	δ	λ	l	k	δ	λ	l
0	0000000000000001	10,26	-7	32	0000000100000000	42,58	-7
1	0000000000000002	11,27	-7	33	0000000200000000	43,59	-7
2	0000000000000004	12,28	-7	34	0000000400000000	44,60	-7
3	0000000000000008	13,29	-7	35	0000000800000000	45,61	-7
4	0000000000000010	14,30	-7	36	0000001000000000	46,62	-7
5	0000000000000020	15,31	-7	37	0000002000000000	47,63	-7
6	0000000000000040	32	-6	38	0000004000000000	0	-5

7	0000000000000080	17,33	-6	39	0000008000000000	1	-5
8	0000000000000100	18,34	-6	40	0000010000000000	2	-5
9	0000000000000200	19,35	-6	41	0000020000000000	3	-6
10	0000000000000400	20,36	-7	42	0000040000000000	4,52	-7
11	0000000000000800	21,37	-7	43	0000080000000000	5,53	-7
12	0000000000001000	22,38	-7	44	0000100000000000	6,54	-7
13	0000000000002000	23,39	-7	45	0000200000000000	55	-6
14	0000000000004000	24,40	-7	46	0000400000000000	56	-6
15	0000000000008000	25,41	-7	47	0000800000000000	13	-6
16	0000000000010000	26,42	-7	48	0001000000000000	10,58	-7
17	0000000000020000	27,43	-7	49	0002000000000000	11,59	-7
18	0000000000040000	28,44	-7	50	0004000000000000	12,60	-7
19	0000000000080000	29,45	-7	51	0008000000000000	13,61	-7
20	000000000100000	46	-7	52	0010000000000000	14,62	-7
21	000000000200000	31,47	-7	53	0020000000000000	15,63	-7
22	000000000400000	32	-5	54	0040000000000000	16,0	-7
23	000000000800000	33	-5	55	0080000000000000	17,1	-7
24	0000000001000000	34	-6	56	0100000000000000	18,2	-7
25	0000000002000000	35	-6	57	0200000000000000	19,3	-7
26	0000000004000000	36	-6	58	0400000000000000	20,4	-7
27	0000000008000000	37	-7	59	0800000000000000	21	-6
28	0000000010000000	38	-7	60	1000000000000000	22	-5
29	0000000020000000	39,55	-7	61	2000000000000000	23	-5
30	0000000040000000	40,56	-7	62	4000000000000000	24	-5
31	0000000080000000	41,57	-7	63	8000000000000000	25	-4

Explanations for Table 4. Symbol k denotes the input differential bit and δ is the corresponding 1-bit input difference. Symbol λ shows output bits with the greatest imbalance under input difference δ . Let b_λ denote the bias of output bit λ . Negative integer l reflects the level of bias b_λ , which means $|b_\lambda| > 2^l$ holds for all tested keys. This bias level is related to the complexity of distinguishing.

Source codes are provided at <https://github.com/hele27/SipHash>, including both bias test of this section and key classification discussed in Section 6.

In this paper, we are not concerned about why it shows a rotation property or why it reaches such a bias level. However, a great number of experiments can support those observations. Based on these results, we construct a distinguisher and propose a key recovery method for reduced-round SipHash. The details are presented in the following sections.

5 Distinguishing for SipHash-2-1 and SipHash-2-2

We have revealed the relationship between the output bias and the data complexity in Section 3. To detect an output bias $|b| > 2^{-t}$, $n = 2^{2t+2}$ data complexity is required. With the same complexity, a distinguisher can be constructed for SipHash-2-1 according to 4σ -principle.

The algorithm is given below.

Algorithm 1 Distinguisher for SipHash-2-1

Input: a 64-bit-output function $f(m)$ to be distinguished.
Output: 0 for $f(m)$ is pseudorandom and 1 for $f(m)$ is SipHash-2-1.
1: $k = 63$; $\lambda = 25$; $l = -4$; $//k$ is the input differential bit, λ is the output biased bit and l is the bias level.
2: $n = 2^{-2l+2}$;
3: **for** $i = 1 : n$ **do**
4: Randomly generate 64-bit message pair (m, m') holding $m \oplus m' = 1 \lll k$;
5: Query $output = f(m)$ and $output' = f(m')$;
6: Compute the difference of the λ -th output bit $output_\lambda \oplus output'_\lambda$;
7: Set Bernoulli variable $X_i = output_\lambda \oplus output'_\lambda$;
8: **end for**
9: Compute $\bar{X} = \sum_{i=1}^n X_i/n - 0.5$;
10: Return $(|\bar{X}| > 2^l)$

Parameters k, λ, l can be determined according to **Table 4**.

For the best performance, we choose $k = 63$, $\lambda = 25$ and $l = -4$. Experiments have shown that under $k = 63$ in SipHash-2-1, the output bias b_{25} meets $|b_{25}| > 2^{-4}$ for all keys. Therefore, complexity $n = 2^{10}$ is enough for the algorithm. If $f(m)$ is SipHash-2-1, $|\bar{X}| > 2^{-4}$ holds with 100% probability. If $f(m)$ is a pseudorandom function without output biases, $|\bar{X}| < 2^{-4}$ holds with probability over 99.99% according to 4σ -principle. Thus **Algorithm 1** can distinguish SipHash-2-1 from a pseudorandom function with success rate over 99.99%.

Similar algorithm can be applied to SipHash-2-2.

Algorithm 2 Distinguisher for SipHash-2-2

Input: a 64-bit-output function $f(m)$ to be distinguished.
Output: 0 for $f(m)$ is pseudorandom and 1 for $f(m)$ is SipHash-2-2.
1: $k = 63$; $\lambda = 57$; $l = -17$; $//k$ is the input differential bit, λ is the output biased bit and l is the bias level.
2: $n = 2^{-2l+2}$;
3: **for** $i = 1 : n$ **do**
4: Randomly generate 64-bit message pair (m, m') holding $m \oplus m' = 1 \lll k$;
5: Query $output = f(m)$ and $output' = f(m')$;
6: Compute the difference of the λ -th output bit $output_\lambda \oplus output'_\lambda$;
7: Set Bernoulli variable $X_i = output_\lambda \oplus output'_\lambda$;
8: **end for**
9: Compute $\bar{X} = \sum_{i=1}^n X_i/n - 0.5$;
10: Return $(|\bar{X}| > 2^l)$

For the reason of computing power, we cannot test all k in SipHash-2-2 like **Table 4**. Inspired by the good performance of **Algorithm 1** (and the differential trail in **Table 3** as well), we remain $k = 63$ in distinguishing for SipHash-2-2. It is found that $|b_{57}| > 2^{-17}$ holds for all 256 tested keys. Similar analysis can

be given as the above. Thus **Algorithm 2** can distinguish SipHash-2-2 from a pseudorandom function with success rate over 99.99% and the complexity is 2^{36} .

