Quantum Multi-Collision Distinguishers

Zhenzhen Bao[1], Jian Guo[1], Shun Li[1,2], and Phuong Pham[1]

[1] Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore.
{zzbao,guojian}@ntu.edu.sg, pham0079@e.ntu.edu.sg
[2] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China.lishun93@sjtu.edu.cn

Abstract. In EUROCRYPT 2020, Hosoyamada and Sasaki find differential paths with probability $2^{-2n/3}$ can be useful in quantum collision attacks, v.s. $2^{-n/2}$ for classical collision attacks. This observation led to attacks for more rounds on some AES-like hash functions. In this paper, we quantize the multi-collision distinguisher proposed by Biryukov, Khovratovich, and Nikolić at CRYPTO 2009, and propose quantum multi-collision distinguishers. Compared against the tight bound $2^{\frac{n}{2}\cdot(1-\frac{1}{2^q-1})}$ for quantum multi-collision on ideal functions by Liu and Zhang in EUROCRYPT 2019, we find the probability of useful differential paths can be as low as 2^{-n}. This leads to even more attacked rounds than both classical multi-collision distinguishers and quantum collision attacks. To demonstrate the effectiveness, we applied the attack model to AES, Rijndael, and the post-quantum block cipher design Saturnin. Distinguishing attacks are found on the full version of AES-192, AES-256, Rijndael-128-160, and Rijndael-128-224. Other results include 8-round AES-128, 11-round Rijndael-160-192, 12-round Rijndael-160-256, and 10-round Saturnin-256.

Keywords: post-quantum cryptography, multicollision, free variable, BHT, related-key differential trail, distinguisher

1 Introduction

Recently, post-quantum security of cryptographic systems and primitives has received more and more attention from cryptographic researchers, developers, and users due to the progress in the development of quantum computers. The security of *public-key* crypto-systems such as RSA, DSA, and elliptic curve can often be reduced to some mathematically difficult problems such as factoring and discrete logarithm. However, Shor's seminal work [46] could be used to solve both problems efficiently in a sufficiently large quantum computer, which directly destroys the security of the public-key cryptographic schemes based on them in the post-quantum world. Due to such concerns, researchers have begun to investigate and develop post-quantum cryptographic algorithms, serving as replacements of the current public-key crypto-systems, with security against attackers aided by both quantum and classical computers. In the meanwhile, NIST has initiated a competition to develop post-quantum standards for key

establishment schemes and digital signature schemes since 2017 [39] . On the other hand, *symmetric-key* crypto-systems usually are not built upon the security assumption of hard math problems due to performance needs, and the research on how quantum computers would affect their security strength shows to be less direct. In 1996, Grover [16] found quantum algorithms could be faster for database search than in classical settings. This algorithm could be used for *bruteforce* search in time $\sqrt{N}$ for a space of size N, due to which halved (in bits) security strength is now considered as the *generic* lower security bound of a classical symmetric-key primitive in the quantum setting. Besides, recent studies showed that there exist non-trivial quantum attacks other than direct Grover search. In 2010, Kuwakado and Morii [29] used SIMON's algorithm [47] to distinguish the 3-round Feistel scheme from a random permutation in the quantum setting. After that, SIMON's algorithm has been applied to other symmetric-key schemes such as Even-Mansour scheme [28], message authentication codes (MACs) [25], and FX construction [31]. Invented in 1997, SIMON's algorithm allows to find a "hidden period" with only polynomially many queries and time. In most of the previous works utilizing SIMON's algorithm, the attacker tries to construct a function in such a way that the existence of the hidden period depends on the key values, which can be recovered once the period is detected. The price of such attacks is the strong requirement to access quantum superposition oracle of the keyed primitives, which directly affects the practical relevance of SIMON's algorithm. In addition to SIMON's algorithm, collision-finding utilizing Grover search is another prominent approach as dedicated attacks against symmetric-key primitives in quantum setting.

1.1 Collision

Preimage, second-preimage, and collision resistance form the basic security requirements for a hash function in the classical setting, and the same is expected in the quantum setting, *e.g.*, some public-key schemes have been proven to be post-quantum secure in the quantum random oracle model (QROM) [4] when instantiated with a post-quantum secure hash function. The known generic best time bounds in the quantum setting so far are $n/2$ bits for preimage resistance due to Grover's, and $n/3$ bits for collision resistance due to the BHT algorithm [6] named after Brassard, Høyer, and Tapp in 1998.

The BHT algorithm finds collisions with a query complexity of $O(2^{n/3})$ and $O(2^{n/3})$-qubit quantum random access memory (qRAM), which is a quantum analogue of the random access memory (RAM) allowing to efficiently access data in quantum superpositions. In 2017, Chailloux, Naya-Plasencia, and Schrottenloher [8] proposed the CNS collision finding algorithm with a time complexity of $O(2^{2n/5})$, a quantum memory of $O(n)$ qubits, and a classical memory of $O(2^{n/5})$ bits. The complexities of both BHT and CNS algorithms are optimized towards lowest possible time. When it comes to time-memory tradeoff with the merit of $T \times M$, the simple Grover search achieves the best $2^{n/2}$ (although this is not proven) with $O(2^{n/2})$ time and $O(1)$ memory, while BHT gives $2^{n/3} \cdot 2^{n/3} = 2^{2n/3}$ and CNS $2^{2n/5} \cdot 2^{n/5} = 2^{3n/5}$. Memoryless version of birthday attack for collision

finding in classical setting is offered by the Pollard's rho method [43] in 1975, and an extension for parallelism was given by Van Oorschot and Wiener [50] in 1999.

The above mentioned algorithms are generic and do not exploit any internal characteristics of the primitives. The first dedicated quantum collision attack on hash functions was proposed at EUROCRYPT 2020 by Hosoyamada and Sasaki in [20], which shows differentials whose probability was too low for classical collision search can become useful in the quantum setting. They applied a quantum version of the rebound attack [33] to round-reduced AES hashing modes and Whirlpool, and extended the number of attacked rounds for collision finding from 6 and 5 rounds in classical setting to 7 and 6 rounds in quantum setting, respectively. These collision finding algorithms are considered as *attacks* when they require lesser time and memory than that of BHT algorithm ($T = M = O(2^{n/3})$. Later, Dong *et al.* [12] followed the CNS algorithm and presented an improved the quantum rebound attacks on AES hashing modes and Grøstl-512 in a setting, where only a small amount of qRAM is available and the required resources are lesser than that of CNS ($T = O(2^{2n/5})$, $qM = O(n)$, $cM = O(2^{n/5})$). Very recently, Hosoyamada and Sasaki [21] proposed the first dedicated quantum collision attacks on SHA-256 and SHA-512 in another setting where the efficiency is evaluated by the time-memory tradeoff compared against $O(2^{n/2})$ by Pollard's rho.

1.2 Multi-Collision

Multi-Collision (or multicollision) finding is an important problem arising from the generalization of collision finding. A q-multicollision for a function H is a set of q distinct inputs $\{x_1, \ldots, x_q\}$ leading to the same output, *i.e.*, $H(x_i) = H(x_j)$ for all $1 \leq i, j \leq q$. Multi-Collisions appear to be useful in the cryptanalysis against hash functions, block ciphers, and other cryptographic primitives. The notion of multicollisions was first introduced by Merkle in [34] to analyze the security of a hash function based on DES. It was later used against MicroMint [45], RMAC [22], chopMD [9], Leamnta-LW [19], PHOTON and Parazoa [37], and the Keyed-Sponge [24], all of which assume the multi-Collision resistance of the underlying function. In 2004, Joux [23] proposed the multicollision attack on iterated hash functions, where a 2^q-multicollision can be found at the computational cost of q collisions. In 2009, Knudsen *et al.* [27] presented a faster method to find multicollisions using preimage attacks. Then, Naito *et al.* [38] gave generic state-recovery and forgery attacks on MACs in 2013. For block ciphers, Biryukov *et al.* [2] introduced the notation of q-multicollisions, by which they mounted a distinguishing attack against the full AES-256 in a chosen-key setting. In 2013, Nikolić *et al.* [41] proposed a cryptanalysis of round-reduced version of the lightweight block cipher LED, and in the year after, Itai *et al.* [11] presented a cryptanalysis against iterated Even-Mansour schemes with two keys, all of which utilize multi-Collisions as the core of the attacks.

When the underlying is an ideal random function, Suzuki *et al.* [49] shows the number of function evaluations needed to find one q-multicollision is $(q!)^{1/q} \cdot N^{(q-1)/q} \approx \frac{q}{e} \cdot N^{(q-1)/q}$, where N denotes the size of co-domain and e is the base

of natural logarithm. In [2], Biryukov, Khovratovich, and Nikolić considered a variant of this problem, where the underlying function is the XOR difference of two ciphertexts of the same block cipher queried under a chosen-key setting. The query complexity is proven to be at least $q \cdot N^{\frac{q-1}{q+1}}$ when q is small and $q \cdot N^{\frac{q-2}{q+2}}$ otherwise. This model is applied to distinguish the full AES-256, where a high probability differential path in related-key setting was found and the function is defined to be the ciphertext difference following the differential path. Then the q-multicollision is found with a query complexity of $q \cdot 2^{67}$, which is lower than $q \cdot 2^{\frac{q-1}{q+1} \cdot 128}$ in the ideal case with small q and $N = 2^{128}$ for AES.

1.3 Quantum Multi-Collision

The generic bound of collision resistance in classical setting is $2^{n/2}$ due to birthday attack, hence a differential based collision finding algorithm constitutes an attack only if the the probability of the underlying differential path of the bruteforce search phase is higher than $2^{-n/2}$. Hosoyamada and Sasaki [20] observed that, while in the quantum setting, the generic time bound is $2^{n/3}$ due to BHT algorithm, and the admissible differential probability can be as low as $2^{-2n/3}$, for which the bruteforce search of the conforming pair costs time $2^{n/3}$ and negligible quantum or classical memory by Grover's search in quantum computers. Taking advantage of this gap in the admissible probability ($2^{-2n/3}$ in quantum v.s. $2^{-n/2}$ in classical setting), differential paths with lower probability, but for more rounds, become useful hence lead to collision attacks for more rounds in quantum setting. This gap was further enlarged by considering higher time bound in CNS algorithm in [12], with $T = O(2^{2n/5})$ and admissible probability $2^{-4n/5}$, and time-memory tradeoff in [21] with $T = O(2^{n/2})/S$ and admissible probability $2^{-n} \cdot S^2$ when S qubits are needed to implement the attack in quantum circuit or for qRAM, as summarized in Table 1.

collision algo.		Time	q-Mem	c-Mem	Prob.	Reference	Acronym
Classical		$2^{n/2}$	-	1 or $2^{n/2}$	$2^{-n/2}$	bir., [43]	CCB
Quantum	BHT [6]	$2^{n/3}$	$2^{n/3}$	1	$2^{-2n/3}$	[20]	QCB1
	CNS [8]	$2^{2n/5}$	n	$2^{n/5}$	$2^{-4n/5}$	[12]	QCB2
	rho [43]	$2^{n/2}/S$	S	1	$2^{-n} \cdot S^2$	[21]	QCB3

Table 1: Comparison of the asymptotic complexities in Big-O notations of collision finding algorithms and the lowest admissible probabilities of differential path in quantum setting calculated by Prob. = Time^{-2}. Here the quantum memory (q-Mem) includes both the quantum circuit needed to implement the computation of the underlying attack, and the qRAM for storage and access of data.

Motivated by [20], in this paper we consider the problem of q-multicollision finding in the quantum setting, which is a natural generalization of the collision finding problem. And similarly to [20], we also consider the scenario where bruteforce (resp. Grover search in quantum setting) is used to find conforming

pairs of a given differential path. When the search is limited by time T, the admissible differential probability is at least T^{-1} (resp. T^{-2} in quantum). In 2019, Liu and Zhandry [32] proved that the necessary and sufficient query complexity (hence tight bound) for the quantum q-multicollision problem is $N^{\frac{1}{2}\cdot(1-\frac{1}{2^q-1})}$ aided by the same amount of qRAM. Following the $N^{(q-1)/q}$ bound by Suzuki *et al.* (after removing the polynomial factors), the admissible probability is $N^{\frac{1}{q}-1}$ in classical v.s. $N^{\frac{1}{2^q-1}-1}$ in quantum setting, which exhibits a similar gap as for the differential based collision attacks.

1.4 Our Contributions

Following the observation on the gap of admissible probabilities, in this paper we propose the Quantum Multi-Collision (QMC) distinguisher, as quantized version of the q-multicollision distinguisher proposed in [2]. Our model shows differentials with probability as low as 2^{-n}, for a block cipher with n-bit block size, will be useful in mounting QMC attack, compared with $2^{-2n/3}$ and $2^{-4n/5}$ for quantum collision attack considered in [20] and [12], respectively. This wider range of admissible probability potentially leads to larger number of attacked rounds. When applied to AES-like block cipher, automatic search tools based on [17] are utilized to find differential characteristics with highest possible probabilities under the related-key setting, then the triangulation algorithm proposed in [26] is applied to fulfill as many S-boxes as possible unitizing the degrees of freedom from both key and state variables, which further reduces the overall attack complexities. Equipped with these powerful tools, we apply the attack framework to AES, Rijndael, and Saturnin, and find a rich set of results including full version of AES-192, AES-256, Rijndael-128-160, and Rijndael-128-224, and 8-round AES-128, 11-round Rijndael-160-192, 12-round Rijndael-160-256, and 10-round Saturnin-256, as summarized in Table 2. These results can be interpreted in the following three ways.

- Compared with other distinguishing attacks, our results reach the largest number of rounds on most of the targets including AES-192, AES-256, Rijndael-128-160, Rijndael-128-224, Rijndael-160-192, Rijndael-160-256, and Saturnin-256.
- Compared with the classical multi-collision distinguisher, both accept admissible probability 2^{-n}, but for different q values. If the attacks are valid for $q = 3$, they are valid also for all other q values. When $q = 3$ is set as the threshold, QMC distinguishers can cover more rounds than Classical Multi-Collision (CMC) distinguishers, *e.g.*, 8 v.s. 6 for AES-128, 10 v.s. 9 for Rijndael-128-160, 13 v.s. 11 for Rijndael-128-224, and 8 v.s. 7 for Saturnin-256.
- Compared with the quantum collision attacks, QMC distinguisher also covers more rounds, *e.g.*, 8 v.s. 7 for AES-128, while the data for Quantum Collision (QC) attacks are not available for comparison on other targets.

