Protocols, cryptanalysis and mathematical cryptology [INFO-F514]

Saturnin: a suite of lightweight symmetric algorithms for post-quantum security

Paper reference (published 2020-06-22)

Cochez Benjamin
Croche Loïc
Hanquin Benjamin Claessens Louis

January 11, 2022

Contents

1 Introduction

With the recent developments in quantum computing research, tech industries are going to face new threats against actual cryptosystems. It is well know that some of them are weak against quantum adversaries because the security strength is sometimes reduced by quantum capabilities and some hard problems such as factorization can be solved in polynomial time with quantum algorithms[1], i.e. Shor's algorithm. In the field of block cipher, Grover's algorithm as an important weight.

Saturnin was designed to answer the need for post quantum cryptographic algorithms in the first part and to provide a secure post-quantum solution for IoT devices in the second part, thanks to its lightness.

Saturnin is a block cipher primitive based on the AES primitive which is supposed more efficient in terms of post quantum security and in terms of CPU usage and time. It's a new algorithm submission and therefore few cryptanalysis was applied on it. Nevertheless, since it is based on AES, it is arguable that they share a lot in common in terms of cryptanalysis, and we know that AES has been heavily reviewed and tested by cryptanalysts.

This subject is meaningful to us since IoT devices and post quantum cryptography are the most up-to-date concerns. It should be noted that we decided to focus this paper on Saturnin-CTR-Cascade analysis which is the 256-bits block cipher encryption and not on Saturnin-Hash nor Saturnin-Short which are other Saturnin-based implementations for hashing and short message encryption.

2 Security of Saturnin

2.1 Security properties

Since the goal of a block cipher is to look like pseudo-random permutations there are multiple properties that need to be implemented to avoid certain types of attacks. We will make a small reminder of them, see how AES applied them and finally verify that Saturnin applied them all.

There are different important basic security properties such as confusion, diffusion[17], avalanche effect (also known as butterfly effect) and alignement[24].

- Confusion : this operation implies that the relation between the key and the ciphertext is convoluted while still being entangled.
- Diffusion : consist of depleting the plaintext structure over the majority of the ciphertext.
- Avalanche effect : means that by changing only one bit (small change) of the initial input values of the primitive (plaintext, key or nonce, depending on the mode of operation) should produce a radical change in the final output.
- Alignement : this property characterizes the interactions between linear and non-linear layers with respect to the linear and differential propagation. An analysis of Saturnin shows that it is aligned.

To apply these lasts, different techniques are used. We apply nonlinear operations, local and global diffusion, transposition, break rounds symmetry (confusion can be implemented in any of these operations).

- Nonlinear operation : a nonlinear operation breaks the linearity and makes a cipher less prone to linear cryptanalysis and techniques such as Gaussian elimination.
- Local and global diffusion : used for diffusion but on a different scale. Generally, a first local diffusion is made and then a wider diffusion is applied on top of a local one.
- Breaking symmetry : Each round has to be slightly different in order to break symmetry, as the same set of operations is repeated at each round.

With this small reminder done, we can now analyze how AES and Saturnin applied them.

2.1.1 AES

In AES, operations are:

- SubBytes : nonlinear substitution, byte-by-byte substitution according to a S-Box.
- ShiftRows : local diffusion, shift the rows byte from 0 (first row) to 3 (last row) steps.
- MixColumns : global diffusion, combined with the shiftRow operations it diffuses on the columns, so it implies a global diffusion.
- AddRoundKey, breaks the symmetry. Each byte of the state is combined with a byte of the subkey using bitwise xor. The subkey is different for each round and is derived using Rijndael's key schedule.

With these four operations, the diffusion is easy to spot in the ShiftRows and in the MixColumns operations. The confusion is mainly in the AddRound key operation because of the key schedule. Finally the avalanche effect comes from every operations. The avalanche effect has been calculated by Drashti O. Vadaviya[19] in the table following (to achieve good avalanche effect, at least one half of the bits must change with a slight modification, the more the better).