6 Key Recovery Attack for SipHash-2-1

In this section, we discuss key recovery attack for SipHash-2-1. First, we discuss the key classification by carry existences in the first modular addition $v_2+ = v_3$. Next, we present the results of our key classification experiments. Finally, we propose a method to recover the 128-bit key according to those results and give a success rate analysis.

6.1 Key Classification

Experiments in Section 4 have shown that values of the greatest output biases vary from 2^{-3} to 2^{-7} under different keys. This inspires us to design a classification so that keys in different classes will lead to output biases of different levels. With the classification, some information about the key can be obtained through the output bias.

For 1-bit input difference on $m[k]$, step $v_3 = v_3 \oplus m$ leads to 1-bit difference on $v_3[k]$ firstly. The differential trail continues with step $v_2+ = v_3$, while step $v_0+ = v_1$ is independent of m .

Notice the fact that for an ARX-based algorithm, the differential trail diverges and the number of active bits increases when carry appears in modular additions. For example, the difference after $v_2+ = v_3$ remains $1 \ll k$ without carry while it becomes $11 \ll k$ or even more consecutive 1 if carries exist.

Based on this fact, we propose an assumption that with difference $1 \ll k$ after first modular addition $v_2+ = v_3$, output biases may be greater and easier to detect than those with difference $11 \ll k$ or more consecutive 1. This assumption provides a classification method by carry existences and the results of key classification experiments can support our assumption.

Details of key classification are given below. Suppose 1-bit difference is set on the k -th bit, which means $m[k] \oplus m'[k] = 1$.

- *Situation $k = 0$.* The difference is introduced to $v_3[0]$ first.
 - In this situation, it can be divided into 2 classes $v_2[0] = 0$ and $v_2[0] = 1$.
 - If $v_2[0] = 0$ holds, the difference must be $0x0000000000000001$ without carry for both of $m[0]$ and $m'[0]$.
 - If $v_2[0] = 1$ holds, the difference becomes $0x0000000000000011$ at least with carry existing for either of $m[0]$ and $m'[0]$.
- *Situation $k > 0$.* The difference is introduced to $v_3[k]$ first.
 - In this situation, $v_2[k-1]$, $v_3[k-1]$ and even less significant bits also work besides $v_2[k]$. For simplicity, we divide them into 8 classes by the values of $v_2[k]$, $v_2[k-1]$ and $v_3[k-1]$.
 - If $v_2[k-1] = v_3[k-1] = 0$ holds, carry existences for $m[k]$ and $m'[k]$ are irrelevant to less significant bits, only depending on $v_2[k]$. Difference $1 \ll k$ can be ensured with $v_2[k] = 0$ or avoided with $v_2[k] = 1$.

- If $v_2[k-1] = v_3[k-1] = 1$ holds, carry existences for $m[k]$ and $m'[k]$ are irrelevant to less significant bits, only depending on $v_2[k]$. Difference $1 \ll k$ can be ensured with $v_2[k] = 1$ or avoided with $v_2[k] = 0$.
- In other cases of $v_2[k-1] \oplus v_3[k-1] = 1$, carry existences are indefinite because of unknown less significant bits. The difference after $v_2+ = v_3$ is still uncontrollable no matter what $v_2[k]$ is.

According to the classification, we randomly generate a great number of keys that meet the conditions separately and perform the key classification experiments (4096 keys in each group). We use a boundary-line $e_j \leq bound$ to filter those keys with difference $1 \ll k$ after $v_2+ = v_3$, where e_j equal to $\log_2 |b_j|$ is the exponent part of tested output bias. It is expected that output biases under keys in different classes will locate on different sides of the boundary-line.

The results of our key classification experiments are given in **Table 5**, which is a simplified table for SipHash-2-1. The full table is given in **Appendix**.

Table 5. Results of key classification experiments (partial)

k	conditions	$e_j \leq bound$	proportion	success rate
0	$v_2[k] = 0$	$e_{26} \leq -5.255$	0%	100%
	$v_2[k] = 1$		100%	
1	$v_2[k] = 0 \& v_2[k-1] = 0 \& v_3[k-1] = 0$	$e_{27} \leq -5.285$	0%	100%
	$v_2[k] = 0 \& v_2[k-1] = 0 \& v_3[k-1] = 1$		0%	
	$v_2[k] = 1 \& v_2[k-1] = 0 \& v_3[k-1] = 0$		100%	
	$v_2[k] = 1 \& v_2[k-1] = 0 \& v_3[k-1] = 1$	100%	$e_{30} \leq -7.635$	19.5%
	$v_2[k] = 0 \& v_2[k-1] = 1 \& v_3[k-1] = 0$	51.0%		
	$v_2[k] = 0 \& v_2[k-1] = 1 \& v_3[k-1] = 1$	49.7%		
$v_2[k] = 1 \& v_2[k-1] = 1 \& v_3[k-1] = 0$	16.7%			
...
31	$v_2[k] = 0 \& v_2[k-1] = 0 \& v_3[k-1] = 0$	$e_9 \leq -6.505$	0%	90.5%
	$v_2[k] = 0 \& v_2[k-1] = 0 \& v_3[k-1] = 1$		16.8%	
	$v_2[k] = 1 \& v_2[k-1] = 0 \& v_3[k-1] = 0$		97.3%	
	$v_2[k] = 1 \& v_2[k-1] = 0 \& v_3[k-1] = 1$	17.1%	$e_9 \leq -6.525$	15.8%
	$v_2[k] = 0 \& v_2[k-1] = 1 \& v_3[k-1] = 0$	96.3%		
	$v_2[k] = 0 \& v_2[k-1] = 1 \& v_3[k-1] = 1$	16.3%		
$v_2[k] = 1 \& v_2[k-1] = 1 \& v_3[k-1] = 0$	0%			
...
63	$v_2[k] = 0 \& v_2[k-1] = 0 \& v_3[k-1] = 0$	$e_{41} \leq -3.935$	88.8%	50.9%
	$v_2[k] = 0 \& v_2[k-1] = 0 \& v_3[k-1] = 1$		88.8%	
	$v_2[k] = 1 \& v_2[k-1] = 0 \& v_3[k-1] = 0$		90.3%	
	$v_2[k] = 1 \& v_2[k-1] = 0 \& v_3[k-1] = 1$	89.3%	$e_{44} \leq -5.095$	28.3%
	$v_2[k] = 0 \& v_2[k-1] = 1 \& v_3[k-1] = 0$	29.1%		
	$v_2[k] = 0 \& v_2[k-1] = 1 \& v_3[k-1] = 1$	29.9%		
$v_2[k] = 1 \& v_2[k-1] = 1 \& v_3[k-1] = 0$	30.9%			