Organization. Section 2 gives a brief introduction of AES-like primitives, quantum computation, qRAMs, and quantum adversary models. Section 3 introduces

Target	#B,#R	Rounds	Attack	Time	Mem	Ref.	Notes
AES-128	128,10	6	CC	2^{56}	2^{32}	[15, 30]	CCB
			CMC	2^{36}	-	[48],Sect. 5.1	
		7	QC	$2^{45.4}$	2^{16}(qRAM)	[12]	QCB1
			QC	$2^{45.8}$	-	[12]	QCB2
			QC	$2^{42.5}$	2^{48}(qRAM)	[20]	QCB1
			QC	$2^{59.5}$	-	[20]	QCB2
			CMC	$q \cdot 2^{90}$	-	Sect. 5.1	$q \geq 4$
			QMC	$q \cdot 2^{45}$	-	Sect. 5.1	
		8	QMC	$q \cdot 2^{52.5}$	2^{16}(qRAM)	Sect. 5.1	
			QMC	$q \cdot 2^{56}$	-	Sect. 5.1	$q \geq 4$
AES-192	128,12	10	CMC	$q \cdot 2^{78}$	-	[17],Sect. 5.1	
		12	r.k. rec	2^{176}	2^{152} (data 2^{123})	[1]	
		12	QMC	$q \cdot 2^{51}$	-	Sect. 5.1	
AES-256	128,14	**14**	CMC	$q \cdot 2^{67}$	-	[2]	
		14	QMC	$q \cdot 2^{33}$	-	[17],Sect. 5.1	
Rijndael-128-160	128,11	9	CMC	$q \cdot 2^{60}$	-	Sect. 5.2	
		10	QMC	$q \cdot 2^{45}$	-	Sect. 5.2	
		11	QMC	$q \cdot 2^{59}$	-	Sect. 5.2	$q \geq 4$
Rijndael-128-224	128,13	11	CMC	$q \cdot 2^{67}$	-	Sect. 5.2	
		13	QMC	$q \cdot 2^{45.5}$	-	Sect. 5.2	
Rijndael-160-192	160,12	11	QMC	$q \cdot 2^{45}$	-	Sect. 5.2	
		11	CMC	$q \cdot 2^{90}$	-	Sect. 5.2	
Rijndael-160-256	160,14	12	QMC	$q \cdot 2^{51}$	-	Sect. 5.2	
		12	CMC	$q \cdot 2^{102}$	-	Sect. 5.2	
SATURNIN-256	256,16	7	CMC	$q \cdot 2^{143}$	-	Sect. 5.3	
		8	QMC	$q \cdot 2^{85.6}$	-	Sect. 5.3	
		10	r.k. rec	2^{236}	2^{128} (data 2^{236})	[7]	
		10	QMC	$q \cdot 2^{124.65}$	-	Sect. 5.3	$q \geq 6$

Table 2: Summary of results on quantum multi-collision distinguishers against AES, Rijndael, and SATURNIN. Hereafter, CC is classical collision attack; QC is quantum collision attack; CMC is classical multi-collision attack; QMC is quantum multi-collision attack. The range of q values to make the corresponding attack valid is listed in the last "Notes" column, and left empty if the attack is valid for all $q \geq 3$. Early-abortion technique for bruteforce may help to reduce the time complexity of all our QMC and CMC attacks listed here by a factor around 2^8, as noted in [20], which are not included in this table for simplicity.

the related quantum collision algorithms and quantum multi-collision algorithms for ideal functions. Then, Section 4 give the our attack framework and techniques involved, followed by is followed by applications to AES-128, Rijndael, and SATURNIN in Section 5. Section 6 concludes the paper. Some details of the work are postponed to Appendix.

2 Preliminary

Before moving onto the description of detailed attacks, in this section we give a brief introduction of the necessary preliminaries to understand better quantum computations, quantum (multi-)collision algorithms, and quantum memories.

2.1 Quantum Computation and Quantum RAM

Similar to the time complexity estimation on classical computers, the unit of time complexity on quantum computers refers to the computational effort required to execute the underlying primitive once. The actual time to run a quantum attack will depend on many factors including the hardware architectures of quantum computers. In what follows, we consider the simple computational model that each pair of qubits in a quantum computer can interact with one another. Based on this model, the time complexity of dedicated algorithms is evaluated and compared against the generic bounds under the same model. Such algorithms are only considered as *valid attacks* if they require lesser resources like time and/or space than the generic bounds. Here, space complexity refers to the number of qubits to implement the attack, and similarly that needed to implement the underlying primitive is one *unit* of space.

Random-access memory (RAM) is a form of computer memory that supports read and write in any order, and the access time is often assumed to be constant in the time complexity evaluation of cryptanalysis. Quantum random-access memory (qRAM) is the quantum analog of the RAM, which supports data access and computation in superpositions. For simplicity of complexity evaluation, we assume similarly to RAM that access time of qRAM is constant, and that for reading or writing of one cell is considered as a unit. Furthermore, we do not distinguish qubits used as memory like qRAM and the qubits used for quantum circuit implementations of a function.

2.2 Grover's algorithm

Given a search space of N elements $\{1, 2, \ldots, N\}$, a Boolean function $f : \{1, 2, \ldots, N\} \to \{0, 1\}$, and $a \triangleq |f^{-1}(1)|/N$ the probability for a random x resulting in $f(x) = 1$, the best classical algorithm with black-box access to f requires $1/a$ queries in order to find one x with $f(x) = 1$ for a probability more than 0.5. This is usually referred to as the bruteforce search in the classical setting. However, in the quantum setting with quantum black-box oracle access to f, Grover's algorithm finds x with $\Theta(\sqrt{1/a})$ quantum queries to the quantum oracle O_f, which is defined as:

$$|x\rangle |y\rangle \mapsto |x\rangle |y \oplus f(x)\rangle .$$

Starting with a uniform superposition $|\phi\rangle = \frac{1}{\sqrt{N}} \sum_{x=1}^{N} |x\rangle$, by applying the Hadamard transformation $H^{\otimes n}$ to $|0\rangle^{\otimes n}$ where $n = \log_2^N$, the Grover's algorithm iteratively applies the unitary transformation $(2 |\phi\rangle \langle\phi| - I)O_f$ to $|\phi\rangle$ such that the amplitudes of those x's with $f(x) = 1$ are amplified. When measuring the resulting state, a

value of x of interest will be returned with overwhelming probability. Due to this nature, Grover's is also viewed as quantum analog of bruteforce search.

Similar to classical bruteforce search, the time complexity of Grover's algorithm is usually dominated and approximated by the number of function evaluations. The quadratic speedup of the search ($\sqrt{1/a}$ v.s. $1/a$) is based on the premise that the oracle circuit can be constructed efficiently. Thus, it is important to have a precise estimate of the resources needed to implement the quantum oracle *e.g.*,, the amount of qRAM. Given a list of classical data $L = (x_0, \ldots, x_{2^n-1})$ with $x_i \in \mathbb{F}_2^m$, the qRAM for L is modeled as an unitary transformation O_{qRAM}^L such that

$$O_{qRAM}^L : |i\rangle_{Addr} \otimes |y\rangle_{Out} \mapsto |i\rangle_{Addr} \otimes |y \oplus x_i\rangle_{Out},$$

where $i \in \mathbb{F}_2^n, y \in \mathbb{F}_2^m$, and $|\cdot\rangle_{Addr}$ and $|\cdot\rangle_{Out}$ may be regarded as the address and output registers, respectively. In such a way, quantum superposition of the memory cells can be accessed by the corresponding superposition of addresses. When qRAM is available, a quantum gate that realizes the unitary operation is available in addition to basic quantum gates.

2.3 Quantum and Classical Collision-Finding Algorithms

The BHT Algorithm. Developed by Brassard, Høyer, and Tapp [6], the BHT algorithm finds a collision in time $O(2^{n/3})$ by making $O(2^{n/3})$ quantum queries when $O(2^{n/3})$ qRAM is available. BHT algorithm consists of two steps. The first step performs a classical precomputation that chooses a subset $X \subseteq \{0,1\}^n$ of size $|X| = 2^{n/3}$ and computes the value $f(x)$ for all $x \in X$. The list of $2^{n/3}$ pairs $L = \{(x, f(x))\}_{x \in X}$ are stored into qRAM so that they can be accessed in quantum superpositions. Then, the second step performs the Grover search to find $x' \in \{0,1\}^n \backslash X$ such that $f(x') = f(x)$ for some $x \in X$ by comparing the value of $f(x')$ against the stored list L, which takes time $O(\sqrt{2^n/|L|}) = O(2^{n/3})$ on average. The overall optimal time complexity is $O(2^{n/3})$ for both steps, and memory requirement is $O(2^{n/3})$ qRAM for storing the list L. More time-memory tradeoffs are available. When $|L| < 2^{n/3}$ pairs are obtained in the first step leading to lesser qRAM requirement, the second step will take a dominating time $\sqrt{2^n/|L|}$ achieving a time-qRAM tradeoff with $T^2 \times S = 2^n$ and $S \leq 2^{n/3}$.

The CNS Algorithm. Suppose only a small quantum computer of polynomial size is available but an exponentially large classical memory can be accessed. In this situation, Chailloux *et al.* [8] showed that a collision in time $O(2^{2n/5})$ can be found with a quantum computer of size $O(1)$ and $O(2^{n/5})$ classical memory. The algorithm consists of four steps. The first performs a Grover's search to find 2^{t-r} pairs with r out of the n bits of the output prefixed to zeros, which costs time $2^{t-\frac{r}{2}}$. The second step uses Quantum Amplitude Amplification algorithm [5] to build the superposition collecting all pairs from the first step, and this costs $2^{r/2}$ iterations. The third step constructs the quantum "membership" oracle which can run in time 2^{t-r} by testing sequentially against the elements found in step 1 using classical memory and a small quantum circuit of size $O(n)$. The last

step is the Grover's search using quantum oracle built in step 3 to find multi-preimage from the superposition of step 2. Total running time of the algorithm is $2^{\frac{n-t-1}{2}}(2^{r/2}+2^{t-r})+2^{t-\frac{r}{2}}$ which attain minimum $\frac{2n}{5}$ when $t=\frac{3n}{5}$ and $r=\frac{2n}{5}$.

The product of T and S becomes around $2^{3n/5}$, which is larger than $2^{n/2}$, but it is quite usual to consider a classical memory of size $O(2^{n/5})$, whose availability is obviously higher than qRAM in reality as of now. Hence, CNS shows another more realistic tradeoff between time and space when quantum and classical memories are treated separately.

Time-Space Tradeoff. As observed by Bernstein, from the view point of time-space complexity, BHT is worse than the classical parallel rho method by Van Oorschot and Wiener [50]. The parallel rho method works as follow. Firstly, each processor of P selects a starting point and produces a trail of points until it reaches a point of distinguished property such as certain number of leading zeros, in the meanwhile keeps track on the number of elements produced. When a distinguished point is found, the second step adds it to a common list and start a new search from a new starting point. This trail production is repeated until a collision is found among the distinguished points, which implies a collision with high probability. Roughly speaking, when P classical processors are available, the parallel rho method [35] finds a collision in time $O(2^{n/2}/P)$ according to the birthday paradox. Thus, if a quantum computer of size $2^{n/3}$ is available without qRAM, by running the parallel rho method on the quantum computer, a collision could be found in time $2^{n/6}$, which is much faster than BHT. Let S denote the size of computational resources required for a quantum algorithm (*i.e.*, S is the maximum size of quantum computers and classical memory) and T denote its time complexity. Then the tradeoff $T\cdot S=2^{n/2}$ given by the parallel rho method is the best one even in the quantum setting.

Admissible Probabilities. Given a path with probability p, it costs $p^{-1/2}$ time to find a conforming pair by Grover's search. Since Grover's requires no qRAM, one needs only time lower than the respective bounds to ensure the validity of the attack, which gives the admissible probability of $p\geq 2^{-2n/3}, 2^{-4n/5}, 2^{-n}\cdot S^2$, respectively for BHT, CNS, and the parallel rho method as summarized in Table 1.

2.4 Quantum multicollision algorithm

As a generalization of quantum collision-finding algorithm, quantum q-multicollision algorithm considers the scenario where $F: X\to Y$ is a q-to-1 function with $|X|=q|Y|=qN$. The following algorithm could be abstracted from [32] with $O(N^{\frac{1}{2}(1-\frac{1}{2^q-1})})$ quantum queries:

- Prepare a list $L_1=(x_i, y_i=F(x_i))_{i=1}^{t_1}$ where x_i are distinct and $t_1=N^{(2^{q-1}-1)/(2^q-1)}$. This requires $O(N^{(2^{q-1}-1)/(2^q-1)})$ classical queries on random points.
- With L_1 stored in qRAM as targets, run Grover's algorithm to find t_2 collisions. Each collision costs $O(\sqrt{N/N^{(2^{q-1}-1)/(2^q-1)}})=O(N^{2^{q-2}/(2^q-1)})$ quan-

tum queries. With t_2 set to be $N^{(2^{q-2}-1)/(2^q-1)}$, the overall query complexity in this step is $O(N^{(2^{q-1}-1)/(2^q-1)})$. Store all t_2 collisions in L_2.
- With L_2 as targets, repeat the above steps to find $t_3 = N^{2^{q-3}/(2^q-1)}$ 3-collisions, and store them in L_3. This takes $O(t_3 \cdot \sqrt{N/t_2}) = O(N^{(2^{q-1}-1)/(2^q-1)})$ quantum queries.
- Repeat the above step for $t_4, t_5, \ldots, t_i = N^{(2^{q-i}-1)/(2^q-1)}, \ldots, t_{q-1} = N^{1/(2^q-1)}$. Each repetition costs the same $O(N^{(2^{q-1}-1)/(2^q-1)})$ quantum queries.
- Finally given t_{q-1} $(q-1)$-collisions, run Grover's algorithm to find one x' that leads a q-multicollision with one of the $(q-1)$-multicollision from L_{q-1}. This step also takes $\sqrt{N/t_{q-1}} = N^{(2^{q-1}-1)/(2^q-1)}$ quantum queries.

The overall quantum queries made by this algorithm is $O(N^{(2^{q-1}-1)/(2^q-1)})$, aided by the same amount of qRAM. It is noted that the BHT algorithm is a special case of $q = 2$.

3 The Quantum q-multicollision Distinguisher

When the underlying function is ideal and can only be queried as a blackbox, Liu and Zhandry [32] as reviewed in previous section give an algorithm of complexity $O(N^{(2^{q-1}-1)/(2^q-1)})$ for the q-multicollision finding problem with proven tight bounds. Hence, a dedicated algorithm which finds q-multicollision with lesser time and/or qRAM will be considered a valid distinguisher, which in turn implies the function under attack is not ideal. Blockciphers are permutations when key is fixed, for which collisions or multi-collisions do not exists. Hence, to fit into this model, Biryukov *et al.* [2] define the function to be

$$F_{\Delta_K,\Delta_P}(K,P) = E_K(P) \oplus E_{K\oplus\Delta_K}(P \oplus \Delta_P),$$

where E_K is the block cipher of interest. F with (K,P) as the input can be considered as a pseudo-random function according to Patarin [42]. Then the q-multicollision for F can be defined as follows.

Definition 1. *Given two fixed differences Δ_K and Δ_P. A q-multicollision of a cipher $E_K(\bullet)$ is a set of q $(q \geq 2)$ pairs,*

$$(P_1, K_1), (P_2, K_2), \ldots, (P_q, K_q)$$

that satisfies

$$\begin{aligned} E_{K_1}(P_1) \oplus E_{K_1\oplus\Delta_K}(P_1 \oplus \Delta_P) = E_{K_2}(P_2) \oplus E_{K_2\oplus\Delta_K}(P_2 \oplus \Delta_P) = \\ = \cdots = E_{K_q}(P_q) \oplus E_{K_q\oplus\Delta_K}(P_q \oplus \Delta_P). \end{aligned} \tag{1}$$

Here the two differences Δ_K and Δ_P are fixed.

From the block cipher's respective when there exists a high probability differential path, F is essentially the XOR difference of a ciphertext pair following

the differential path under plaintext difference Δ_P and key difference Δ_K. In such case, the above q-multicollision can be translated into finding q conforming pairs, which costs time $q \cdot p^{-1/2}$ by Grover's search for a probability p differential path. To ensure this leads to a valid distinguisher with lower time complexity, we need $q \cdot p^{-1/2} \leq q \cdot N^{(2^{q-1}-1)/(2^q-1)}$ with $N = 2^n$, then the admissible probability should be $p \geq 2^{\frac{n}{2}(1-\frac{1}{2^q-1})}$.

Note, although the differential is under the related-key setting with key difference fixed, the entire model is however under the chosen-key setting since the secret key K is also an input to F which can be chosen freely by the attackers.