AES										
	KEY					PLAINTEXT				
	1 bit	2 bits	3 bits	4 bits	5 bits	1 bit	2 bits	3 bits	4 bits	5 bits
N°1	51,56	53,13	51,56	40,63	50,78	55,47	49,22	52,34	53,13	52,34
N°2	50	53,91	53,13	62,5	50	58,59	53,13	49,22	49,22	57,81
N°3	55,47	50,78	46,9	52,34	55,47	51,56	55,47	52,34	50	52,34
N°4	52,34	53,91	50,78	57,03	52,34	51,56	53,13	50,78	51,56	50
N°5	53,13	53,91	49,22	53,13	50,78	57,81	57,81	53,91	50	52,34
Mean	52,5	53,12	50,31	53,12	51,87	54,99	53,75	51,71	50,78	52,96

2.1.2 Saturnin

In Saturnin, operations are :

- S-Box Layer : this operation provides a nonlinear permutation. Because Saturnin is based on a 3D nibble block representation, its S-Box is a bit different compared to AES which uses a 2D representation, but the principle is the same.
- Nibble permutation Shift Row : this provides local diffusion, Shift of each sheet and slice independently and in parallel. The choice between the two operations depends on the round number. Slice and sheet terminology are inspired by the Keccak hash function.
- Linear MixColumns : provides global diffusion, composed of 16 linear operations applied to every column in parallel.
- Sub-key addition : breaks the symmetry. Applied on odd rounds, it consists in the XOR of a round constant and either the master key or the rotated master key.

The diffusion and confusion properties of Saturnin have a lot in common with AES. Here the Linear MixColumns and the Nibble permutation give the diffusion while the Sub-key addition helps with the confusion. Regarding the avalanche effect properties, it is slightly less than 50% bits flipped on average. The bits were modified with the key and the nonce, since with a Counter mode of operation the plaintext has no impact on it's encryption. The code used to get these numbers is available on github[21].

Saturnin										
	KEY					NONCE				
	1 bit	2 bits	3 bits	4 bits	5 bits	1 bit	2 bits	3 bits	4 bits	5 bits
N°1	47,5	47,81	50,94	52,5	50,63	48,44	53,12	52,81	48,12	48,12
N°2	50	47,5	40,94	51,25	52,81	46,56	45,62	48,75	52,19	47,5
N°3	47,81	48,75	51,88	54,06	48,75	46,88	53,75	54,06	50,94	44,38
N°4	49,38	49,38	48,12	53,44	49,38	49,38	44,69	51,88	50,94	47,5
N°5	48,75	50,31	49,38	48,44	46,88	45,31	47,5	46,25	44,38	47,5
Mean	48,68	48,75	48,25	51,93	49,69	47,31	48,93	50,75	49,31	47

2.2 Security resilience

2.2.1 Known generic attacks

If we transfer the security claims from AES primitive to Saturnin, we can say that there are no better attacks than the generic attacks on the primitive if we respect minimal properties and requirements of the primitive. In the case of Saturnin, these properties are previously explained. Generally, when a symmetric encryption based on block cipher like AES provides a strong pseudo-randomness in it's encryption and respects these security properties, its security resides only in the key and its mode of operation. In quantum, we are limited because the knowledge about possible attacks is still unclear because quantum analysis is relatively new. Yet, there are some claims on this subject. For classical attacks, there are years of research on AES so it will be simpler to determine possible attacks based on these cryptanalysis. When we talk about symmetric encryption, the idea to be resilient against Grover's alogorithm is to double the key size, but we will see that is not really true in some cases, especially if the primitive or the mode of operation are not intrinsically secure. Now, we will show known generic attacks and later we will talk about more specific attacks about the primitive or the used mode of operation.

1. Known classical attacks

   - Birthday paradox: with the problem of the birthday paradox we can reduce the security of the hash functions at $2^{n/2}$ for the collision resistance. We can say that the security strength for a 256-bits key is 2^{128} bits of security for the collision resistance.