Explanations for Table 5. Symbol k denotes the input differential bit. The conditions have been discussed above that there are 2 classes for $k = 0$ and 8 classes for $k > 0$. The percentage of a group shows the proportion of keys that meet the boundary-line inequation. For example, among all 4096 keys in class $v_2[k] = 1 \& v_2[k-1] = 1 \& v_3[k-1] = 1$, only 16.7% of them show $e_{30} \leq -7.635$ under $k = 1$.

The boundary-line is chosen to make great disparities among the proportions. Notice that the boundary-line must be the same in different groups when determining a certain $v_2[k]$. As for how to define the boundary-line or how to calculate the expected success rate, more details are given in the next part.

6.2 Key Recovery Method

Making use of the results in **Appendix**, we can propose a bit-by-bit key recovery method. Firstly, we set the input difference on $k = 0$ and test e_{26} , determining $v_2[0]$ by whether $e_{26} \leq -5.255$ holds. We can continue recovering $v_2[1]$ by choosing boundary-line $e_{27} \leq -5.285$ for $v_2[0] = 0$ or $e_{30} \leq -7.635$ for $v_2[0] = 1$ after $v_2[0]$ is determined. Similar procedures can be executed until $v_2[63]$.

Now we discuss the specific method and the calculation of expected success rate in a single step. **Table 6** shows the prior probabilities in different situations.

Table 6. Situations of $k = 0$ and $k > 0$

k	conditions	$P(e_j \leq bound)$	$P(e_j > bound)$
$k = 0$	$v_2[k] = 0$	p_1	$1 - p_1$
	$v_2[k] = 1$	p_2	$1 - p_2$
$k > 0$	$v_2[k] = 0 \& v_3[k-1] = 0$	p_1	$1 - p_1$
	$v_2[k] = 0 \& v_3[k-1] = 1$	q_1	$1 - q_1$
	$v_2[k] = 1 \& v_3[k-1] = 0$	p_2	$1 - p_2$
	$v_2[k] = 1 \& v_3[k-1] = 1$	q_2	$1 - q_2$

Step $k = 0$. A similar determination has been discussed in Section 3. Using Bayes' Formula, we get:

$$\begin{aligned}
 P(v_2[k] = 0 | e_j \leq bound) &= \frac{p_1}{p_1 + p_2} \\
 P(v_2[k] = 1 | e_j \leq bound) &= \frac{p_2}{p_1 + p_2} \\
 P(v_2[k] = 0 | e_j > bound) &= \frac{1 - p_1}{2 - p_1 - p_2} \\
 P(v_2[k] = 1 | e_j > bound) &= \frac{1 - p_2}{2 - p_1 - p_2}
 \end{aligned}$$

And the expected success rate is $(1 + |p_1 - p_2|)/2$.

Step $k > 0$ with $v_2[k-1]$ determined. A problem is we cannot get the value of $v_3[k-1]$: if we are able to get $v_3[k-1]$, the calculation is the same as step $k = 0$.

However, without knowing $v_3[k-1]$, we can still control $m[k-1]$ to support the determination. We first fix $m[k-1] = 0$, set input difference on $m[k]$, and test corresponding e_j . Then we flip $m[k-1]$ and perform the same experiment once again. According to the combination of two results, we are able to reach a determination. There are three kinds of combinations: both $e_j \leq bound$ hold (denoted as ll), neither $e_j \leq bound$ hold (denoted as rr), and either of them holds (denoted as lr). Since we have controlled $m[k-1]$ in two experiments, we know one meets $v_3[k-1] = 0$ and the other meets $v_3[k-1] = 1$.

According to **Table 6**, probabilities of those three events can be calculated as shown in **Table 7**. Notice that the results of $m[k-1] = 0$ and $m[k-1] = 1$ are actually relevant, but we still regard them as independent of each other to give an approximate analysis that can be calculated.

Table 7. Determination with three events

Event	$B_1 : ll$	$B_2 : lr$	$B_3 : rr$
$A_1 : v_2[k] = 0$	$p_1 q_1$	$p_1 + q_1 - 2p_1 q_1$	$(1 - p_1)(1 - q_1)$
$A_2 : v_2[k] = 1$	$p_2 q_2$	$p_2 + q_2 - 2p_2 q_2$	$(1 - p_2)(1 - q_2)$

Similarly, we have:

$$\begin{aligned}
P(v_2[k] = 0|ll) &= \frac{p_1 q_1}{p_1 q_1 + p_2 q_2} \\
P(v_2[k] = 1|ll) &= \frac{p_2 q_2}{p_1 q_1 + p_2 q_2} \\
P(v_2[k] = 0|lr) &= \frac{p_1 + q_1 - 2p_1 q_1}{p_1 + q_1 + p_2 + q_2 - 2p_1 q_1 - 2p_2 q_2} \\
P(v_2[k] = 1|lr) &= \frac{p_2 + q_2 - 2p_2 q_2}{p_1 + q_1 + p_2 + q_2 - 2p_1 q_1 - 2p_2 q_2} \\
P(v_2[k] = 0|rr) &= \frac{(1 - p_1)(1 - q_1)}{2 - p_1 - q_1 - p_2 - q_2 + p_1 q_1 + p_2 q_2} \\
P(v_2[k] = 1|rr) &= \frac{(1 - p_2)(1 - q_2)}{2 - p_1 - q_1 - p_2 - q_2 + p_1 q_1 + p_2 q_2}
\end{aligned}$$

And the expected success rate is $(\max(p_1 q_1, p_2 q_2) + \max(p_1 + q_1 - 2p_1 q_1, p_2 + q_2 - 2p_2 q_2) + \max((1 - p_1)(1 - q_1), (1 - p_2)(1 - q_2)))/2$.