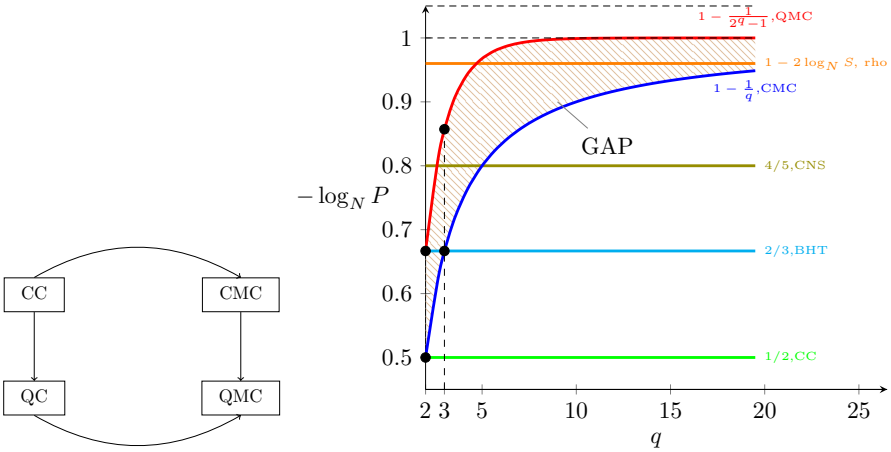

Figure 1: Left: Development from Collision to q-Multicollision [2] and Quantum Collision [20], then to Quantum q-Multicollision in this paper. Right: Comparison of Admissible Probabilities.

Comparison with CMC. In [2], the CMC distinguisher model was introduced and applied to the full 14-round AES-256. Rather than following directly the bound of CMC ($2^{1-1/q}$) from [49], they considered the repeated queries under the related-key setting and came up with a slightly twisted bound. As Liu and Zhanry's QMC algorithm [32] follows directly from the procedure considered in [49], we are following these two bounds for comparisons. Clearly, the admissible probability for QMC $2^{-n(1-1/(2^q-1))}$ is smaller than that for CMC ($2^{-n(1-1/q)}$) for all $q \geq 2$. We define the logarithm of probability as $p_l = \log_{2^{-n}}(p)$, then the gap of two admissible probabilities is $p_l^{\text{QMC}} - p_l^{\text{CMC}} = \frac{1}{q} - \frac{1}{2^q-1}$. The gap is largest of $\frac{1}{3} - \frac{1}{2^3-1} = \frac{4}{21}$ at $q = 3$ as depicted in Figure 1, hence $q = 3$ is used in the rest of the paper unless otherwise specified. If a differential path of probability p_l falls in between of p_l^{QMC} and p_l^{CMC}, it means this path will lead to a valid QMC attack, but invalid CMC attack. Hence, the larger the gap, the higher the likelihood to result in a round difference of the two attacks for the same cipher.

Such differences will be highlighted later when we discuss the applications to the target ciphers.

Comparison with QC. When compared with admissible probabilities of CC and QC as depicted in Figure 1, QMC covers a wider range (or smaller probability) than CC, and the QC algorithms BHT and CNS, for all $q \geq 3$. p_l^{QMC} is larger than p_l^{rho} when $q \geq \log_2\left(\frac{n}{2 \cdot \log_2(S)} + 1\right) \approx \log_2(n) - (1 + \log_2(\log_2(S)))$, as summarized in Table 3. This shows QMC can cover no lesser rounds than QC in most of the cases, and this is supported by the result summary in Table 2.

algorithm	CC	BHT	CNS	parallel rho	CMC
condition	$q \geq 2$	$q \geq 2$	$q \geq 3$	$q \geq \log_2(n) - (1 + \log_2(\log_2(S)))$	$q \geq 2$

Table 3: Range of q when QMC accepts a lower admissible probability, where S is the size of available qRAM.

4 The Attack Framework and Techniques

To find the q-multicollision of a given block cipher, we follow the Definition 1 to find q conforming pairs following a differential path. The overall attack procedure, as depicted in Figure 2, works in the following 4 steps, with time complexity optimization in mind.

Step 1: find a high differential path under the related-key setting. To find differential path with highest possible probability, automatic search tools are reviewed and that from [17] are invoked here. Δ_P and Δ_K are determined together with the path.

Step 2: some probabilistic transitions in the differential path can be fulfilled deterministically by presetting some state values. In case of our mainly concerned AES-like ciphers, the input or output of these active Sboxes can be fixed utilizing such degree of freedoms from both state and key bytes. To maximize the number of such fixed active Sboxes, triangulation algorithm developed in [26] will be reviewed and used.

Step 3: the remaining active Sboxes are fulfilled by Grover search, where candidates are generated from the remaining degree of freedoms from Step 2.

Step 4: additional adhoc optimizations are done to minimize the final complexity.

In the sequel, we introduce the techniques and algorithms used in each step.

4.1 Automatic tools for related-key differential paths

Generic solver Constraint Programming (CP) is used to solve Constraint Satisfaction Problems (CSPs). A CSP is defined by a triple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where

- $\mathcal{X}$ is a finite set of variables;
- $\mathcal{D}$ is refers to the domain, *i.e.*, the set of values each $x_i \in X$ can take;

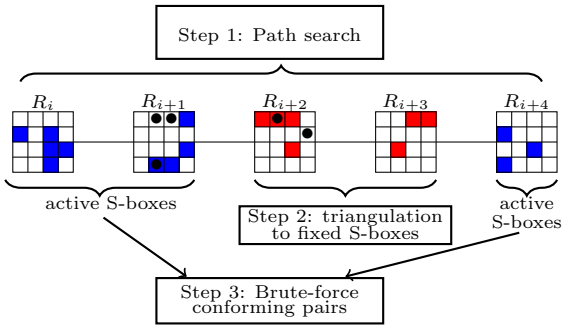

Figure 2: The attack framework

– $\mathcal{C}$ is a set of constraints including relations between variables.

When an objective function is defined, the CSP becomes a Constrained-Optimization Problem (COP). A solution of a COP is an assignment of values to all the variables in $\mathcal{X} = \{x_0, \cdots, x_{n-1}\}$ such that all constraints from $\mathcal{C} = \{c_0, \cdots, c_{m-1}\}$ are satisfied and objective function achieves maximum or minimum.

Finding an optimal related-key differential trail is a highly combinatorial problem that hardly scales. To simplify this problem, a usual and efficient way is to divide it into two steps [3, 13]. Step 1 searches for all truncated differential characteristics under a given bound on the numbers of round and active S-Boxes. It may happen that no actual differential characteristic follows the truncated differential found in Step 1. Hence Step 2 examines and decides whether the truncated differential characteristics are valid, and finds the actual differential characteristic that maximizes the probability. Both steps can be approached by CP. Such CP strategy has been successful in finding related-key differential characteristics for AES [18], Midori [14], and SKINNY [48], in the sense that the truncated differentials match the lower bound on the number of active S-boxes (a.k.a. optimal truncated differentials).

In 2014, Minier *et al.* [36] proposed to use tools to automatize Step 1 for finding best truncated differentials. In 2020, Gérault *et al.* [17] added more constraints to describe the KeySchedule and MixColumn more precisely to filter out those truncated differentials without valid differential characteristics following them, which in turn also reduced the search space and improved the efficiency of the tool. Within the space of valid truncated differentials, Step 2 follows the same CP program used in Step 1, but considers the exact byte differences for the entire differential characteristics instead of a binary value 0/1 in truncated differentials. Following these significant developments in [17, 36], our search model can be described as follows. The constraint includes the 5 steps of the round function in a AES-like cipher, *i.e.*, S-boxes, AddRoundKey, ShiftRow, MixColumn, and KeySchedule, and the objective function is to minimize the number of active S-boxes and to maximize the overall differential probability. These constraints together describe the underlying cipher in the language of CP, which ensures the differential characteristics found by the program will follow exactly the cipher.

$$
\begin{cases}
\text{1. Objective function:} \\
\quad obj = \sum_{(x^r_{i,j}\text{is active})} \Delta x^r_{i,j} + \sum_{(k^r_{i,3}\text{is active})} \Delta k^r_{i,3} \\
\text{2. Constraints on S-boxes in states and subkeys:} \\
\quad \Delta x^r_{i,j} = \Delta y^r_{i,j} \\
\quad \Delta SK^r_i = \Delta K^r_{i,3} \\
\text{3. Constraints on AddRoundKey:} \\
\quad \Delta w^r_{i,j} + \Delta K^{r+1}_{i,j} + \Delta x^{r+1}_{i,j} \neq 1 \\
\text{4. Constraints on ShiftRow:} \\
\quad \Delta y^r_{i,(i+j) \bmod 4} = \Delta z^r_{i,j} \\
\text{5. Constraints on MixColumn:} \\
\quad \sum_{i=0}^{3} \Delta z^4_{i,j} + \sum_{i=0}^{3} \Delta w^4_{i,j} \in \{0,5,6,7,8\} \\
\text{6. Constraints on KeySchedule:} \\
\quad \Delta K^{r+1}_{i,0} + \Delta K^r_{i,0} + \Delta SK^r_{(i+1) \bmod 4} \neq 1 \\
\quad \Delta K^{r+1}_{i,j} + \Delta K^{r+1}_{i,j-1} + \Delta K^r_{i,j} \neq 1
\end{cases}
$$

where $i, j \in [0,3]$ and SK^r_i are introduced variables after S-boxes operations in r_{th}-round KeySchedule.

Technically, the truncated differential search of Step 1 is implemented by MiniZinc [40] language, which is subsequently solved by Picat-SAT [51]. Then the search of actual differential characteristics in Step 2 is defined and solved by Choco solver [44]. The execution of our search programs is performed on a single core Intel Core i9 processor at 3.6 GHz, and all can complete within one hour while most in less than 10 minutes. To minimize the final complexity of our distinguishers, a set of sub-optimal differentials, rather than the optimal differential only, are prepared for the triangulation algorithm.

4.2 Triangulation Algorithm

The Triangulation Algorithm(TA) proposed in [26] uses Gaussian elimination to solve systems of non-linear equations. Unlike a universal algorithm dealing with any non-linear function, it is efficient for solving system of bijective functions only. To illustrate how it works, we use the following system of equations as a toy example.

$$
\begin{cases}
x_1 \oplus F(x_2) & = 0 \\
x_1 \oplus (3 \cdot x_3) \oplus G(x_4) & = 0 \\
x_2 \oplus x_4 \oplus a & = 0
\end{cases} \tag{2}
$$

Here F and G are some bijective functions, and a is a constant. Note all other operations including multiplication by 3 and exclusive-or are also bijective. To translate the above system into TA, all bijective functions are removed, and the system can be re-expressed as a matrix with columns for the variables and rows for the equations. The entries are binary values, and are "1" only when a variable

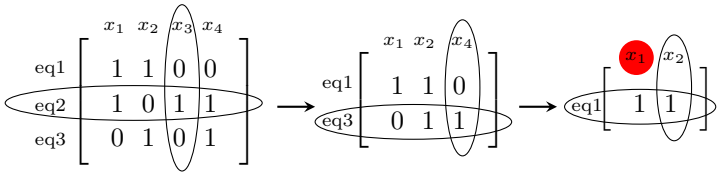

Figure 3: Procedure of Triangulation Algorithm for solving the system (2)

appears in the respective equation. The algorithm starts with marking all the equations as unprocessed. Next, the algorithm finds that x_3 is involved in the eq2 only, *i.e.*, there is only one "1" in the third column of the matrix. Then, the third column and the second row corresponding to this "1" are marked as processed, and will be removed from the matrix, as highlighted in the left most matrix in Figure 3. Within the resulted matrix after removal, the next column with a single "1" will be the column x_4, then this column and the corresponding eq3 are marked and to be removed from the matrix. After that, x_2 is the next variable to be marked as processed and removed. Finally, the TA outputs x_1 as free variable. 3. Besides the set of free variables, the TA also gives us who the system of equations can be fulfilled, following the reversed order of the execution procedure, *i.e.*, after the free variable(s) x_1 are assigned to some values, x_2 is used to fulfill eq1, followed by x_4 for eq3, and finally x_3 for eq2. Note, in this toy example, Gaussian elimination is not used, but necessary for a general case at the beginning of each step to ensure a column with a single "1" will appear.

When a differential characteristic of an AES-like block cipher is given, and the attacker is given full control over the state and key values, we are interested in finding the maximum amount of active S-boxes that can be fulfilled by setting to the respective conforming values. TA serves this purpose, with state bytes and key bytes as variables, and the round function and key schedule as the system of equations. This can be applied to multiple rounds one by one, *i.e.*,TA is applied to one round, then to the next rounds with only those free variables returned by the TA from the previous round, this is repeated until all free variables are exhausted. Our implementation shows the problem sizes in our attack are small and all our TA programs can finish execution instantly on a PC.

4.3 Complexity Optimizations

To further minimize the overall complexity, the following measures are integrated into the attack procedure.

1. TA is applied to consecutive rounds, for as many rounds as possible. This process is repeated for all starting round, and the maximum number of fixed active S-boxes is selected among all choices.
2. We note it is not necessary that the optimal differential characteristic from Step 2 leads to the lowest attack complexity after the execution of TA. Hence, instead of a single optimal path, a set of sub-optimal paths are collected from

Step 2, then TA is run for all the paths to identify the best one with highest remaining probability.

3. It is noted that S-box operation only applies to the last column of the round key bytes in the key schedule, while this is for all state bytes in round function. Hence, it is likely many degrees of freedom from key bytes will be used to fulfill some active S-boxes in the state. However, a key byte variable may not affect all state byte in a bijective way. For example, the first byte of i-th round key x is added into the state at round i, the first state byte of the i-th round will be $2 \cdot S(x)$, then the same key byte will propagate through key schedule to the first key byte for round $i+1$, and added to the first state byte at round $i+1$ resulting in $2 \cdot S(x) + x$. If the first state S-box at round $i+1$ is active and x is the variable used to fulfill it, one can precompute $2 \cdot S(x) + x$ for all possible 2^8 possible x values and store them in a lookup, which saves the trouble of bruteforce every time. While the function can be more complicated involving many variables, such a lookup table with pre-computation is always possible as long as the complexity here does not dominate the whole attack.

5 Applications on AES, Rijndael and Saturnin

Our QMC attack framework is generic, and can be applied to any given block cipher. To demonstrate the effectiveness, well studied targets including AES, Rijndael, and Saturnin are used here. To make a comprehensive comparison, besides QMC we also apply the attack framework to find CMC. Detailed attack procedure on AES will be given for the readers to follow the techniques, then brief results are described for subsequent targets.

5.1 AES

Description of AES. AES-k is a block cipher family of 128-bit block and k-bit key for $k \in \{128, 192, 256\}$. The state has 16 bytes and can be represented as a 4×4 matrix. Given a $\mathrm{N_{row}} \times \mathrm{N_{col}}$ bytes of the state matrix, where $\mathrm{N_{row}} = \mathrm{N_{col}} = 4$, we order the byte cells as in Figure 4. Then the state is encrypted by an iterative process which is repeated for 10, 12, and 14 rounds, for AES-128, AES-192, and AES-256, respectively. An AES round function, as depicted in Figure 4, is an Substitution-Permutation Network (SPN), and composed of four consecutive operations: SubBytes (SB), ShiftRows (SR), MixColumn (MC), and AddRoundKey (AK). The master key k is added to the state before the application of the first round function, and is used to generated to r subkeys through the KeySchedule (KS) function. We refer to Appendix A.1 the details of both AES and Rijndael.

Attack Procedure on AES. We apply the search tool described in Section 4.1 to find the related-key differential paths of AES-128 reduced to 7 rounds, 8 rounds, and the full 12-round AES-192, while that for the full 14-round AES-256

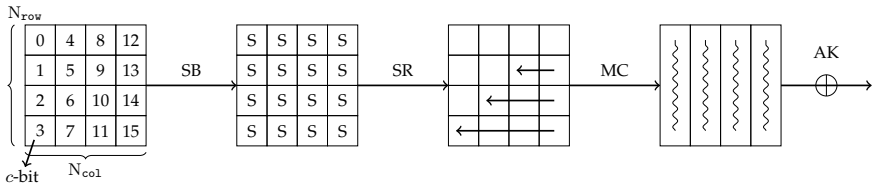

Figure 4: The AES round function

has been already found and used in [2, 17]. A lower bound of probability $2^{-(n+k)}$ is set to limit the search space, where n is the state size and k is the key size in bits. This bound is used to ensure there will be at least one pair of message conforming the differential path, utilizing all degrees of freedom from both state and key. A set of differential paths are collected.