2. Known quantum attacks

   - Grover's algorithm[6,12] : Because of the Grover's algorithm which is used for exhaustive search with a time complexity of $O(\sqrt{N})$, the secuirity is reduced at $2^{n/2}$ for the block cipher encryption and the hash function. This way, if we assume that Saturnin-CTR-Cascade provides a correctly pseudo-randomized encryption, we can say that the security strength for a 256-bits key is 2^{128}.

   - Simon's algorithm[10] : Usage of this algorithm for the period finding can break a lot of modes of operation used in symmetric cryptography if we apply a reduction of this problem. The reduction of Simon's problem can lead to a forgery attack using superposition queries and assuming a linear time complexity in O(n). The affected mode of operation is CBC or authenticated encryption modes like GCM, OCB, CBC-MAC and GMAC. CTR and OFB modes are not vulnerable to this attack. In the case of Saturnin, these modes are not used so we can assume that Saturnin should not be subject to this attack.

   - Deutsch-Josza's algorithm (less generic) : This algorithm can be used by quantum attackers against stream cipher-based and some online modes of operation, like CTR or OFB. Therefore, it's a potential problem for the CTR-Cascade used by Saturnin. An alternative mode of operation is given later. Under online modes we define that all modes where ciphertexts only depending on the current and old plaintexts. For example, C0 depends of P0 and C1 depends of P0 and P1.

   - Birthday paradox: In quantum this paradox affects hash functions and leads to a security of $2^{n/3}$ for the collision resistance. For this reason, if we take the case of Saturnin, the security result for the collision resistance will be $\sim 2^{85}$ bits of security.

2.2.2 The primitive security

Due to the similar structure of Saturnin block cipher and the AES block cipher on 16-bit words, Saturnin claims the benefits from the past decades cryptanalysis background of AES. The claim is that AES attacks can be transposed to Saturnin by replacing the AES 8-bit-S BOX by the Saturnin 16-bit Super-S-Boxes. We have to trust the claim for this section.

The claims on Saturnin cryptanalysis will be compared to the well known AES results to check if Saturnin is stronger, or equivalent to AES in terms of primitive. Multiple attacks will be tested against Saturnin such as differential and linear cryptanalysis. AES has proven his worth against these attacks, Saturnin being based on AES has to be equivalently resistant, if not stronger to these attacks.

Differential Cryptanalysis Differential cryptanalysis refers to the study of how much impact has an input difference on the output. This is, in the case of a block cipher, tracing the path through the rounds and looking for non procedural behavior in order to exploit this weakness and recover the secret key. The metric used in differential cryptanalysis is the time complexity of the attack, this is what we will try to compare next between Saturnin and AES.

By using differential cryptanalysis on Saturnin, the authors stated that : ”for 4 super-rounds, the best differential characteristics have at most probability $2^{-241.9}$. This probability is of $2^{-483.9}$ for 8 super-rounds”.

There is no context from the number given by the Saturnin team. There is no explanation on what submethod is used (Chosen plaintext or Chosen difference), what are the calculations that led to these results, and why the results are not expressed in time complexity but in “best differential characteristics”.

Most of the literature on this subject express the differential cryptanalysis in terms of complexity and for no more than 4 rounds. So it is really hard to compare the number given by the Saturnin team to existing AES cryptanalysis.

For example AES with 4 rounds, the complexity is varying from $= 2^{52}$ to 2^{55} [22] depending on the method used to do the cryptanalysis (chosen plaintext or chosen difference).

Linear Cryptanalysis Linear cryptanalysis is used to make a linear approximation of the ciphering algorithm by simplifying it. The approximation will be perfected by testing all the formulae possibilities and finding the one that fits, the one that can represent the nonlinear operations as close as possible. This technique is much more efficient on vulnerable block-ciphers than differential cryptanalysis.