As for how to define the boundary-line, since the formula of expected success rate calculation has been known, we can scan all 4^*64^*4096 possible $(e_j, bound)$ ($bound > -9$ must hold first) for each determination and choose one with the highest expected success rate, which is a kind of optimization. A question is how

well an optimized boundary-line will work in another data group. This problem is discussed in the final part of this section.

6.3 Success Rate Analysis

According to **Table 5** above or **Appendix**, we can recover $v_2[0]$ and $k_0[0]$ with 100% success rate. Behind $v_2[0]$, we can continue recovering $v_2[1]$ and $k_0[1]$: if $v_2[0] = 0$ holds, the success rate is 100% and if $v_2[0] = 1$ holds, the success rate becomes 51.2%. So the average success rate of recovering $k_0[1]$ is $(100\% + 51.2\%)/2 = 75.6\%$. Bits $k_0[2]$ to $k_0[63]$ can be similarly recovered step by step.

The probability of correctly recovering all k_0 is:

$$\frac{100\% + 51.2\%}{2} * \frac{99.6\% + 75.1\%}{2} * \dots * \frac{85.5\% + 86.3\%}{2} * \frac{50.9\% + 51.2\%}{2} > 2^{-10}$$

As for the 64-bit k_1 , after $v_2[k]$ is determined, we can further recover $k_1[k-1]$ by the results of experiments $m[k-1] = 0$ and $m[k-1] = 1$. Denoting $m[k-1] = 0 == l$ as S and $m[k-1] = 1 == r$ as T , we have (the same as **Table 2**):

$$\begin{aligned} P(k_1[k-1] = h_3[k-1]|S) &= P(v_3[k-1] = 0|S) = \frac{p_1}{p_1 + p_2} \\ P(k_1[k-1] = h_3[k-1]|\bar{S}) &= P(v_3[k-1] = 0|\bar{S}) = \frac{1 - p_1}{2 - p_1 - p_2} \\ P(k_1[k-1] = h_3[k-1]|T) &= P(v_3[k-1] = 1|T) = \frac{1 - p_2}{2 - p_1 - p_2} \\ P(k_1[k-1] = h_3[k-1]|\bar{T}) &= P(v_3[k-1] = 1|\bar{T}) = \frac{p_2}{p_1 + p_2} \end{aligned}$$

Suppose $p_1 > p_2$. It is seen that we should choose $k_1[k-1] = h_3[k-1]$ when S or T happens and $k_1[k-1] = \bar{h}_3[k-1]$ when \bar{S} or \bar{T} happens. If two results are contradictory, we cannot get any information about $k_1[k-1]$ because the probabilities are exactly symmetrical, which means $P(k_1[k-1] = h_3[k-1]) = 50\%$ if $S\bar{T}$ or $\bar{S}T$ happens, with probability of $p_1p_2 + (1 - p_1)(1 - p_2)$.

The main question is how to deal with $P(k_1[k-1] = h_3[k-1]|ST)$. Actually, it can hardly be calculated because the relevance of events S and T is hard to analyze. In this situation, we use the larger of $(P(k_1[k-1] = h_3[k-1]|S))$ and $(P(k_1[k-1] = h_3[k-1]|T))$ to give an approximate analysis since S and T both support $k_1[k-1] = h_3[k-1]$.

So the expected success rate of recovering $k_1[k-1]$ is:

$$(1 - p_1 - p_2 + 2p_1p_2) * \frac{1}{2} + (p_1 + p_2 - 2p_1p_2) * \max\left(\frac{\max(p_1, p_2)}{p_1 + p_2}, \frac{1 - \min(p_1, p_2)}{2 - p_1 - p_2}\right)$$

Take $k = 31$ and $v_2[k-1] = 0$ in **Table 5** as an example. If we determine $v_2[k] = 0$, we have $p_1 = 0\%$ and $p_2 = 16.8\%$, with calculated success rate 58.4%.

In the other situation with $p_1 = 97.3\%$ and $p_2 = 17.1\%$, the result is 88.0%. So the success rate in average is 73.2%. All the probabilities are given in **Appendix**.

The probability of correctly recovering all k_1 is:

$$\frac{50\% + 61.8\%}{2} * \frac{62.2\% + 86.6\%}{2} * \dots * \frac{70.1\% + 71.2\%}{2} * \frac{50.2\% + 50.3\%}{2} * \frac{1}{2} > 2^{-31}$$

In summary, the probability of recovering the 128-bit key can be increased from 2^{-128} to 2^{-41} .

6.4 Stability of Group Results

In previous sections, we optimize the result by scanning all possible boundary-lines. An apparent question is how well an optimized boundary-line will work in another data group, or whether the probability of a random key satisfying the inequation does meet the proportion. The problem is discussed in this part.

We have mentioned that the number of tested keys per group is 4096. Suppose the probability of a random key satisfying $e_j \leq bound$ is p in theory (which is hard to calculate). Then the counted proportion is actually a sample of $N(p, \sigma^2)$ with $\sigma = \sqrt{(p - p^2)/n} < 1\%$. Therefore, each entry of proportion in the tables may have a disturbance of about 2%, which may lead to 3% decrease of success rate in the worst case. In addition, we optimize the success rate by choosing the best boundary-line, while the theoretical probability is more likely to be lower than the optimization.