Let denote by y_i, w_i, k_i as the state after SB, MC, and the sub-key at i-th round. Each of them is an array of 16 bytes following the same order as in Figure 4. Then, the i-th round function involving y_i, w_{i-1}, k_i and w_i can be re-expressed by Equation (3), where the first line $y_i \oplus S(w_{i-1} \oplus k_i)$ reassembles two operations AK and SB, and the second line reassembles SR and MC. The i-th round key schedule can be re-expressed by the Equation (4), relating key bytes of the current round k_i with that from the previous round k_{i-1}. The same Equation (4) is used for both AES-128 with $i = 0, \ldots, 10$ and $b = 16$, and AES-192 with $i = 0, \ldots, 8$ and $b = 24$. There are 32 free variables (16 from the state and 16 from the key) for AES-128, and 40 free variables (16 from the state and 24 from the key) for AES-192. Each equation will form a line with 1/0 indicating the presence of the respective variable in the matrix input to TA.

$$R_i : \begin{cases} y_i \oplus S(w_{i-1} \oplus k_i) = 0 \\ w_i \oplus \begin{pmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{pmatrix} \times \begin{pmatrix} y_i[0] & y_i[4] & y_i[8] & y_i[12] \\ y_i[5] & y_i[9] & y_i[13] & y_i[1] \\ y_i[10] & y_i[14] & y_i[2] & y_i[6] \\ y_i[15] & y_i[3] & y_i[7] & y_i[11] \end{pmatrix} = 0 \end{cases} \tag{3}$$

$$KS_i : \begin{cases} k_i[j] \oplus k_i[j-4] \oplus k_{i-1}[j] = 0, & j = 4, \ldots b-1 \\ k_i[0] \oplus k_{i-1}[0] \oplus S(k_{i-1}[b-3]) \oplus \mathbf{RCON}_i = 0 \\ k_i[1] \oplus k_{i-1}[1] \oplus S(k_{i-1}[b-2]) = 0 \\ k_i[2] \oplus k_{i-1}[2] \oplus S(k_{i-1}[b-1]) = 0 \\ k_i[3] \oplus k_{i-1}[3] \oplus S(k_{i-1}[b-4]) = 0 \end{cases} \tag{4}$$

Results on 8-round AES-128. After CP tool run on a PC for a few minutes, the desired differential characteristics are found on 8-round AES-128. The one, as depicted in Figure 5, is formed with 37 active S-boxes, whereas 6 of them are in

[3] Legend is used for the same meaning for the subsequent differential paths

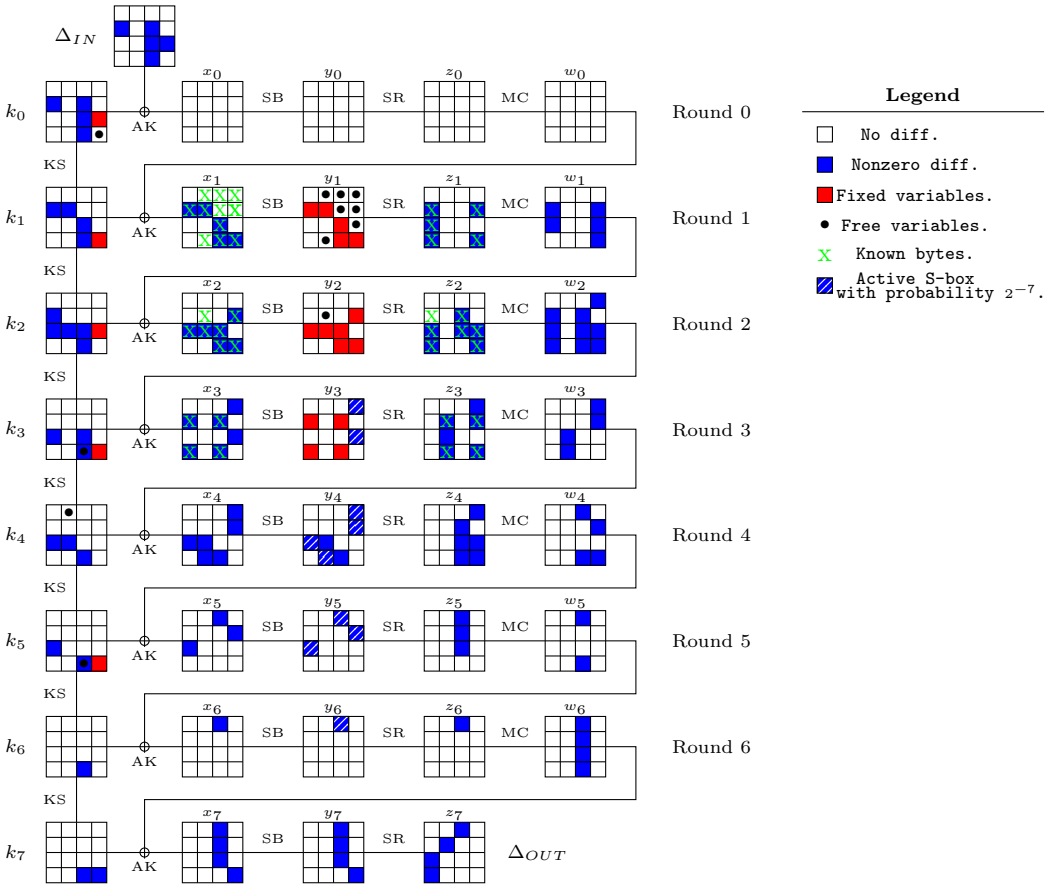

Figure 5: A differential path [3] for 8-Round AES-128

Cipher	Attacked Rounds	Active S-boxes Type-I + Type-II	Fixed bytes Type-I + Type-II	Final Probability p_{out}	Ref.
AES-128	6	16 + 5	10 + 5	2^{-36}	[48],Fig. 9
	7	13 + 16	5 + 10	2^{-90}	Fig. 10
	8	10 + 27	(3 + 17) or (3 + 18)	2^{-112} or 2^{-105}	Fig. 5
AES-192	10	22 + 8	9 + 8	2^{-78}	[17],Fig. 12
	12	23 + 16	6 + 16	2^{-102}	Fig. 13
AES-256	14	22 + 2	11 + 2	2^{-66}	[17]

Table 4: Summary of AES related-key differential paths

the sub-keys and 31 are in the state. For the AES S-box, there are two types of differentials with probability of 2^{-6} and 2^{-7}, which we will refer to as Type-I and Type-II. They are depicted in the figures of differential path as boxes in blue only, and blue with white lines, respectively. Among the 37 active S-boxes, there are 10 Type-I and 27 Type-II, which gives an overall probability of $2^{-(10\times6+27\times7)} = 2^{-249}$. After execution of TA with all possible starting round, we find the one starting with Round 1 is the best choice, which allows to fix all active S-boxes in Round 1 and Round 2, and 4 of the state in Round 3, as well as 5 out of the 6 active S-boxes in sub-keys, as highlighted in red in the Figure 5. TA finds the active S-box in k_7 cannot be fixed probably because it is too far from the Round 1 variables. After TA, 7 Type-I and 10 Type-II active S-boxes are left unfixed, resulting in a probability of $p_{out} = 2^{-(7\times6+10\times7)} = 2^{-112}$. Grover search then finds a conforming pair in 2^{56} quantum queries, and hence a QMC in $q\cdot 2^{56}$. This complexity is lower than the generic bound when $p_l = 112/128 = 0.875 < 1 - \frac{1}{2^{q-1}}$, *i.e.*, $q \geq 4$. The same differential leads to a CMC attack with complexity $q\cdot 2^{112}$, which is a valid attack when $p_l < 1 - 1/q$, *i.e.*, $q \geq 9$. The gap of the q ranges can be interpreted as the differential leads to a valid QMC attack but invalid CMC attack in the range $4 \leq q \leq 8$.

To check if TA works as expected and no byte is over-defined in the system, we verified the entire procedure to reproduce all other state and key bytes from the set of fixed bytes and free bytes. As also highlighted in red, the fixed bytes are $\{k_0[14], k_1[15], k_2[14], k_3[15], k_5[15], y_1[1], y_1[5], y_1[10], y_1[11], y_1[15], y_2[2], y_2[6], y_2[10], y_2[11], y_2[13], y_2[15], y_3[1], y_3[3], y_3[9], y_3[11]\}$, and the free bytes are $\{k_0[15], k_3[11], k_4[4], k_5[11], y_1[4], y_1[7], y_1[8], y_1[9], y_1[12], y_1[13], y_1[14], y_2[5]\}$. From these 32 bytes, Table 5 shows step by step how the entire key state k_2 (highlighted in red) and state y_1 (highlighted in blue) are derived. MC^{-1} is the operator acting on any 4 bytes out of 8 bytes of the columns before and after MC and resulting in the remaining 4 bytes.

Fixing one more active S-box. In this part, we describe a method to fulfill one additional active S-box at $y_3[12]$ for free at the cost of some qRAM. Along with the previous fixed bytes, we fix one more byte $k_3[12]$ and receive other free variables from another run of TA. Note that byte $k_3[12]$ will not be fixed to particular value, but will be chosen so that $k_3[12] \oplus x_3[12]$ fulfill the S-box at

Fixed bytes: $k_0[14], k_1[15], k_2[14], k_3[15], k_5[15], y_1[1], y_1[5], y_1[10], y_1[11], y_1[15],$ $y_2[2], y_2[6], y_2[10], y_2[11], y_2[13], y_2[15], y_3[1], y_3[3], y_3[9], y_3[11]$	
Free bytes: $k_0[15], k_3[11], k_4[4], k_5[11], y_1[4], y_1[7], y_1[8], y_1[9], y_1[12], y_1[13], y_1[14], y_2[5]$	
1. $k_1[11] = k_0[15] \oplus k_1[15]$	27. $k_2[3] = S(k_2[12]) \oplus k_3[3]$
2. $k_2[15] = k_3[11] \oplus k_3[15]$	28. $k_1[14] = k_2[10] \oplus k_2[14]$
3. $k_4[15] = k_5[11] \oplus k_5[15]$	29. $k_1[10] = k_0[14] \oplus k_1[14]$
4. $k_4[11] = k_3[15] \oplus k_4[15]$	30. $k_2[6] = k_1[10] \oplus k_2[10]$
5. $k_4[7] = k_4[11] \oplus k_3[11]$	31. $x_1[6] = x_2[6] \oplus k_2[6]$
6. $k_5[7] = k_5[11] \oplus k_4[11]$	32. $y_1[3] = z_1[7], w_1[4,5,7]$ $= MC^{-1}(z_1[4,5,6], w_1[6])$
7. $k_5[3] = k_5[7] \oplus k_4[7]$	33. $k_2[5] = w_1[5] \oplus x_2[5]$
8. $w_1[15] = x_2[15] \oplus k_2[15]$	34. $x_2[7] = w_1[7] \oplus k_2[7]$
9. $y_1[6] = z_1[14], w_1[12,13,14]$ $= MC^{-1}(z_1[12,13,15], w_1[15])$	35. $z_2[8], w_2[8,9,10]$ $= MC^{-1}(z_2[9,10,11], w_2[11])$
10. $k_2[13] = x_2[13] \oplus w_1[13]$	36. $k_2[8] = x_2[8] \oplus w_1[8]$
11. $x_2[14] = k_2[14] \oplus w_1[14]$	37. $k_3[9] = w_2[9] \oplus x_3[9]$
12. $w_2[11] = x_3[11] \oplus k_3[11]$	38. $k_3[4] = k_2[8] \oplus k_3[8]$
13. $k_2[11] = k_1[15] \oplus k_2[15]$	39. $k_4[0] = k_3[4] \oplus k_4[4]$
14. $k_2[7] = k_1[11] \oplus k_2[11]$	40. $k_3[13] = k_3[9] \oplus k_2[13]$
15. $k_3[7] = k_2[11] \oplus k_3[11]$	41. $k_3[0] = S(k_3[13]) \oplus k_4[0]$
16. $k_3[3] = k_2[7] \oplus k_3[7]$	42. $k_2[4] = k_3[0] \oplus k_3[4]$
17. $k_4[3] = k_3[7] \oplus k_4[7]$	43. $k_2[0] = S(k_2[13]) \oplus k_3[0]$
18. $k_4[12] = S^{-1}(k_4[3] \oplus k_5[3])$	44. $w_2[3] = k_3[3] \oplus x_3[3]$
19. $k_3[12] = S^{-1}(k_3[3] \oplus k_4[3])$	45. $z_2[0], w_2[0,1,2]$ $= MC^{-1}(z_2[1,2,3], w_2[3])$
20. $k_4[8] = k_3[12] \oplus k_4[12]$	46. $w_1[0] = k_2[0] \oplus x_2[0]$
21. $k_3[8] = k_4[4] \oplus k_4[8]$	47. $y_1[0] = z_1[0], w_1[1,2,3]$ $= MC^{-1}(z_1[1,2,3], w_1[0])$
22. $k_2[12] = k_3[8] \oplus k_3[12]$	48. $k_2[2] = w_1[2] \oplus x_2[2]$
23. $w_1[11] = k_2[11] \oplus x_2[11]$	49. $k_3[1] = w_2[1] \oplus x_3[1]$
24. $y_1[2] = z_1[10], w_1[8,9,10]$ $= MC^{-1}(z_1[8,911], w_1[11])$	50. $k_2[1] = k_3[1] \oplus S(k_2[14])$
25. $k_2[10] = w_1[10] \oplus x_2[10]$	51. $k_3[5] = k_2[5] \oplus k_3[1]$
26. $x_2[12] = w_1[12] \oplus k_2[12]$	52. $k_2[9] = k_3[5] \oplus k_3[9]$

Table 5: Steps to derive the entire key k_2 and state y_1 from the fixed and free bytes.

$y_3[12]$. Let denote the value of $k_3[12]$ as x. Then the value of $y_3[12]$ depends on x and some more free variables and fixed variables, more precisely

$$y_3[12] = S(03 \times S(01 \times 02^{-1}x \oplus c_1) \oplus c_2 \oplus x), \tag{5}$$

where c_1 and c_2 are 8-bit values depending on the 31 free bytes and fixed bytes. To verify the expression above is possible, the procedure to find the relation between $y_3[12]$ and x from the free and fixed bytes are listed in Table 6. A lookup table with the triplet (c_1, c_2, x) can be precomputed and stored in qRAM for superposition access. When the values of the 31 main bytes are fixed, the corresponding (c_1, c_2) can be computed, and suitable x can be identified from the lookup table so that the active S-box at $y_3[12]$ can be fulfilled. The cost of this lookup table is classical computation effort of 2^{16} and 2^{16} qRAM. This

method allows to fix one additional Type-II active S-box, resulting in the final $p_{out} = 2^{-105}$ as depicted in Figure 5.