The chosen Walsh transform (linear approximation equation) of the Saturnin Super-S-Box is the following :

$$\widehat{\mathsf{S}_{16}}(\alpha,\beta) = \sum_{x\in\mathbb{F}_2^{16}} (-1)^{\beta\cdot\mathsf{S}_{16}+\alpha\cdot x}, \quad \alpha,\beta \in \mathbb{F}_2^{16}, \beta \neq 0 \ .$$

By using linear cryptanalysis on Saturnin, the authors show that for 4 super-rounds, the linear potential is at most $2^{-220.7}$ and at most $2^{-441.5}$ for the 8 super-rounds method. We have the same problem here, the numbers given by the teams come out of nowhere and have no context or prior explanation in the paper. The linear cryptanalysis of AES shows that linear cryptanalysis is completely useless because the S-Boxes of AES are insensible to a linear approximation. This kind of attack is so much more complex and time consuming than a basic exhaustive search.

Impossible Differential Cryptanalysis The Impossible Differential cryptanalysis goal is to the contrary of basic differential cryptanalysis not to examine the difference in the behaviour of the process between inputs but to find a behaviour in the process that is impossible with a good key in order to reject that key. The paper sets the best impossible differential attack over AES-128 7-rounds to be 2^{105} chosen plaintext (offline complexity), $2^{106.88}$ time complexity and 2^{74} memory complexity. From our knowledge[15] the numbers are for 7 rounds, $2^{106.2}$ chosen plaintext, $2^{110.2}$ time complexity and $2^{94.2}$ memory complexity. Those numbers are quite close to the ones given in the paper except for the memory complexity that is a whole 20 exponent higher. We were not able to find the paper with the exact number given here. The claim from the Saturnin team is that their cipher is, for 7 super-rounds and similar attacks, double the exponent values of the AES-128 bit. It means the best differential attack over Saturnin will be 2^{210} chosen plaintext (offline complexity), $2^{213.76}$ time complexity and 2^{148} memory complexity. Such claims can't be verified since there is no methodology nor the computation and development that led to this result. The only way for us to verify this would be to make our own Impossible Differential Cryptanalysis or to find a paper that does this for us, but unfortunately we did not find such a paper.

Quantum attacks The paper claims that there exists no better quantum attacks than classical attacks on either AES or Saturnin that the generic ones explained earlier in this paper. They even claim that these attacks could take even more time than classical ones in the current shape of the field.

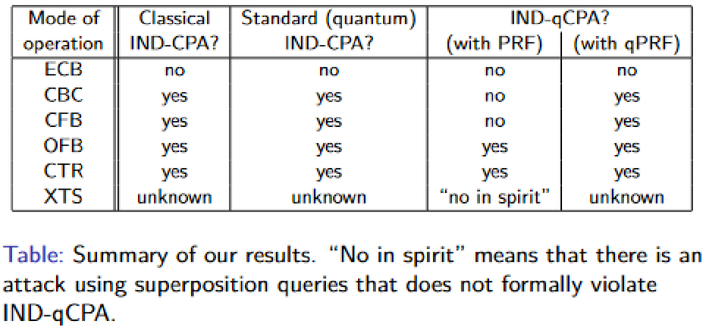
2.2.3 The mode of operation security

Saturnin-CTR-Cascade The Saturnin structure for the mode of operation is based on different properties which are already proven secure and efficient for block cipher encryption, even in the post quantum environment. These properties are :

1. The CTR mode for encryption

   The CTR is a well known mode of operation for block cipher encryption. CTR mode requires two important things : a nonce so that the scheme is not deterministic and a counter to ensure that each encryption with the same nonce is different. CTR has some interesting properties:

   - In terms of IND-CPA (INDistinguishability under Chosen-Plaintext Attack) and IND-qCPA (in quantum) security, which are important security proof methods for the ciphertext indistinguishability, a comparison of some useful modes of operation has been determined in this following comparison table[1].



Mode of operation	Classical IND-CPA?	Standard (quantum) IND-CPA?	IND-qCPA? (with PRF)	(with qPRF)
ECB	no	no	no	no
CBC	yes	yes	no	yes
CFB	yes	yes	no	yes
OFB	yes	yes	yes	yes
CTR	yes	yes	yes	yes
XTS	unknown	unknown	"no in spirit"	unknown

Table: Summary of our results. "No in spirit" means that there is an attack using superposition queries that does not formally violate IND-qCPA.