To examine the stability of our group results, we randomly generate other 4096 keys for each group, divide them into 8 parts ($\sigma \approx 2\%$) and test the chosen boundary-line separately. It is found that disturbances of proportions vary from 1% to 4%, according with the analysis above. Besides, the variation of success rate is no more than 2%, which can be regarded as acceptable.

Statistics of situation $k = 63$ (see in **Table 5**) also indicate the variation of success rate. Notice that the difference after first $v_2 + = v_3$ must be $1 \ll k$ under 1-bit input difference on $k = 63$ no matter what the key is (carries do not work on $k = 63$). So the success rate should be 50%. However, the optimization still reaches an increasement of about 1%, which indicates that the stability of group results restricts the disturbance of success rate to 1%.

7 Conclusion

This paper deals with truncated differential of reduced-round SipHash. In order to find out biased output bits, we study the propagation properties of output differences. As a result, we obtain some output bits with great imbalances under all possible 1-bit input differences, which can be used to distinguish SipHash-2-1 and SipHash-2-2 from a pseudorandom function with practical complexity.

Furthermore, we find a fact that the output imbalance under a certain input difference has a great relation with corresponding key bits. Based on this observation, some key bits can be recovered with high probabilities. We finally

propose a new key recovery method for SipHash-2-1, with success rate increased from 2^{-128} to 2^{-41} .

Our results do not endanger the recommended version SipHash-2-4, which is still remained indistinguishable from a pseudorandom function. But this work reveals that for the version of SipHash-2-2, some information of the key can be obtained especially when attackers have the ability to deal with higher data complexity.

References

1. Jean Philippe Aumasson and Daniel J. Bernstein. Siphash: A fast short-input prf. *Lecture Notes in Computer Science*, 7668:489–508, 2012.
2. Jean Philippe Aumasson, Simon Fischer, Shahram Khazaei, Willi Meier, and Christian Rechberger. New features of latin dances: Analysis of salsa, chacha, and rumba. In *Fast Software Encryption*, pages 470–488, 2008.
3. Amir Azodi, Marian Gawron, Andrey Sapegin, Feng Cheng, and Christoph Meinel. Leveraging event structure for adaptive machine learning on big data landscapes. In *Mobile, Secure, and Programmable Networking*, pages 28–40. Springer, 2015.
4. Thomas A. Berson. Differential cryptanalysis mod 2^{32} with applications to md5. *Lecture Notes in Computer Science*, pages 71–80, 1995.
5. Eli Biham, Rafi Chen, and Antoine Joux. Cryptanalysis of sha-0 and reduced sha-1. *Journal of Cryptology*, 28(1):110–160, 2014.
6. Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.
7. Céline Blondeau and Benoît Gérard. On the data complexity of statistical attacks against block ciphers. *Cryptology Eprint*, 2009.
8. Paul Crowley. Truncated differential cryptanalysis of five rounds of salsa20. In *SASC 2006 Stream Ciphers Revisited, 2006. 10. ECRYPT. eSTREAM, the ECRYPT Stream Cipher Project*. <http://www.ecrypt.eu.org/stream>, 2005.
9. Christoph Dobraunig, Florian Mendel, and Martin Schl affer. *Differential Cryptanalysis of SipHash*. Springer International Publishing, 2014.
10. Bob Jenkins. Spookyhash: a 128-bit noncryptographic hash, 2012.
11. Lars R. Knudsen. Truncated and higher order differentials. *Fse*, 1008:196–211, 1994.
12. Lars R. Knudsen and Thomas A. Berson. *Truncated differentials of SAFER*. Springer Berlin Heidelberg, 1996.
13. Lars R. Knudsen, M. J. B. Robshaw, and David Wagner. *Truncated Differentials and Skipjack*. Springer Berlin Heidelberg, 1999.
14. Ga etan Leurent. Analysis of differential attacks in arx constructions. *Lecture Notes in Computer Science*, 7658:226–243, 2012.
15. Ga etan Leurent. Improved differential-linear cryptanalysis of 7-round chaskey with partitioning. In *International Conference on Advances in Cryptology-eurocrypt*, 2016.
16. Mitsuru Matsui and Toshio Tokita. Cryptanalysis of a reduced version of the block cipher e2. In *International Workshop on Fast Software Encryption*, pages 71–80, 1999.
17. Shiho Moriai and Lisa Yin Yiqun. 2000-csec-10-16 cryptanalysis of twofish(ii). *Ipsj Sig Notes*, 2000(68):107–114, 2000.

18. Marc Mosko. A content-centric networking forwarding design for a network processor. In *Communications (ICC), 2015 IEEE International Conference on*, pages 5658–5664. IEEE, 2015.
19. Alfandi Omar, Bochem Arne, Kellner Ansgar, GöGe Christian, and Hogrefe Dieter. Secure and authenticated data communication in wireless sensor networks. *Sensors*, 15(8):19560–19582, 2015.
20. Geoff Pike and Jyrki Alakuijala. Introducing cityhash, 2011.
21. Won So, Ashok Narayanan, David Oran, and Mark Stapp. Named data networking on a router: forwarding at 20gbps and beyond. *Acm Sigcomm Computer Communication Review*, 43(4):495–496, 2013.
22. Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In *Proceedings of the 24th annual international conference on Theory and Applications of Cryptographic Techniques*, pages 19–35, 2005.

Appendix

The full table of our key classification experiments is shown below.

For simplicity, we use the common condition $v_2[k-1]$ to represent the determination with four groups $v_2[k] = 0 \& v_3[k-1] = 0$, $v_2[k] = 0 \& v_3[k-1] = 1$, $v_2[k] = 1 \& v_3[k-1] = 0$ and $v_2[k] = 1 \& v_3[k-1] = 1$.

Symbols p_1, q_1, p_2, q_2 denote the proportions of corresponding groups like **Table 6**. Analysis of the expected success rate (for both of k_0 and k_1) has been presented in Section 6.