Fixed bytes: $k_0[14], k_1[15], k_2[14], k_3[12] = x, k_3[15], k_5[15], y_1[1], y_1[5], y_1[10], y_1[11], y_1[15], y_2[2], y_2[6], y_2[10], y_2[11], y_2[13], y_2[15], y_3[1], y_3[3], y_3[9], y_3[11]$	
Free bytes: $k_0[15], k_3[11], k_4[4], y_1[4], y_1[7], y_1[8], y_1[9], y_1[12], y_1[13], y_1[14], y_2[5]$	
1. $k_2[15] = k_3[11] \oplus k_3[15]$	27. $k_4[15] = k_4[11] \oplus k_3[15]$
2. $w_1[15] = x_2[15] \oplus k_2[15]$	28. $k_5[11] = k_4[15] \oplus k_5[15]$
3. $z_1[14], w_1[12, 13, 14]$ $= MC^{-1}(z_1[12, 13, 15], w_1[15])$	29. $k_5[7] = k_5[11] \oplus k_4[11]$
4. $k_2[13] = w_1[13] \oplus x_2[13]$	30. $k_5[3] = k_4[7] \oplus k_5[7] = S(x) \oplus c$
5. $k_2[13] = w_1[13] \oplus x_2[13]$	31. $k_4[12] = S^{-1}(k_4[3] \oplus k_5[3]) = S^{-1}(c \oplus k_3[3])$
6. $k_3[7] = k_2[11] \oplus k_3[11]$	32. $k_4[8] = x \oplus k_4[12]$
7. $w_1[11] = x_2[11] \oplus k_2[11]$	33. $k_3[8] = k_4[8] \oplus k_4[4]$
8. $z_1[10], w_1[8, 9, 10]$ $= MC^{-1}(z_1[8, 9, 11], w_1[11])$	34. $k_2[12] = x \oplus k_3[8] = c'$
9. $k_2[10] = w_1[10] \oplus x_2[10])$	35. $x_2[12] = w_1[12] \oplus k_2[12]$
10. $k_1[14] = k_2[10] \oplus k_2[14]$	36. $k_2[1] = S(k_2[14]) \oplus k_3[1]$
11. $k_1[10] = k_0[14] \oplus k_1[14]$	37. $k_2[3] = S(k_2[12]) \oplus k_3[3]$
12. $k_1[11] = k_0[15] \oplus k_1[15]$	38. $x_2[7] = w_1[7] \oplus k_2[7]$
13. $k_2[7] = k_1[11] \oplus k_2[11]$	39. $z_2[8], w_2[8, 9, 10]$ $= MC^{-1}(z_2[9, 10, 11], w_2[11])$
14. $k_3[3] = k_2[7] \oplus k_3[7]$	40. $k_2[8] = w_1[8] \oplus x_2[8]$
15. $k_2[6] = k_1[10] \oplus k_2[10]$	41. $k_3[4] = k_2[8] \oplus k_3[8] = x \oplus c''$
16. $w_1[6] = x_2[6] \oplus k_2[6]$	42. $k_3[9] = x_3[9] \oplus w_2[9]$
17. $z_1[7], w_1[4, 5, 7]$ $= MC^{-1}(z_1[4, 5, 6], w_1[6])$	43. $k_3[13] = k_3[9] \oplus k_2[13]$
18. $k_2[5] = w_1[5] \oplus x_2[5]$	44. $k_4[0] = k_4[4] \oplus k_3[4] = x \oplus d$
19. $w_2[3] = x_3[3] \oplus k_3[3]$	45. $k_3[0] = S(k_3[13]) \oplus k_4[0] = x \oplus d'$
20. $w_2[11] = x_3[11] \oplus k_3[11]$	46. $k_2[0] = S(k_2[13]) \oplus k_3[0] = x \oplus d''$
21. $z_2[0], w_2[0, 1, 2]$ $= MC^{-1}(z_2[1, 2, 3], w_2[3])$	47. $w_1[0] = x_2[0] \oplus k_2[0] = x \oplus d'''$
22. $k_3[2] = w_2[2] \oplus x_3[2]$	48. $z_1[0], w_1[1, 2, 3]$ $= MC^{-1}(z_1[1, 2, 3], w_1[0])$
23. $k_3[6] = k_2[6] \oplus k_3[2]$	49. $w_1[1] = 01 \times 02^{-1} x \oplus c$
24. $k_4[3] = k_3[3] \oplus S(x)$	50. $y_2[1] = S(01 \times 02^{-1} x \oplus c_1)$
25. $k_4[7] = k_4[3] \oplus k_3[7]$	51. $w_2[12] = 03 \times S(01 \times 02^{-1} x \oplus c_1) \oplus c_2$
26. $k_4[11] = k_4[7] \oplus k_3[11]$	52. $x_3[12] = 03 \times S(01 \times 02^{-1} x \oplus c_1) \oplus c_2 \oplus x$

Table 6: Steps to derive the relation between $x_3[13]$ and the variable x in key byte $k_3[13]$, c, c', c'' and d, d', d'' are 8-bit values depending on the 31 free bytes and fixed bytes (excluding x)

Results on 7-round AES-128. Since the QMC for 8-round AES-128 are not valid for all q values, and the previous quantum collision attacks work for 7 rounds only, we also run our attack framework to 7-round for comparison purposes. The CP tool returns in a few mins the differential path depicted in Figure 10 with 29 actives S-boxes, out of which 22 actives S-are in states and 7 are in subkeys. With $(13, 16)$ and $(8, 6)$ Type-I and Type-II active S-boxes before and after TA,

the respective probabilities are 2^{-190} and 2^{-90}. Then $p_l = 90/128 = 0.703$ gives a valid QMC with complexity $q \cdot 2^{45}$ for $q \geq 3$ and a valid CMC with complexity $q \cdot 2^{90}$ for $q \geq 4$. To find the CMC attack valid for all possible q, we move on to reduce the attacked round to 6.

Results on 6-round AES-128. The best related-key differential of 6-round AES-128 has been found in [48], with 16 Type-I and 5 Type-II active S-boxes, and only 6 Type-I active S-boxes are left. This path gives a final $p_{out} = 2^{-36}$ and $p_l = 0.28$ leading to a valid CMC with complexity $q \cdot 2^{36}$ for all $q \geq 3$.

Results on 12-round AES-192. Similar attack procedure is applied to AES-192, and the probabilities of the differential path before and after application of TA, as depicted in Figure 13, are 2^{-250} and 2^{-102}, respectively. The $p_l = 102/128 \approx 0.80$ gives a valid QMC with complexity $q \cdot 2^{51}$ for $q \geq 3$ and a valid CMC with complexity $q \cdot 2^{102}$ for $q \geq 4$. The attacked round reduces to 10 to obtain the CMC attack valid for $q \geq 3$ with time complexity 2^{78}. All our results on AES are summarized in Table 4.

5.2 Rijndael

Description of Rijndael. Rijndael-b-k (where b is the block size and k is the key size in bits) is the predecessor of AES designed by Daemen and Rijmen [10]. It has 25 variants corresponding to each case of $4 \times \mathrm{N_{col}}$ block size (128, 160, 192, 224 or 256 bits) and the key size (128, 160, 192, 224 or 256 bits). The number of rounds for the 25 instances are 10, 11, 12, 13, and 14 depending on the maximum of block size and key size. The encryption process is the same as AES as described in Section 5.1, except for the KS, SR, and the round numbers. Here we focus on the variants of block size 128 and 160 bits, for which SR works the same as AES by circularly shifting i-th row to the left by i positions.

Results on Rijndael. Our related-key differential characteristics are obtained by modifying the tool from [17] to fit Rijndael. There are 25 instances of Rijndael, we report in this paper only the longest rounds attacked due to space limit. The QMC can be mounted on full rounds of Rijndael-128-160, Rijndael-128-224, 11-round Rijndael-160-192, and 12-rounds of Rijndael-160-256. For Rijndael-128-160 (and similarly for Rijndael-128-224), we listed the results from 9 to 11 rounds, because 9 is the maximum rounds CMC attack can reach for $q = 3$, 10 is the valid QMC attack can archive for $q = 3$, and 11 is the maximum rounds a valid QMC works for some q. When TA is applied, active S-boxes in up to 4 rounds can be fixed, compared with 3 for AES. Details of the differential paths, before and after the application of TA, are summarized in Table 7. It is interesting to note that differential paths leading to the best attack for 3 out of the 4 variants are optimal, except for Rijndael-160-256. In this special case, a differential path with 41 active S-boxes instead of the optimal one with 40 active S-boxes is used. This path has more active S-boxes in the keys rather than the state than the optimal path. After application of TA, this sub-optimal path leads to higher final probability p_{out}. This is consistent with out observation that more free variables

are available from key bytes since there are less S-box operations, hence TA has better chance to fix active S-boxes in key.

Cipher	Attacked Rounds	Active S-boxes Type-I + Type-II	Fixed bytes Type-I + Type-II	Final Probability p_{out}	Ref.
Rijndael-128-160	9	22 + 9	12 + 9	2^{-60}	Fig. 19
	10	27 + 9	12 + 9	2^{-90}	Fig. 14
	11	21 + 20	6 + 16	2^{-118}	Fig. 15
Rijndael-128-224	11	19 + 7	9 + 6	2^{-67}	Fig. 20
	13	27 + 11	13 + 10	2^{-91}	Fig. 16
Rijndael-160-192	11	31 + 14	16 + 14	2^{-90}	Fig. 17
Rijndael-160-256	12	34 + 7	17 + 7	2^{-102}	Fig. 18

Table 7: Summary of Rijndael related-key differential paths

5.3 SATURNIN

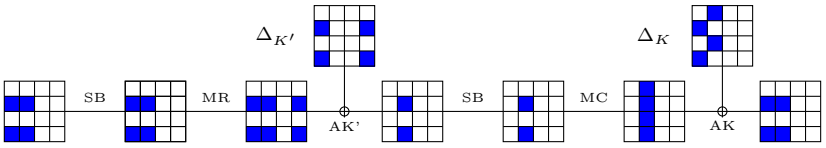

Figure 6: A differential path for 2-Round SATURNIN [7]

Description of SATURNIN. SATURNIN is a block cipher with a 256-bit state and 256-bit key that was designed as the derivative of AES with efficient implementation by Canteaut *et al.* for the NIST lightweight cryptography competition, and it was among the round 2 candidates. It can viewed as a 3-dimensional AES with cell size of 4 bits. The composition of two consecutive rounds starting from even round is called super-round, which is very similar to an AES round operating on 16-bit words except that the SR is replaced by a transposition exactly as used in Square, the predecessor of AES. We refer to Appendix A.2 for the byte orientation and the KeySchedule of SATURNIN for details.

Results on SATURNIN. On SATURNIN's design website, the authors propose a challenge to dig into the security analysis of SATURNIN against the related-key differential attack, starting from 9 rounds. In [7], the designers proposed the first classical related-key attack on 10 rounds, and conclude

"*A quantized version of this attack is expected to reach less rounds ...* "

Contrary to the designers' conclusion, in this paper we successfully mount QMC attack on 10-round SATURNIN. We utilize the differential characteristic proposed by the designers in [7], where a 2-round iterative differential characteristic (refer

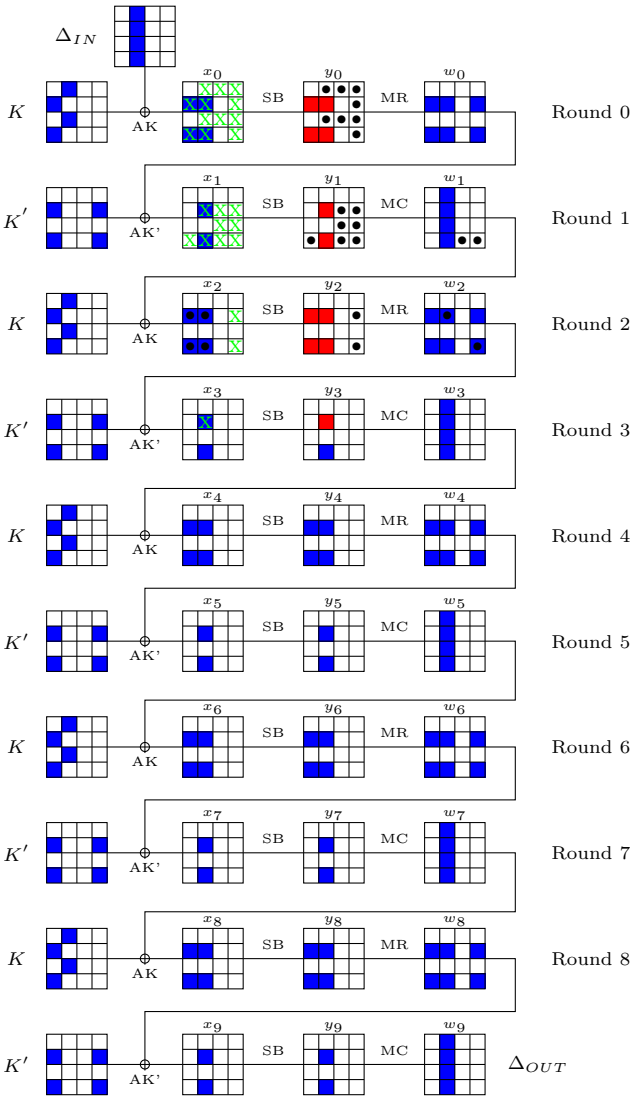

Figure 7: A differential path for 10-Round SATURNIN

to Figure 6) with probability $2^{-78.1}$ is given and repetition by 5 times leads to a 10-round related-key differential characteristic with probability $2^{-390.5}$. Figure 7 shows the full differential characteristic in truncated version.

Next, we prepare the system of equations as the input to TA. The key schedule of SATURNIN is simple byte shuffle, which requires no extra equation to describe. Similar to AES, the round function of SATURNIN can be modeled as the system of equations as follows.

$$i \bmod 2 = 1, R_i : \begin{cases} y_i[j] \oplus S(w_{i-1}[j] \oplus k_0[(j+5) \bmod 16]) = 0, & j = 0, \ldots, 15 \\ w_i \oplus MC \times \begin{pmatrix} y_i[0] & y_i[4] & y_i[8] & y_i[12] \\ y_i[1] & y_i[5] & y_i[9] & y_i[13] \\ y_i[2] & y_i[6] & y_i[10] & y_i[14] \\ y_i[3] & y_i[7] & y_i[11] & y_i[15] \end{pmatrix} = 0 \end{cases},$$

$$i \bmod 2 = 0, R_i : \begin{cases} y_i[j] \oplus S(w_{i-1}[j] \oplus k_0[j]) = 0, & j = 0, \ldots, 15 \\ w_i \oplus \begin{pmatrix} y_i[0] & y_i[1] & y_i[2] & y_i[3] \\ y_i[4] & y_i[5] & y_i[6] & y_i[7] \\ y_i[8] & y_i[9] & y_i[10] & y_i[11] \\ y_i[12] & y_i[13] & y_i[14] & y_i[15] \end{pmatrix} \times MR = 0 \end{cases},$$

where MC and MR are 4×4 MDS matrices. Application of TA shows that all the active S-boxes in first 4 rounds of states except for the last active S-box in byte $y_3[7]$ can be fixed. SATURNIN does not allow us to apply the trick used for AES to save degree of freedom of key bytes, because each byte of the key is involved in various equations. The relationship of the key byte and the corresponding state byte involves many more dependent variables $c_1, c_2, c_3, \cdots$, which increases the requirement of memory significantly. To this end, the probability of the S-box in byte $y_3[7]$ is not lower than 2^{-15}, which results in a $p_{out} = 2^{-249.3}$. This leads to a QMC with complexity $2^{124.65}$ and $q \geq 6$. The similar procedure attack is applied to 8 rounds to archieve general QMC, *i.e.*, $q \geq 3$, with $p_{out} = 2^{-171.2}$, which leads to the complexity $2^{85.6}$. To extend the CMC attack to $q = 3$, the same differential characteristic but reduced further to 7 rounds is used. This leads to a $p_{out} = 2^{-143}$, and a valid CMC attack with complexity 2^{143}, as summarized in Table 8.

Model	Rounds	Active S-boxes	Fixed bytes	Final Probability	Ref.
SATURNIN	7	22	11	2^{-143}	Fig. 7
	8	24	11	$2^{-171.2}$	
	10	30	11	$2^{-249.3}$	

Table 8: Results on SATURNIN' related-key differential paths

6 Conclusions

In this paper, we proposed the quantum multi-collision distinguishers. Our model shows differential paths with probability as low as 2^{-n} will be useful in mounting such attacks, hence resulted in more rounds than both quantum collision attack and classic multi-collision distinguishers. We applied the attack model to AES-like ciphers including all three versions of AES, 4 versions of Rijndael, and the post-quantum block cipher design Saturnin-256. Full attacks are mounted on AES-192, AES-256, Rijndael-128-160, and Rijndael-128-224. Comparing with quantum collision attacks, our attack covered one more round on AES-128. Our attack framework is generic, hence can be applied to more target ciphers. It will be also interesting to see if such distinguishers can be converted into collision attacks on block cipher hashing modes or key recovery.