   N represents the Nonce and it is concatenated with the counter, k is the secret key, E is the block cipher encryption with Saturnin primitive, m0 to ml is the message blocks and c0 to cl is the cipher blocks.

   - CTR mode can be computed in parallel, so it is more efficient in terms of speed. In addition, another variant for Saturnin which uses QCB also allows this parallelism property.

- On contrary with CBC mode, CTR mode avoids propagation of errors because it is not based on a chaining method. Therefore, CTR can be tolerant to a loss of block messages.

  This figure[4] shows the processing of encryption for Saturnin with the CTR mode:

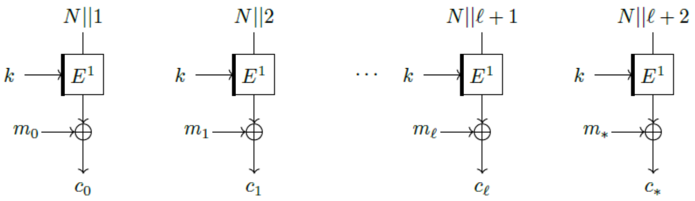
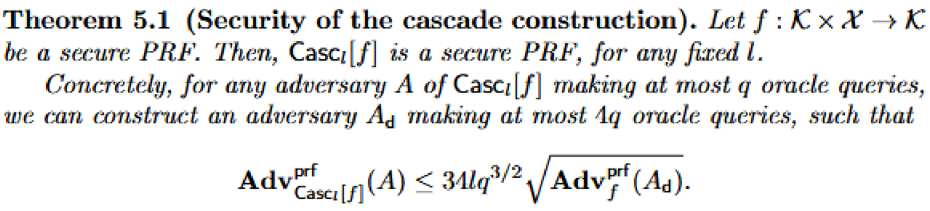


Figure 1: CTR mode

  N represents the Nonce and it is concatenated with the counter, k is the secret key, E is the block cipher encryption with Saturnin primitive, m0 to ml is the message blocks and c0 to cl is the cipher blocks.

2. The Cascade MAC structure for authentication
   The Cascade MAC is similar to the Nested MAC (NMAC) construction[8] of Mihir Bellare, Ran Canetti, and Hugo Krawczyk which implement a cascade construction. NMAC is already used in the well known HMAC construction. In addition, the Cascade MAC uses the Matyas-Meyer-Oseas (MMO) mode which is based on the Merkle-Damgard construction[7] to provide the Indifferentiability property. We assume that Saturnin is used as a PRF in the Cascade MAC and blocks length are fixed. With this assumption we can give a security definition for the cascade construction with the following theorem provided by Song and Yun[9]:



Theorem 5.1 (Security of the cascade construction). *Let $f : \mathcal{K} \times \mathcal{X} \to \mathcal{K}$ be a secure PRF. Then, $\mathsf{Casc}_l[f]$ is a secure PRF, for any fixed l.*

Concretely, for any adversary A of $\mathsf{Casc}_l[f]$ making at most q oracle queries, we can construct an adversary A_d making at most $4q$ oracle queries, such that

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{Casc}_l[f]}(A) \leq 34lq^{3/2}\sqrt{\mathbf{Adv}^{\mathsf{prf}}_{f}(A_\mathsf{d})}.$$

   Due to this indifferentiability property provided by MMO mode and the usage of cascade construction, the possible known attacks are by the way mitigated. We can highlight two main attacks:

   - The Collision attack : we can say that the MMO mode provides a collision resistance except if the block cipher is not secure. If we assume that Saturnin is intrinsically secure, then the Cascade-MAC of Saturnin is also. Therefore, the probability to get a collision is up to 2^{128} according to the birthday paradox.

   - The Extension attack : This attack is not possible on Saturnin-CTR-Cascade because the indifferentiability property provides protection against forgeries. To succeed in this attack, one must recover the secret key.