Table 8. Results of key classification experiments (full table)

k	conditions	boundline	p_1	q_1	p_2	q_2	k_0	k_1
0	/	$e_{26} \leq -5.255$	0%	/	100%	/	100%	/
1	$v_2[k-1] = 0$	$e_{27} \leq -5.285$	0%	0%	100%	100%	100%	50%
1	$v_2[k-1] = 1$	$e_{30} \leq -7.635$	19.5%	51.0%	49.7%	16.7%	51.2%	61.8%
2	$v_2[k-1] = 0$	$e_{44} \leq -6.465$	0%	0.6%	99.6%	51.1%	99.6%	62.2%
2	$v_2[k-1] = 1$	$e_{44} \leq -6.535$	0.3%	98.7%	48.5%	0%	75.1%	86.6%
3	$v_2[k-1] = 0$	$e_{29} \leq -5.485$	0%	3.8%	96.4%	27.5%	96.8%	67.0%
3	$v_2[k-1] = 1$	$e_{29} \leq -5.475$	4.1%	97.1%	28.4%	0%	84.4%	79.0%
4	$v_2[k-1] = 0$	$e_{46} \leq -6.515$	0%	10.5%	97.2%	23.1%	93.7%	70.1%
4	$v_2[k-1] = 1$	$e_{46} \leq -6.525$	11.4%	96.6%	22.0%	0%	87.5%	75.4%
5	$v_2[k-1] = 0$	$e_{47} \leq -6.505$	0%	13.3%	97.6%	19.9%	92.4%	71.9%
5	$v_2[k-1] = 1$	$e_{31} \leq -5.685$	13.1%	96.3%	17.4%	0%	89.7%	73.7%
6	$v_2[k-1] = 0$	$e_{32} \leq -4.825$	0%	13.2%	96.4%	15.9%	91.9%	72.0%
6	$v_2[k-1] = 1$	$e_{32} \leq -4.825$	14.1%	96.4%	16.0%	0%	90.4%	73.2%
7	$v_2[k-1] = 0$	$e_{33} \leq -4.745$	0%	15.4%	96.5%	15.6%	90.8%	72.7%
7	$v_2[k-1] = 1$	$e_{33} \leq -4.725$	16.5%	98.0%	16.6%	0%	90.9%	73.8%
8	$v_2[k-1] = 0$	$e_{34} \leq -4.835$	0%	15.3%	96.4%	16.4%	90.9%	72.5%
8	$v_2[k-1] = 1$	$e_{34} \leq -4.845$	15.3%	96.2%	15.5%	0%	90.6%	72.6%
9	$v_2[k-1] = 0$	$e_{51} \leq -6.515$	0%	16.7%	97.3%	15.7%	90.5%	73.5%
9	$v_2[k-1] = 1$	$e_{51} \leq -6.545$	15.8%	95.7%	15.3%	0%	90.6%	72.2%
10	$v_2[k-1] = 0$	$e_{52} \leq -6.515$	0%	15.9%	97.1%	15.8%	90.8%	73.2%
10	$v_2[k-1] = 1$	$e_{52} \leq -6.505$	16.5%	97.2%	16.3%	0%	90.7%	73.2%
11	$v_2[k-1] = 0$	$e_{53} \leq -6.515$	0%	16.2%	97.2%	17.9%	90.8%	72.8%
11	$v_2[k-1] = 1$	$e_{53} \leq -6.535$	16.4%	96.3%	15.6%	0%	90.7%	72.5%
12	$v_2[k-1] = 0$	$e_{54} \leq -6.525$	0%	14.8%	96.4%	16.5%	91.1%	72.3%
12	$v_2[k-1] = 1$	$e_{54} \leq -6.565$	14.8%	94.9%	15.0%	0%	90.3%	71.9%
13	$v_2[k-1] = 0$	$e_{55} \leq -6.515$	0%	18.0%	97.0%	16.3%	89.8%	73.5%
13	$v_2[k-1] = 1$	$e_{55} \leq -6.485$	17.9%	97.7%	17.5%	0%	90.3%	73.4%
14	$v_2[k-1] = 0$	$e_{56} \leq -6.565$	0%	14.5%	94.4%	13.8%	90.3%	71.6%
14	$v_2[k-1] = 1$	$e_{56} \leq -6.515$	16.2%	96.8%	17.3%	0%	90.0%	73.2%
15	$v_2[k-1] = 0$	$e_{57} \leq -6.525$	0%	15.9%	96.5%	15.5%	90.6%	72.9%
15	$v_2[k-1] = 1$	$e_{57} \leq -6.505$	17.4%	97.2%	17.0%	0%	90.3%	73.1%
16	$v_2[k-1] = 0$	$e_{42} \leq -5.585$	0%	14.4%	96.3%	14.9%	91.2%	72.5%
16	$v_2[k-1] = 1$	$e_{42} \leq -5.565$	16.2%	97.1%	15.5%	0%	91.0%	73.0%