References

1. Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: Matsui, M. (ed.) Advances in Cryptology – ASIACRYPT 2009. LNCS, vol. 5912, pp. 1–18. Springer, Heidelberg, Germany, Tokyo, Japan (Dec 6–10, 2009)
2. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and related-key attack on the full AES-256. In: Halevi, S. (ed.) Advances in Cryptology – CRYPTO 2009. LNCS, vol. 5677, pp. 231–249. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2009)
3. Biryukov, A., Nikolic, I.: Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to AES, Camellia, Khazad and others. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. LNCS, vol. 6110, pp. 322–344. Springer, Heidelberg, Germany, French Riviera (May 30 – Jun 3, 2010)
4. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer, Heidelberg, Germany, Seoul, South Korea (Dec 4–8, 2011)
5. Brassard, G., Hoyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemporary Mathematics 305, 53–74 (2002)
6. Brassard, G., Høyer, P., Tapp, A.: Quantum cryptanalysis of hash and claw-free functions. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN 1998: Theoretical Informatics, 3rd Latin American Symposium. LNCS, vol. 1380, pp. 163–169. Springer, Heidelberg, Germany, Campinas, Brazil (Apr 20–24, 1998)
7. Canteaut, A., Duval, S., Leurent, G., Naya-Plasencia, M., Perrin, L., Pornin, T., Schrottenloher, A.: A note on related-key attacks on Saturnin. https://project.inria.fr/saturnin/files/2020/11/Note-RK-1.pdf (2020)
8. Chailloux, A., Naya-Plasencia, M., Schrottenloher, A.: An efficient quantum collision search algorithm and implications on symmetric cryptography. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017, Part II. LNCS, vol. 10625, pp. 211–240. Springer, Heidelberg, Germany, Hong Kong, China (Dec 3–7, 2017)
9. Chang, D., Nandi, M.: Improved indifferentiability security analysis of chopMD hash function. In: Nyberg, K. (ed.) Fast Software Encryption – FSE 2008. LNCS, vol. 5086, pp. 429–443. Springer, Heidelberg, Germany, Lausanne, Switzerland (Feb 10–13, 2008)

10. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002), https://doi.org/10.1007/978-3-662-04722-4
11. Dinur, I., Dunkelman, O., Keller, N., Shamir, A.: Cryptanalysis of iterated Even-Mansour schemes with two keys. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 439–457. Springer, Heidelberg, Germany, Kaoshiung, Taiwan, R.O.C. (Dec 7–11, 2014)
12. Dong, X., Sun, S., Shi, D., Gao, F., Wang, X., Hu, L.: Quantum collision attacks on AES-like hashing with low quantum random access memories. In: Advances in Cryptology – ASIACRYPT 2020, Part II. pp. 727–757. LNCS, Springer, Heidelberg, Germany (Dec 2020)
13. Fouque, P.A., Jean, J., Peyrin, T.: Structural evaluation of AES and chosen-key distinguisher of 9-round AES-128. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 183–203. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013)
14. Gérault, D., Lafourcade, P.: Related-key cryptanalysis of midori. In: Dunkelman, O., Sanadhya, S.K. (eds.) Progress in Cryptology - INDOCRYPT 2016: 17th International Conference in Cryptology in India. LNCS, vol. 10095, pp. 287–304. Springer, Heidelberg, Germany, Kolkata, India (Dec 11–14, 2016)
15. Gilbert, H., Peyrin, T.: Super-sbox cryptanalysis: Improved attacks for AES-like permutations. In: Hong, S., Iwata, T. (eds.) Fast Software Encryption – FSE 2010. LNCS, vol. 6147, pp. 365–383. Springer, Heidelberg, Germany, Seoul, Korea (Feb 7–10, 2010)
16. Grover, L.K.: A Fast Quantum Mechanical Algorithm for Database Search. In: Miller, G.L. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996. pp. 212–219. ACM (1996), https://doi.org/10.1145/237814.237866
17. Gérault, D., Lafourcade, P., Minier, M., Solnon, C.: Computing aes related-key differential characteristics with constraint programming. Artificial Intelligence 278, 103183 (2020), https://www.sciencedirect.com/science/article/pii/S0004370218303631
18. Gérault, D., Minier, M., Solnon, C.: Constraint programming models for chosen key differential cryptanalysis. In: Rueher, M. (ed.) Principles and Practice of Constraint Programming. pp. 584–601. Springer International Publishing, Cham (2016)
19. Hirose, S., Ideguchi, K., Kuwakado, H., Owada, T., Preneel, B., Yoshida, H.: A lightweight 256-bit hash function for hardware and low-end devices: Lesamnta-LW. In: Rhee, K.H., Nyang, D. (eds.) ICISC 10: 13th International Conference on Information Security and Cryptology. LNCS, vol. 6829, pp. 151–168. Springer, Heidelberg, Germany, Seoul, Korea (Dec 1–3, 2011)
20. Hosoyamada, A., Sasaki, Y.: Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 249–279. Springer, Heidelberg, Germany, Zagreb, Croatia (May 10–14, 2020)
21. Hosoyamada, A., Sasaki, Y.: Quantum Collision Attacks on Reduced SHA-256 and SHA-512. Cryptology ePrint Archive, Report 2021/292 (2021), https://eprint.iacr.org/2021/292
22. Jaulmes, É., Joux, A., Valette, F.: On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. In: Daemen, J., Rijmen, V. (eds.) Fast Software Encryption – FSE 2002. LNCS, vol. 2365, pp. 237–251. Springer, Heidelberg, Germany, Leuven, Belgium (Feb 4–6, 2002)

23. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) Advances in Cryptology – CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 2004)
24. Jovanovic, P., Luykx, A., Mennink, B.: Beyond $2^{c/2}$ security in sponge-based authenticated encryption modes. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 85–104. Springer, Heidelberg, Germany, Kaoshiung, Taiwan, R.O.C. (Dec 7–11, 2014)
25. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Breaking symmetric cryptosystems using quantum period finding. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 207–237. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2016)
26. Khovratovich, D., Biryukov, A., Nikolic, I.: Speeding up collision search for byte-oriented hash functions. In: Fischlin, M. (ed.) Topics in Cryptology – CT-RSA 2009. LNCS, vol. 5473, pp. 164–181. Springer, Heidelberg, Germany, San Francisco, CA, USA (Apr 20–24, 2009)
27. Knudsen, L.R., Mendel, F., Rechberger, C., Thomsen, S.S.: Cryptanalysis of MDC-2. In: Joux, A. (ed.) Advances in Cryptology – EUROCRYPT 2009. LNCS, vol. 5479, pp. 106–120. Springer, Heidelberg, Germany, Cologne, Germany (Apr 26–30, 2009)
28. Kuwakado, H., Morii, M.: Security on the quantum-type Even-Mansour cipher. pp. 312–316 (01 2012)
29. Kuwakado, H., Morii, M.: Quantum distinguisher between the 3-round Feistel cipher and the random permutation. In: 2010 IEEE International Symposium on Information Theory. pp. 2682–2685 (2010)
30. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound distinguishers: Results on the full Whirlpool compression function. In: Matsui, M. (ed.) Advances in Cryptology – ASIACRYPT 2009. LNCS, vol. 5912, pp. 126–143. Springer, Heidelberg, Germany, Tokyo, Japan (Dec 6–10, 2009)
31. Leander, G., May, A.: Grover meets simon - quantumly attacking the FX-construction. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017, Part II. LNCS, vol. 10625, pp. 161–178. Springer, Heidelberg, Germany, Hong Kong, China (Dec 3–7, 2017)
32. Liu, Q., Zhandry, M.: On finding quantum multi-collisions. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 189–218. Springer, Heidelberg, Germany, Darmstadt, Germany (May 19–23, 2019)
33. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In: Dunkelman, O. (ed.) Fast Software Encryption – FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg, Germany, Leuven, Belgium (Feb 22–25, 2009)
34. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) Advances in Cryptology – CRYPTO'89. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 1990)
35. Meyer, C.H., Schilling, M.: Secure program load with manipulation detection code. In: Proc. Securicom. vol. 88, pp. 111–130 (1988)
36. Minier, M., Solnon, C., Reboul, J.: Solving a symmetric key cryptographic problem with constraint programming. In: ModRef 2014, Workshop of the CP 2014 Conference. p. 13 (2014)
37. Naito, Y., Ohta, K.: Improved indifferentiable security analysis of PHOTON. In: Abdalla, M., Prisco, R.D. (eds.) SCN 14: 9th International Conference on Security

in Communication Networks. LNCS, vol. 8642, pp. 340–357. Springer, Heidelberg, Germany, Amalfi, Italy (Sep 3–5, 2014)
38. Naito, Y., Sasaki, Y., Wang, L., Yasuda, K.: Generic state-recovery and forgery attacks on ChopMD-MAC and on NMAC/HMAC. In: Sakiyama, K., Terada, M. (eds.) IWSEC 13: 8th International Workshop on Security, Advances in Information and Computer Security. LNCS, vol. 8231, pp. 83–98. Springer, Heidelberg, Germany, Okinawa, Japan (2013)
39. National Institute for Standards and Technology, USA: Post-Quantum Cryptography Standardization (2017), https://csrc.nist.gov/projects/post-quantum-cryptography
40. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: Minizinc: Towards a standard cp modelling language. In: International Conference on Principles and Practice of Constraint Programming. pp. 529–543. Springer (2007)
41. Nikolic, I., Wang, L., Wu, S.: Cryptanalysis of round-reduced LED. In: Moriai, S. (ed.) Fast Software Encryption – FSE 2013. LNCS, vol. 8424, pp. 112–129. Springer, Heidelberg, Germany, Singapore (Mar 11–13, 2014)
42. Patarin, J.: A proof of security in O(2n) for the xor of two random permutations. In: Safavi-Naini, R. (ed.) ICITS 08: 3rd International Conference on Information Theoretic Security. LNCS, vol. 5155, pp. 232–248. Springer, Heidelberg, Germany, Calgary, Canada (Aug 10–13, 2008)
43. Pollard, J.M.: A monte carlo method for factorization. BIT Numerical Mathematics 15(3), 331–334 (1975), https://doi.org/10.1007/BF01933667
44. Prud'homme, C., Fages, J.G., Lorca, X.: Choco solver documentation. TASC, INRIA Rennes, LINA CNRS UMR 6241 (2016)
45. Rivest, R.L., Shamir, A.: PayWord and MicroMint: Two simple micropayment schemes. In: International workshop on security protocols. pp. 69–87. Springer (1996)
46. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th Annual Symposium on Foundations of Computer Science. pp. 124–134. IEEE Computer Society Press, Santa Fe, NM, USA (Nov 20–22, 1994)
47. Simon, D.R.: On the Power of Quantum Computation. SIAM J. Comput. 26(5), 1474–1483 (Oct 1997), https://doi.org/10.1137/S0097539796298637
48. Sun, S., Gerault, D., Lafourcade, P., Yang, Q., Todo, Y., Qiao, K., Hu, L.: Analysis of AES, SKINNY, and others with constraint programming. IACR Transactions on Symmetric Cryptology 2017(1), 281–306 (2017)
49. Suzuki, K., Tonien, D., Kurosawa, K., Toyota, K.: Birthday paradox for multi-collisions. In: Rhee, M.S., Lee, B. (eds.) ICISC 06: 9th International Conference on Information Security and Cryptology. LNCS, vol. 4296, pp. 29–40. Springer, Heidelberg, Germany, Busan, Korea (Nov 30 – Dec 1, 2006)
50. van Oorschot, P.C., Wiener, M.J.: Parallel collision search with cryptanalytic applications. Journal of Cryptology 12(1), 1–28 (Jan 1999)
51. Zhou, N.F., Kjellerstrand, H., Fruhman, J.: Constraint solving and planning with Picat. Springer (2015)

A Appendix

A.1 The Rijndael key schedule

The key schedule of AES and Rijndael have same construction. Let $4 \times N$ be the block cipher of $4N$-bit key, and $K_0, K_1, \ldots K_{N-1}$ be the respecting

columns of the original key. Let R be the number of round keys needed, and $WK_0, WK_1, \ldots WK_{4R-1}$ be the columns of expanded key. The expanded key are found by repeating the algorithm of WK_i from $i = 0$ to $4R - 1$ as follow:

$$WK_i = \begin{cases} K_i & i < N \\ WK_{i-N} \oplus SW(RW(WK_{i-1})) \oplus RCON_{i/N} & i \geq N \text{ and } i \bmod N = 0 \\ WK_{i-N} \oplus SW(WK_{i-1}) & i \geq N, N > 6 \text{ and } i \bmod N = 4 \\ WK_{i-N} \oplus WK_{i-1} & \text{otherwise.} \end{cases}$$

where SW and RW are SubWord and RotWord operators acting on 4 bytes which are defined corresponding

$$SW([b_0 \quad b_1 \quad b_2 \quad b_3]) = [SB(b_0) \quad SB(b_1) \quad SB(b_2) \quad SB(b_3)]$$

and

$$RW([b_0 \quad b_1 \quad b_2 \quad b_3]) = [b_1 \quad b_2 \quad b_3 \quad b_0]$$

A.2 The Saturinin key transformation

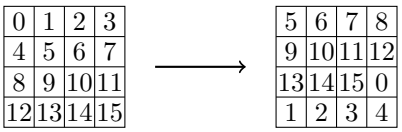

Figure 8: The byte orientation and key transformation of Saturnin

A.3 The differential trails of our attacks

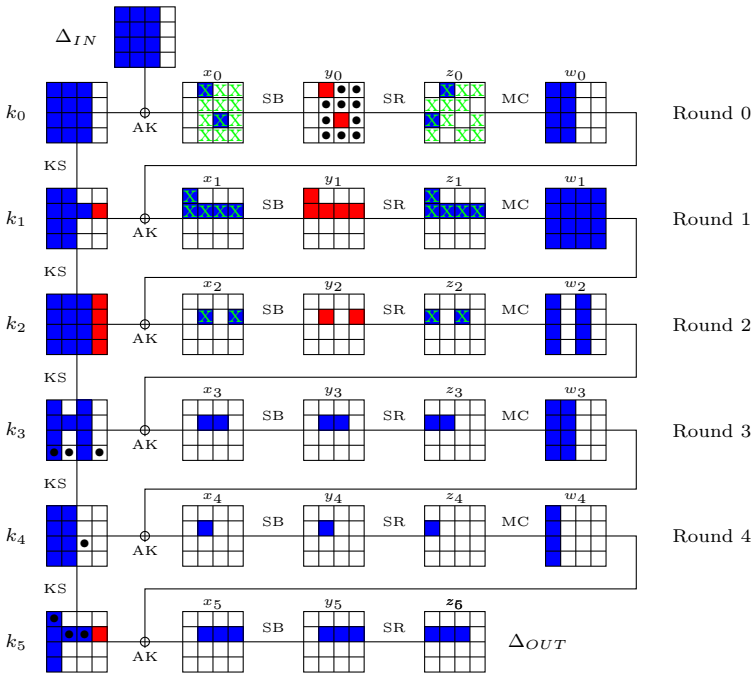

Figure 9: A differential trail for 6-Round AES-128 [48]