   Classically, this construction ensures SUF-CMA (Strong existential UnForgeability under Chosen Message Attack) security. This kind of security proof is similar to the IND-CPA but focusing on the signature. It means that if the MAC construction is SUF-CMA secure, the generated MAC is almost unforgeable.

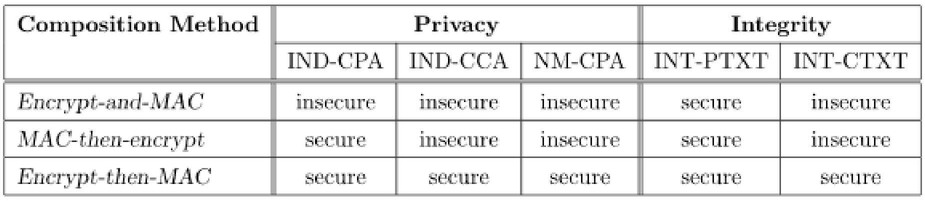
3. Encrypt-then-MAC

In authenticated encryption schemes, there are 3 types of methods : Encrypt-and-MAC, MAC-then-encrypt or encrypt-then-MAC. A comparison table[3] has been provided where we can see which methods allow to be IND-CPA and IND-CCA secure.

We can say that the encrypt-then-MAC method is the best one because it's the only one which provides IND-CPA and IND-CCA securities.

In addition, this method can verify the MAC before decrypting the ciphertext, so it can reject forgeries and be sure that the plaintext has not been recovered before the verification of the MAC.

Quantumly, we can claim that encrypt-then-MAC allows a IND-qCCA security[11] according to the theorem of Soukharev, Jao and Seshadri which explain that if the encryption scheme is IND-qCPA secure and the MAC is SUF-qCMA secure, then the authenticated scheme using an encrypt-then-MAC method is IND-qCCA secure.



Composition Method	**Privacy**			**Integrity**	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

Conclusion for the Saturnin-CTR-Cascade mode With all of these properties, we can deduce that this mode of operation for the authenticated encryption in block cipher is IND-CCA secure and even IND-qCCA secure according to Bellare and Namprempre theorem represented in following figures[23] :

IND-CPA (Enc) and SUF-CMA (MAC) $\Longrightarrow$ IND-CCA (AE)
IND-qCPA (Enc) and SUF-qCMA (MAC) $\Longrightarrow$ IND-qCCA (AE)

Saturnin-QCB[5], an alternative to Saturnin-CTR-Cascade Another mode of operation for Saturnin has been highlighted by Saturnin's authors in an updated article. This mode of operation is called "QCB" and it is inspired by the TAE and OCB modes with a post quantum security. Security proofs of QCB have been performed by explaining that this mode of operation contains the same security properties as Saturnin-CTR-Cascade namely IND-CCA and IND-qCCA security. Saturnin-QCB has been chosen to optimize the speed of computation in the mode of operation, increasing the parallelism property thanks to the OCB mode and reducing the number of necessary super-rounds. The number of super-rounds in Saturnin-CTR-Cascade is 10 + 10 = 20 but in Saturnin-QCB, this number drops to 16. Moreover, QCB provides additional quantum security proofs which leads to a more reliable post-quantum mode of operation because it offers a simplification in security proofs.

To go into more details, OCB-based schemes, including TAE, are intrinsically not quantum secure against Simon's algorithm (superposition queries). To render OCB quantum secure, a solution is to combine it with a Tweakable Block Cipher (TCB), a construction similar to the TAE mode and transform the tweakable block cipher to be resistant to quantum attacks. This combination is implemented in the QCB mode which uses the key-tweak insertion Tweakable Block Cipher,offering a quantum-secure tweakable block cipher. QCB seems to be reliable because it is based on good security proofs for the indistinguishably and unforgeability which are the IND-qCPA security and the BZ-unforgeability[15] (provides the EUF-qCMA security), but it is relatively recent and not much reviewed. Some cryptanalysis of this mode should be a good compromise to enforce the security claims. A recent proposition from Chevalier, Ebrahimi and Vu explains the concept of qIND-qCPA security[13]. The quantum indistinguishability notion is based on a quantum oracle proposed by Zhandry. This notion of security states that some modes of operation like CFB, CTR and OFB are broken with a quantum oracle. Moreover, even the GCM mode for authenticated encryption is broken in this notion. Even if CTR is IND-qCCA secure, it is not qIND-qCPA secure because the usage of stream cipher like in CTR or all online modes of operation do not provide the quantum indistinguishably. To exploit these modes of operation insecure under qIND-qCPA definition, the Deutsch-Josza's algorithm is powerful, even against modes defined as IND-qCPA secure but this attack can not be achieved on CBC and QCB. For the moment, QCB authors assume the insecurity of qIND-qCPA because there isn't yet a fixed definition and there is not a lot of literature on this subject.