17	$v_2[k-1] = 0$	$e_{43} \leq -5.575$	0%	13.8%	95.6%	16.0%	91.2%	71.7%
17	$v_2[k-1] = 1$	$e_{59} \leq -6.545$	15.3%	96.3%	15.0%	0%	90.9%	72.6%
18	$v_2[k-1] = 0$	$e_{60} \leq -6.515$	0%	16.8%	96.5%	16.8%	90.2%	72.8%
18	$v_2[k-1] = 1$	$e_{60} \leq -6.535$	15.7%	96.6%	15.9%	0%	90.6%	72.9%
19	$v_2[k-1] = 0$	$e_{61} \leq -6.515$	0%	16.5%	96.9%	17.5%	90.5%	72.8%
19	$v_2[k-1] = 1$	$e_{61} \leq -6.535$	16.5%	96.3%	15.7%	0%	90.6%	72.5%
20	$v_2[k-1] = 0$	$e_{62} \leq -6.535$	0%	16.2%	96.5%	15.6%	90.4%	72.9%
20	$v_2[k-1] = 1$	$e_{46} \leq -5.485$	15.1%	96.5%	16.0%	0%	90.5%	73.0%
21	$v_2[k-1] = 0$	$e_{63} \leq -6.515$	0%	16.4%	97.2%	16.5%	90.6%	73.2%
21	$v_2[k-1] = 1$	$e_{47} \leq -5.455$	16.0%	97.3%	15.7%	0%	91.0%	73.2%
22	$v_2[k-1] = 0$	$e_{48} \leq -5.465$	0%	16.4%	96.6%	16.7%	90.4%	72.8%
22	$v_2[k-1] = 1$	$e_{48} \leq -5.445$	17.7%	97.4%	18.1%	0%	89.9%	73.4%
23	$v_2[k-1] = 0$	$e_{49} \leq -5.455$	0%	17.5%	96.8%	17.7%	89.9%	72.9%
23	$v_2[k-1] = 1$	$e_{49} \leq -5.475$	16.0%	96.0%	16.4%	0%	90.1%	72.6%
24	$v_2[k-1] = 0$	$e_{50} \leq -5.485$	0%	14.1%	96.1%	15.7%	91.3%	72.1%
24	$v_2[k-1] = 1$	$e_{50} \leq -5.465$	15.5%	96.9%	16.3%	0%	90.5%	73.2%
25	$v_2[k-1] = 0$	$e_{51} \leq -5.455$	0%	16.0%	97.2%	16.7%	90.8%	73.0%
25	$v_2[k-1] = 1$	$e_{51} \leq -5.465$	16.4%	96.9%	16.2%	0%	90.6%	73.0%
26	$v_2[k-1] = 0$	$e_3 \leq -8.635$	0%	8.3%	98.2%	8.8%	95.0%	73.6%
26	$v_2[k-1] = 1$	$e_3 \leq -8.605$	9.2%	98.6%	8.7%	0%	95.0%	73.9%
27	$v_2[k-1] = 0$	$e_5 \leq -6.515$	0%	15.7%	96.1%	16.9%	90.5%	72.3%
27	$v_2[k-1] = 1$	$e_5 \leq -6.525$	15.8%	96.5%	14.8%	0%	91.1%	72.6%
28	$v_2[k-1] = 0$	$e_{17} \leq -8.995$	0%	15.9%	100%	15.9%	92.0%	75.0%
28	$v_2[k-1] = 1$	$e_{17} \leq -8.995$	16.5%	100%	17.5%	0%	91.2%	75.3%
29	$v_2[k-1] = 0$	$e_7 \leq -6.525$	0%	16.2%	96.8%	15.6%	90.6%	73.1%
29	$v_2[k-1] = 1$	$e_7 \leq -6.535$	16.3%	96.1%	15.8%	0%	90.5%	72.4%
30	$v_2[k-1] = 0$	$e_8 \leq -6.515$	0%	15.7%	97.3%	15.7%	91.0%	73.3%
30	$v_2[k-1] = 1$	$e_8 \leq -6.535$	16.9%	96.3%	15.6%	0%	90.7%	72.3%
31	$v_2[k-1] = 0$	$e_9 \leq -6.505$	0%	16.8%	97.3%	17.1%	90.5%	73.2%
31	$v_2[k-1] = 1$	$e_9 \leq -6.525$	15.8%	96.3%	16.3%	0%	90.3%	72.8%
32	$v_2[k-1] = 0$	$e_{58} \leq -5.485$	0%	15.6%	96.6%	16.5%	90.8%	72.6%
32	$v_2[k-1] = 1$	$e_{10} \leq -6.515$	16.6%	97.4%	16.4%	0%	90.7%	73.3%
33	$v_2[k-1] = 0$	$e_{59} \leq -5.495$	0%	14.7%	96.3%	16.0%	91.1%	72.3%
33	$v_2[k-1] = 1$	$e_{11} \leq -6.525$	16.7%	96.7%	16.3%	0%	90.5%	72.8%
34	$v_2[k-1] = 0$	$e_{12} \leq -6.535$	0%	15.6%	96.8%	16.0%	90.8%	72.8%
34	$v_2[k-1] = 1$	$e_{60} \leq -5.485$	17.0%	96.6%	15.9%	0%	90.7%	72.6%
35	$v_2[k-1] = 0$	$e_{61} \leq -5.495$	0%	15.0%	96.2%	15.4%	90.9%	72.5%
35	$v_2[k-1] = 1$	$e_{61} \leq -5.495$	14.8%	96.0%	16.0%	0%	90.3%	72.8%
36	$v_2[k-1] = 0$	$e_{14} \leq -6.555$	0%	15.3%	95.6%	15.7%	90.5%	72.1%
36	$v_2[k-1] = 1$	$e_{14} \leq -6.525$	17.6%	96.9%	17.8%	0%	89.8%	73.1%
37	$v_2[k-1] = 0$	$e_{15} \leq -6.565$	0%	14.9%	95.3%	15.1%	90.6%	72.0%
37	$v_2[k-1] = 1$	$e_{63} \leq -5.485$	16.3%	96.4%	15.9%	0%	90.5%	72.6%
38	$v_2[k-1] = 0$	$e_{16} \leq -6.535$	0%	16.3%	97.0%	16.0%	90.6%	73.1%
38	$v_2[k-1] = 1$	$e_{16} \leq -6.565$	14.6%	96.3%	15.1%	0%	90.9%	72.8%
39	$v_2[k-1] = 0$	$e_{16} \leq -8.995$	0%	19.4%	93.0%	20.3%	87.5%	70.7%
39	$v_2[k-1] = 1$	$e_{16} \leq -8.985$	18.7%	94.0%	19.1%	0%	88.0%	71.5%
40	$v_2[k-1] = 0$	$e_2 \leq -3.925$	0%	23.9%	90.8%	24.1%	84.5%	69.9%