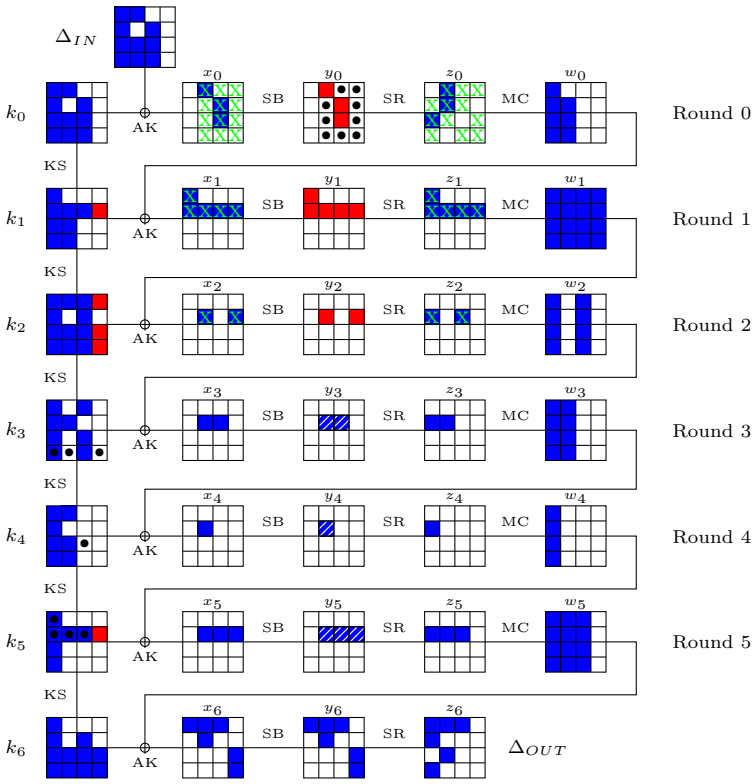

Figure 10: A differential trail for 7-Round AES-128

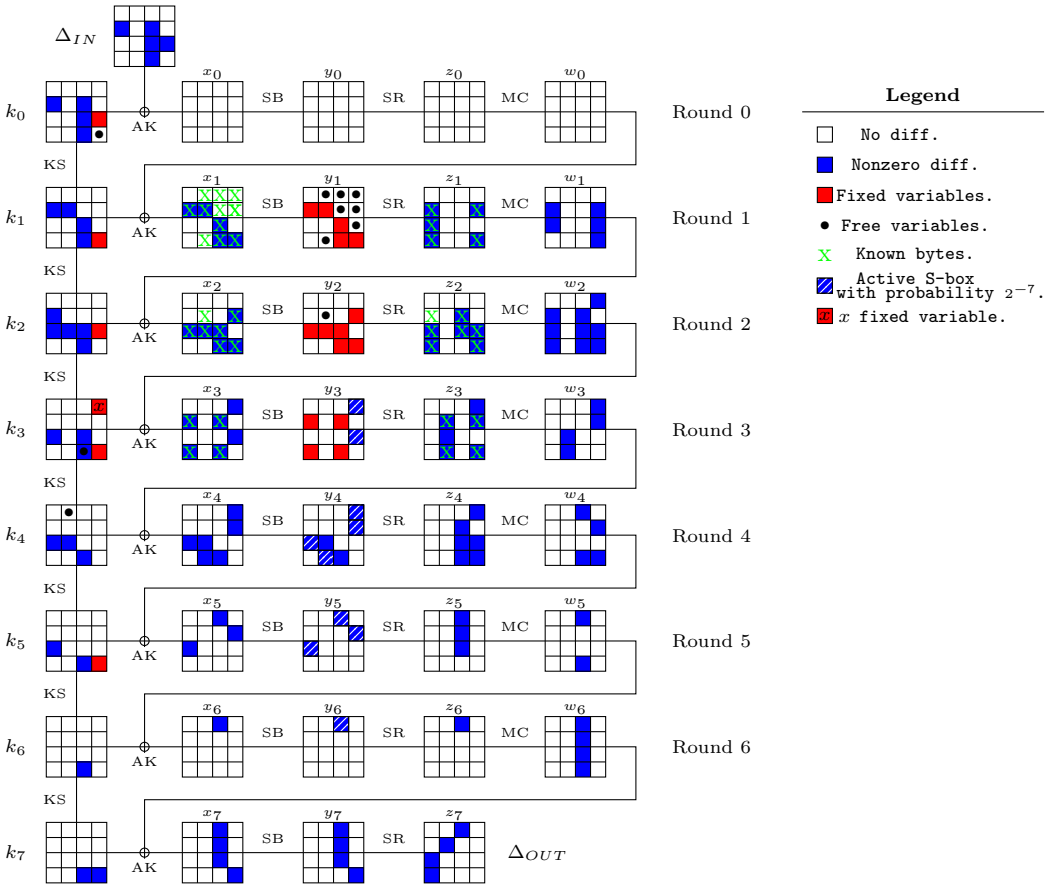

Figure 11: A differential trail for 8-Round AES-128 with variable x

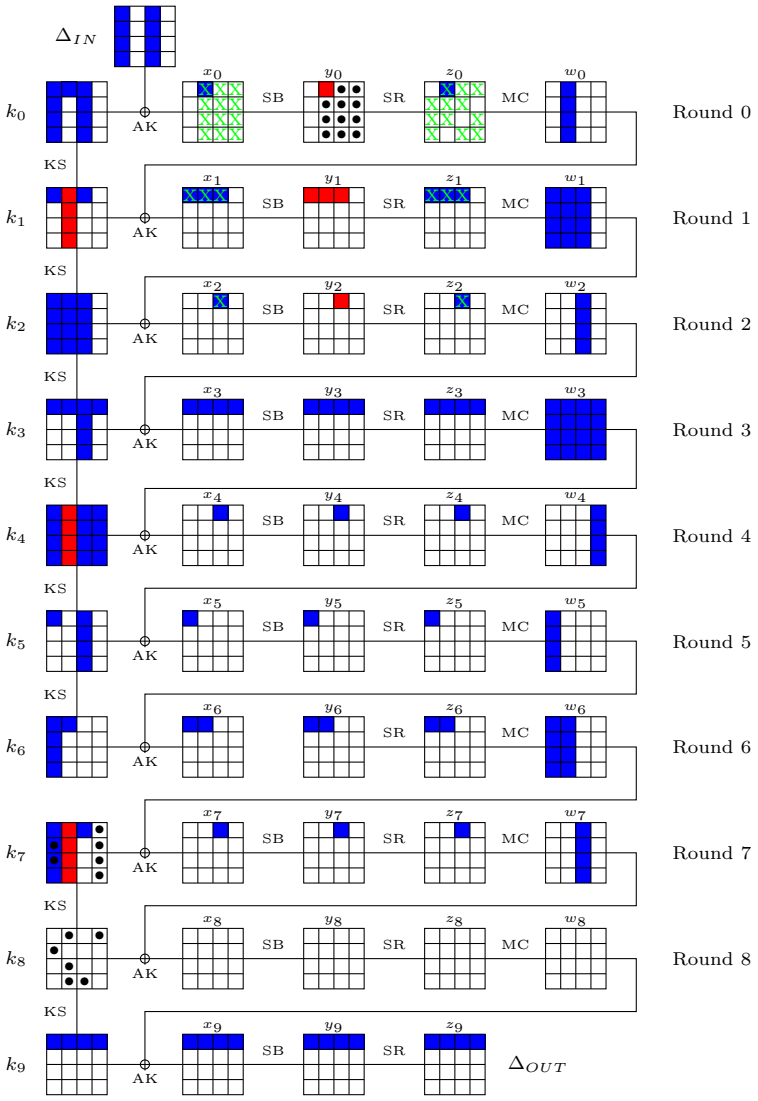

Figure 12: A differential trail for 10-Round AES-192 [17]