The following pictures show the QCB construction:

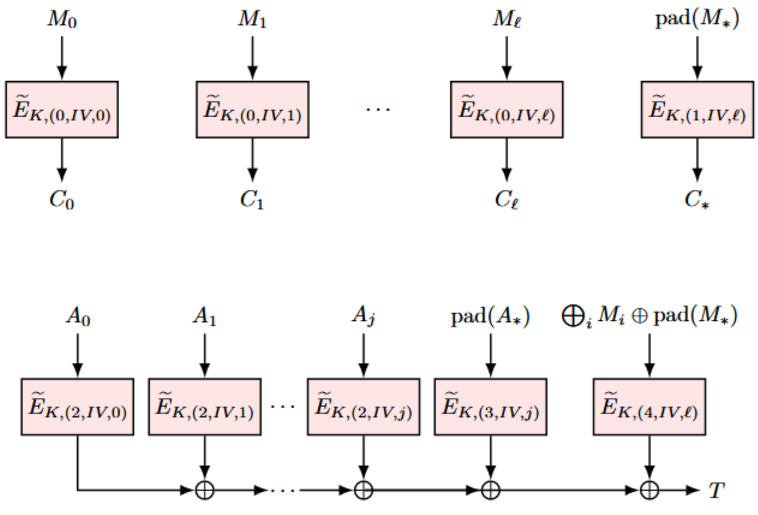

M_0
M_1
M_ℓ
$\mathrm{pad}(M_*)$
$\widetilde{E}_{K,(0,IV,0)}$
$\widetilde{E}_{K,(0,IV,1)}$
$\cdots$
$\widetilde{E}_{K,(0,IV,\ell)}$
$\widetilde{E}_{K,(1,IV,\ell)}$
C_0
C_1
C_ℓ
C_*
A_0
A_1
A_j
$\mathrm{pad}(A_*)$
$\bigoplus_i M_i \oplus \mathrm{pad}(M_*)$
$\widetilde{E}_{K,(2,IV,0)}$
$\widetilde{E}_{K,(2,IV,1)}$
$\cdots$
$\widetilde{E}_{K,(2,IV,j)}$
$\widetilde{E}_{K,(3,IV,j)}$
$\widetilde{E}_{K,(4,IV,\ell)}$
T

Figure 2: QCB construction

In the case of Saturnin-QCB, the idea is to transform the block cipher into a Tweakable Block Cipher using the key-tweak insertion with the following equation : $\vec{E}_{k,t,D}(x) = E^D_{k\oplus t}(x)$ where t is the tweak, a concatenation of the IV and the block number which provides a 256-bit value, k is the secret key and D is the domain. Some assumptions : we can note that the usage of the IV inside the tweak in this mode is indispensable to avoid a possible quantum attack based on Deutsch-Josza's algorithm. Moreover, we have to assume that the nonce is unique and not reused and that Saturnin-QCB continues to use additional data. The following picture shows the Saturnin-QCB encryption:

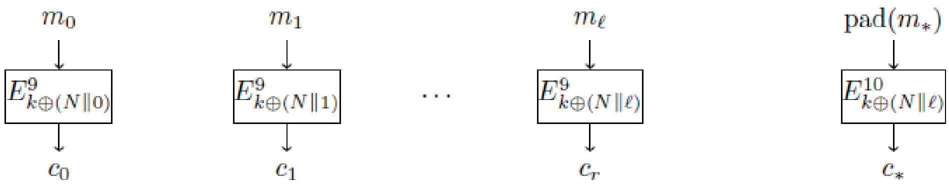


Figure 3: Saturnin-QCB

3 Effectiveness of the primitive

In order to assess the time complexity of Saturnin we can estimate the number of operations in software or estimate the number of gates in a hardware setting. For the software setting we consider as in the original paper a 16-bit instruction set with three-operand instructions.