40	$v_2[k-1] = 1$	$e_{17} \leq -6.155$	23.4%	90.4%	22.4%	0%	85.1%	69.4%
41	$v_2[k-1] = 0$	$e_3 \leq -4.195$	0%	21.3%	93.3%	20.7%	86.7%	71.3%
41	$v_2[k-1] = 1$	$e_3 \leq -4.185$	21.1%	93.3%	21.9%	0%	86.4%	71.3%
42	$v_2[k-1] = 0$	$e_{19} \leq -8.995$	0%	5.1%	99.4%	4.9%	97.2%	74.6%
42	$v_2[k-1] = 1$	$e_{19} \leq -8.965$	5.2%	99.6%	5.4%	0%	97.1%	74.8%
43	$v_2[k-1] = 0$	$e_{21} \leq -5.995$	0%	12.8%	96.6%	13.1%	92.2%	72.7%
43	$v_2[k-1] = 1$	$e_{21} \leq -5.955$	13.8%	97.7%	13.7%	0%	92.1%	73.4%
44	$v_2[k-1] = 0$	$e_{22} \leq -5.815$	0%	12.4%	98.6%	12.2%	93.2%	74.1%
44	$v_2[k-1] = 1$	$e_{22} \leq -5.855$	11.1%	96.5%	11.4%	0%	92.7%	72.8%
45	$v_2[k-1] = 0$	$e_{23} \leq -5.765$	0%	12.3%	98.1%	14.0%	93.0%	73.3%
45	$v_2[k-1] = 1$	$e_{23} \leq -5.775$	12.3%	97.9%	13.3%	0%	92.4%	73.9%
46	$v_2[k-1] = 0$	$e_8 \leq -5.415$	0%	14.0%	97.0%	15.2%	91.7%	72.8%
46	$v_2[k-1] = 1$	$e_8 \leq -5.415$	14.4%	97.5%	14.5%	0%	91.7%	73.4%
47	$v_2[k-1] = 0$	$e_{25} \leq -6.165$	0%	9.4%	98.7%	9.5%	94.7%	74.1%
47	$v_2[k-1] = 1$	$e_{25} \leq -6.135$	10.7%	99.4%	10.5%	0%	94.5%	74.5%
48	$v_2[k-1] = 0$	$e_{10} \leq -5.435$	0%	20.2%	96.2%	20.5%	88.4%	72.6%
48	$v_2[k-1] = 1$	$e_{10} \leq -5.445$	19.6%	95.2%	18.7%	0%	88.7%	71.8%
49	$v_2[k-1] = 0$	$e_{11} \leq -5.495$	0%	15.9%	94.8%	16.1%	89.9%	71.7%
49	$v_2[k-1] = 1$	$e_{11} \leq -5.475$	16.6%	96.0%	16.2%	0%	90.2%	72.4%
50	$v_2[k-1] = 0$	$e_{12} \leq -5.495$	0%	16.3%	96.2%	15.1%	90.3%	72.9%
50	$v_2[k-1] = 1$	$e_{12} \leq -5.505$	15.1%	95.6%	15.6%	0%	90.3%	72.3%
51	$v_2[k-1] = 0$	$e_{29} \leq -6.535$	0%	15.8%	96.4%	15.7%	90.6%	72.7%
51	$v_2[k-1] = 1$	$e_{13} \leq -5.495$	15.2%	95.7%	15.7%	0%	90.3%	72.4%
52	$v_2[k-1] = 0$	$e_{30} \leq -6.535$	0%	16.3%	95.9%	16.1%	90.1%	72.5%
52	$v_2[k-1] = 1$	$e_{14} \leq -5.495$	15.1%	96.1%	15.6%	0%	90.6%	72.7%
53	$v_2[k-1] = 0$	$e_{31} \leq -6.555$	0%	14.9%	95.1%	15.8%	90.5%	71.7%
53	$v_2[k-1] = 1$	$e_{31} \leq -6.495$	17.4%	97.9%	18.3%	0%	90.0%	73.8%
54	$v_2[k-1] = 0$	$e_{32} \leq -6.375$	0%	16.3%	97.2%	17.2%	90.7%	73.0%
54	$v_2[k-1] = 1$	$e_{16} \leq -5.515$	13.9%	94.6%	14.1%	0%	90.6%	71.6%
55	$v_2[k-1] = 0$	$e_{17} \leq -5.285$	0%	16.6%	97.9%	16.1%	90.8%	73.8%
55	$v_2[k-1] = 1$	$e_{17} \leq -5.315$	14.2%	96.6%	15.4%	0%	90.8%	73.1%
56	$v_2[k-1] = 0$	$e_{18} \leq -5.255$	0%	18.1%	98.2%	14.7%	90.2%	74.6%
56	$v_2[k-1] = 1$	$e_{18} \leq -5.275$	15.6%	96.7%	15.1%	0%	91.1%	72.8%
57	$v_2[k-1] = 0$	$e_{19} \leq -5.255$	0%	15.1%	97.5%	13.8%	91.4%	73.7%
57	$v_2[k-1] = 1$	$e_{19} \leq -5.265$	15.6%	96.5%	14.3%	0%	91.4%	72.4%
58	$v_2[k-1] = 0$	$e_{20} \leq -5.275$	0%	13.8%	94.6%	13.2%	90.8%	71.8%
58	$v_2[k-1] = 1$	$e_{20} \leq -5.235$	15.2%	97.7%	15.9%	0%	91.1%	73.7%
59	$v_2[k-1] = 0$	$e_{37} \leq -5.815$	0%	14.9%	97.4%	14.3%	91.4%	73.4%
59	$v_2[k-1] = 1$	$e_{37} \leq -5.835$	13.5%	96.4%	14.7%	0%	91.1%	73.0%
60	$v_2[k-1] = 0$	$e_{37} \leq -7.985$	0%	11.0%	95.6%	10.8%	92.5%	72.1%
60	$v_2[k-1] = 1$	$e_{37} \leq -7.925$	13.3%	97.4%	13.6%	0%	92.1%	73.4%
61	$v_2[k-1] = 0$	$e_{39} \leq -5.545$	0%	14.1%	98.4%	12.8%	92.2%	74.2%
61	$v_2[k-1] = 1$	$e_{39} \leq -5.535$	14.7%	98.2%	13.0%	0%	92.7%	73.4%
62	$v_2[k-1] = 0$	$e_{39} \leq -7.085$	0%	21.7%	90.8%	20.6%	85.5%	70.1%
62	$v_2[k-1] = 1$	$e_{39} \leq -7.065$	21.1%	93.1%	21.9%	0%	86.3%	71.2%
63	$v_2[k-1] = 0$	$e_{41} \leq -3.935$	88.8%	88.8%	90.3%	89.3%	50.9%	50.2%
63	$v_2[k-1] = 1$	$e_{44} \leq -5.095$	28.3%	29.1%	29.9%	30.9%	51.2%	50.3%