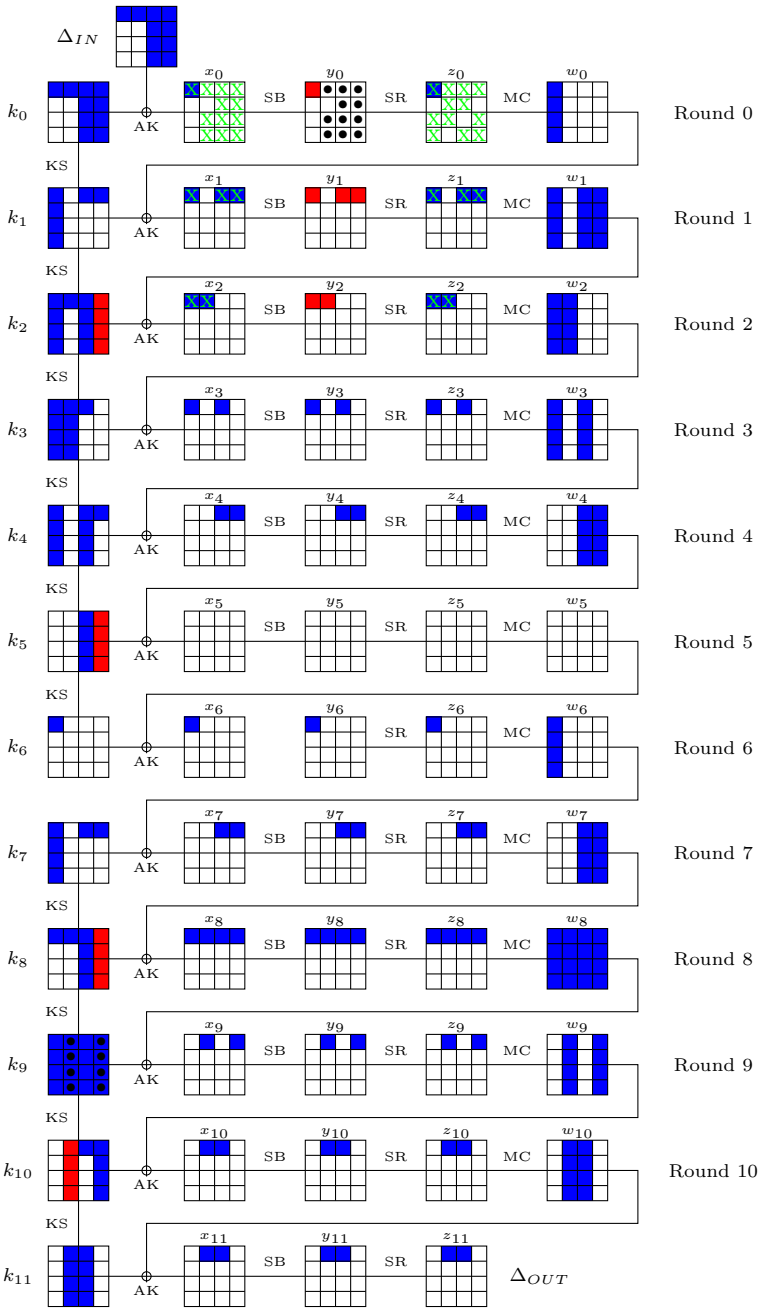

Figure 13: A differential trail for 12-Round AES-192

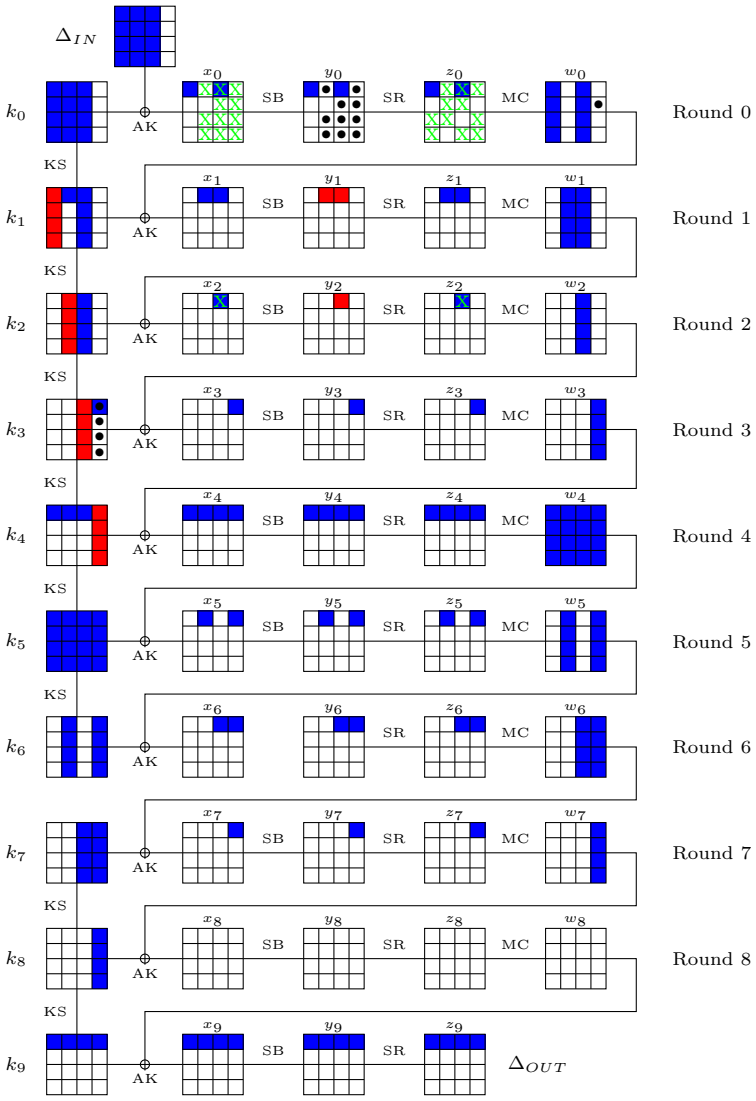

Figure 14: A differential trail for 10-Round Rijndael-128-160

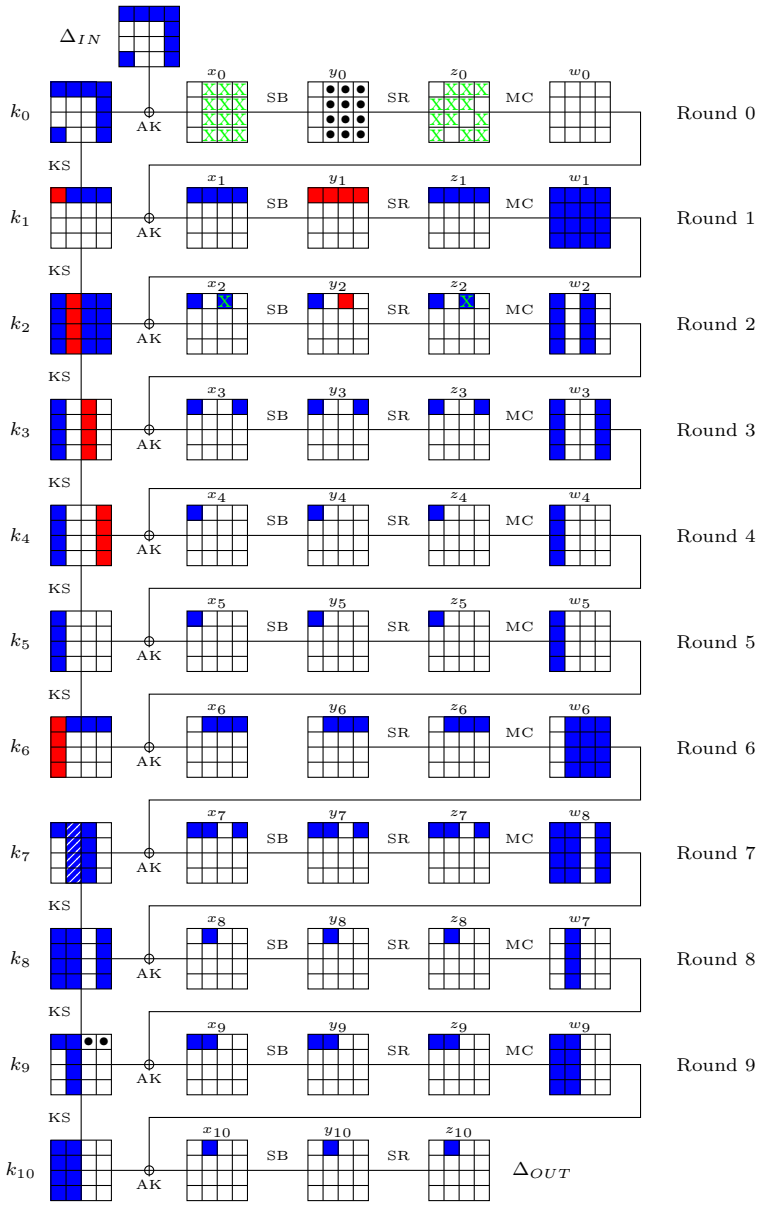

Figure 15: A differential trail for 11-Round Rijndael-128-160

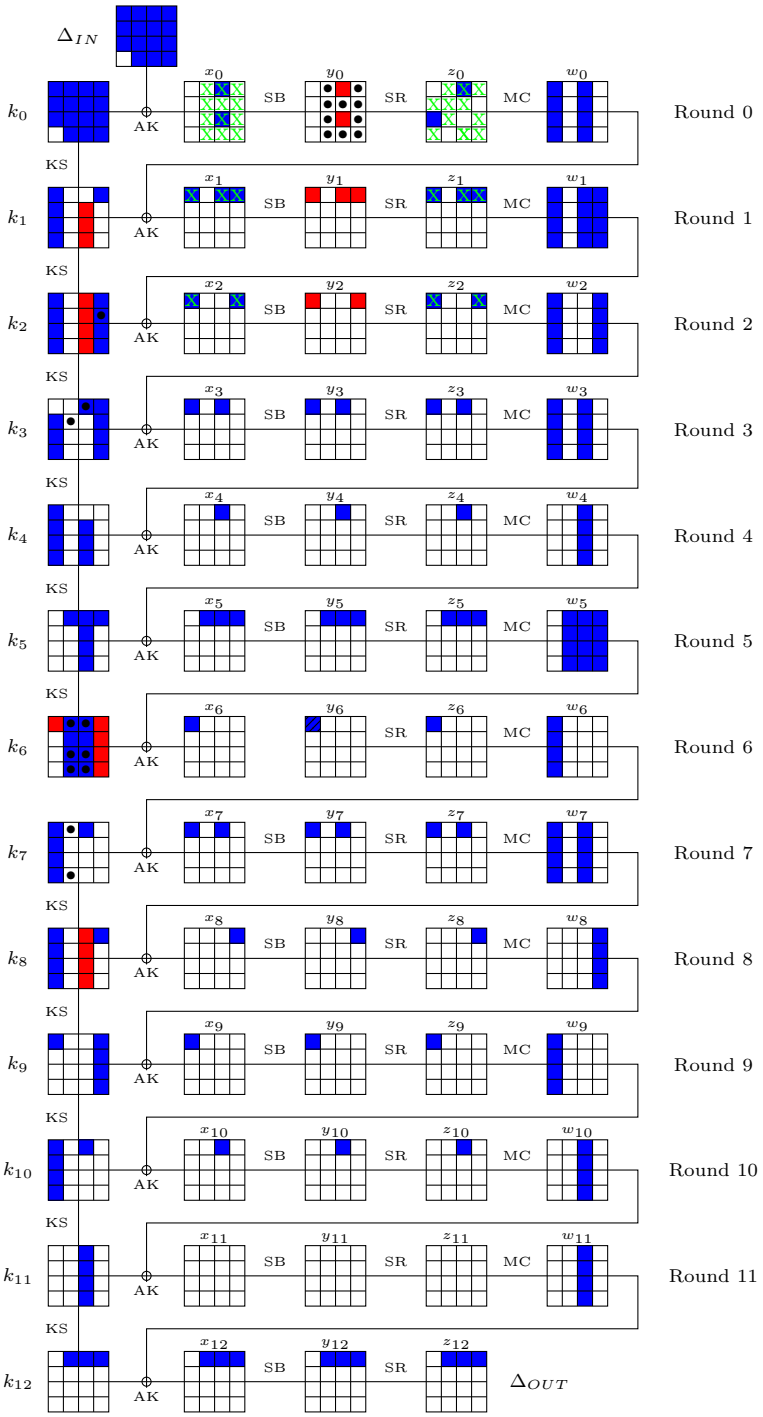

Figure 16: A differential trail for 13-Round Rijndael-128-224

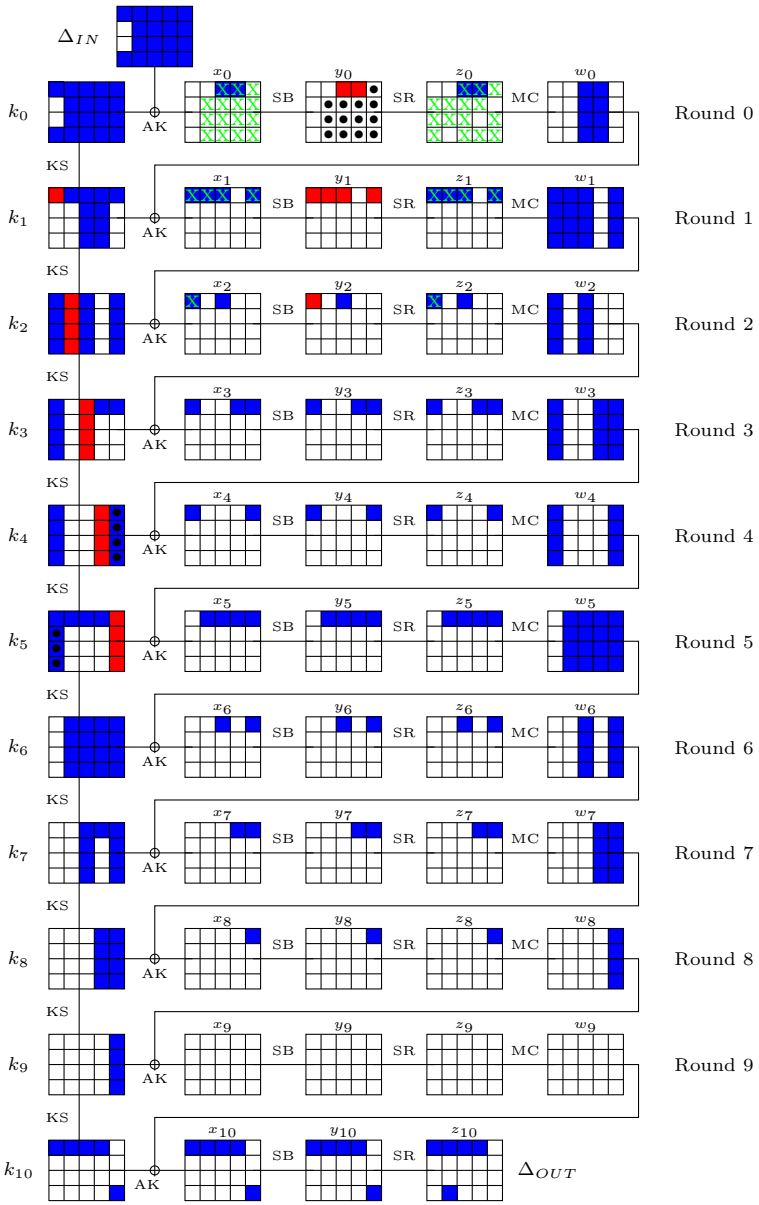

Figure 17: A differential trail for 11-Round Rijndael-160-192

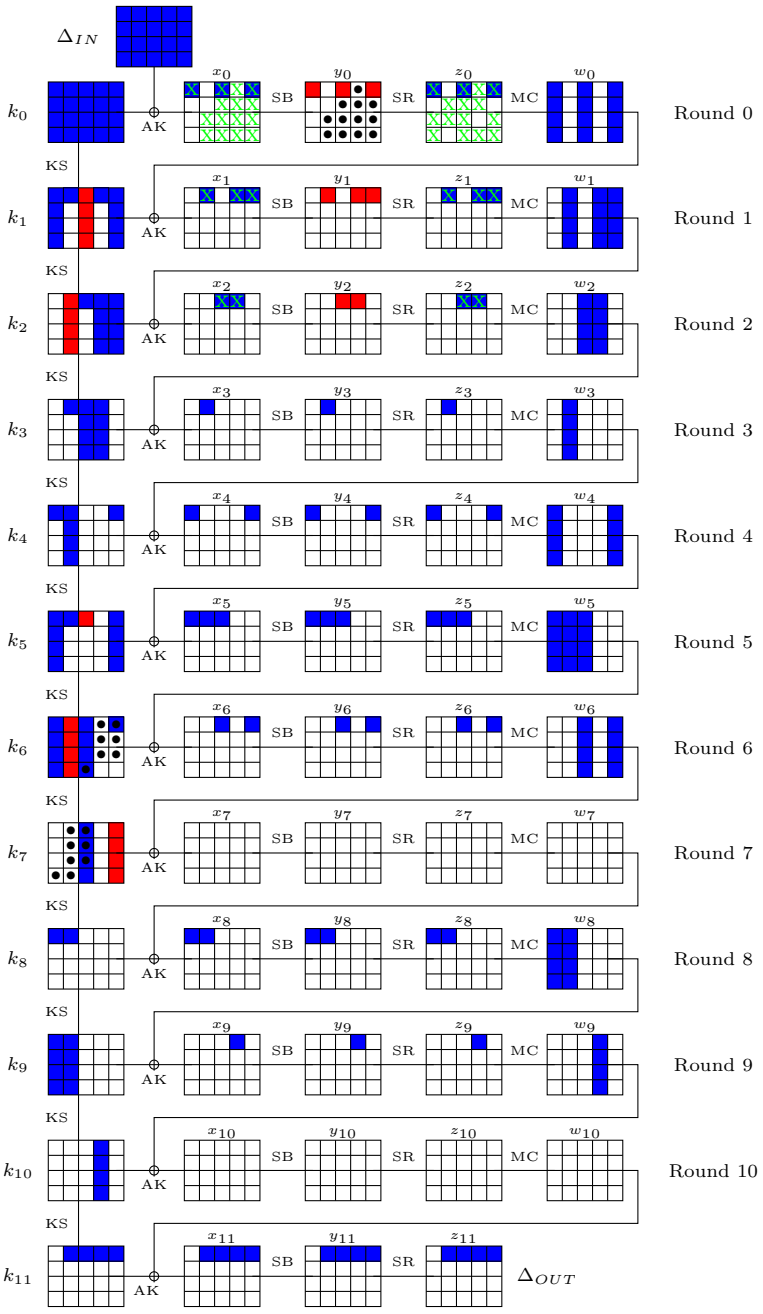

Figure 18: A differential trail for 12-Round Rijndael-160-256

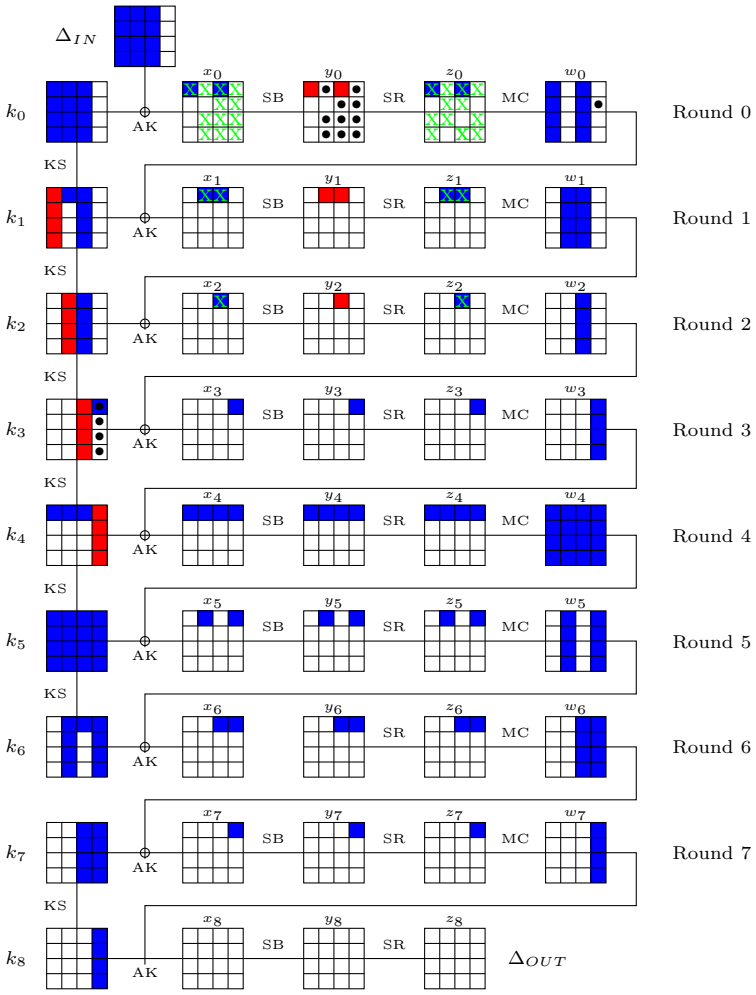

Figure 19: A differential trail for 9-Round Rijndael-128-160

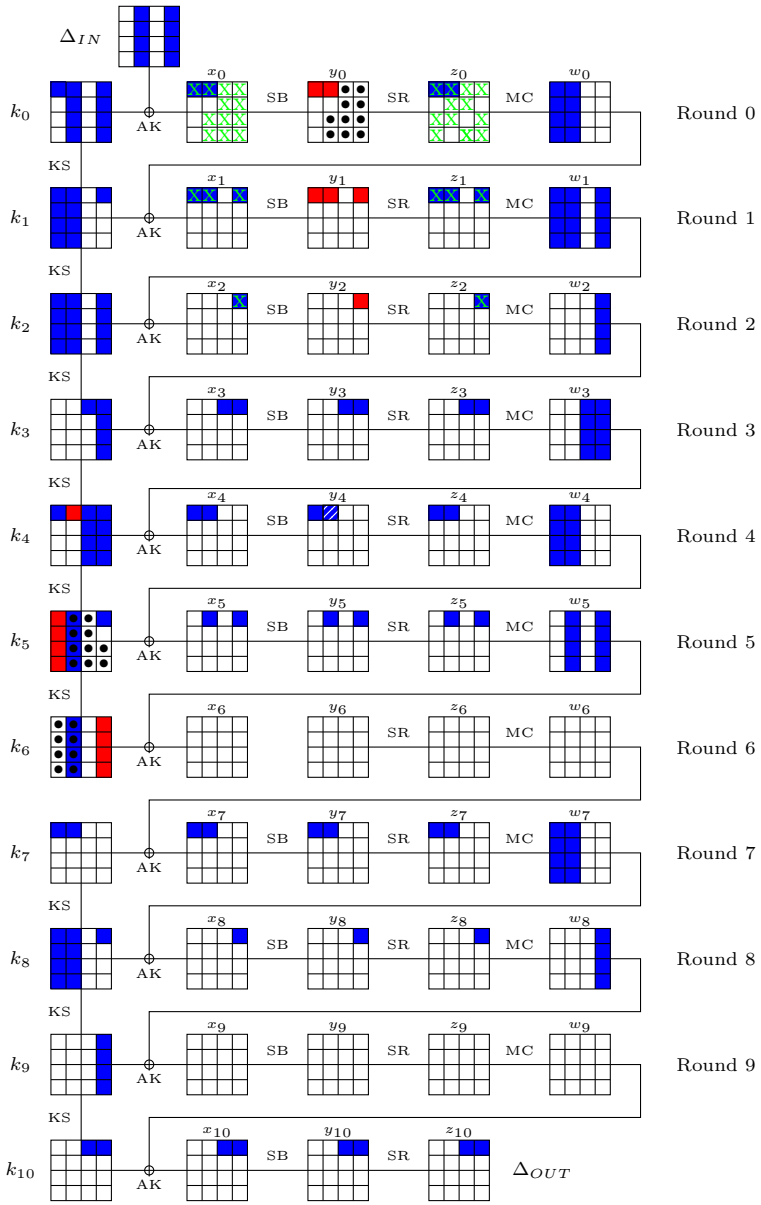

Figure 20: A differential trail for 11-Round Rijndael-128-224