3.1 Theoretical comparison with AES effectiveness

3.1.1 Time complexity analysis

The complexity of each function in the AES primitive is in O(1) and the mode of operation has a complexity of O(N), with N the message's size. For Saturnin it is the same observation for the primitive and for the mode of operation. The time complexity analysis doesn't give a good comparison as often it isn't relevant to block ciphers. We don't have any information about hardware like the FPGA implementation of Saturnin, so it's difficult to have a comparison on the number of gates in an optimized circuit. However, it's possible to give an approximation in terms of software implementation.

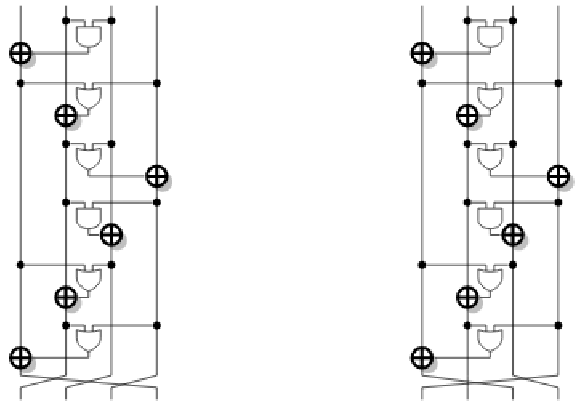
3.2 Operation count

3.2.1 S-box layer

The S-Boxes can be implemented efficiently with a network of 12 basic gates (6 XOR, 2 AND and 4 OR) applied on each nibbles of the block. This gives us $\frac{256}{4} \times 12 = 768$ gates.

Counting the number of instructions with 16-bit registers gives us $\frac{\frac{256}{16}}{4} \times 12 = 48$ instructions.

See figures (a) and (b) bellow, two efficient hardware implementation of the first and the second S-box.



(a) σ_0 (b) σ_1

Figure 4: Implementation of the S-box layer

3.2.2 Permutation and inverse permutation

This operation depends on the parity of the round number r. For all even rounds the permutation is the identity function.

For odd rounds with $r \mod 4 = 1$, the operation SR_{sheet} is applied. We can apply to each register i a rotation of each of it's nibbles by $\lfloor i/4 \rfloor$. It can be express with C macros as such:

$$r = ((x \ \& \ 0x7777) \ << \ 1)| \ ((x \ \& \ 0x8888) \ >> \ 3))$$

This accounts for 2 AND, 2 SHIFTS and 1 OR for each of the 12 registers which makes a total of 60 instructions. For odd rounds with $r \mod 4 = 3$, the operation SR_{slice} is applied. It can be done by a rotation of each of the i register by $4 \times \lfloor i/4 \rfloor$. This accounts for 12 rotations instructions.

As for 1 round, we have $P[0\ instruction] = \frac{1}{2}$, $P[12\ instructions] = \frac{1}{4}$, $P[60\ instructions] = \frac{1}{4}$, we have on average 18 instructions per round for the permutation layer. As there is an inverse permutation layer, this gives us 36 instructions.

In a hardware setting, we can simplify the cost of permutation to zero.

3.2.3 MDS layer

Our MDS is a lightweight MDS matrice discussed in *MDS matrices with lightweight circuits* from Sébastien Duval and Gaëtan Leurent[20]. It costs 38 XOR instructions. In hardware it costs 16×38 XOR gates.

3.2.4 Subkey addition

The subkey addition is only applied at odd rounds. Furthermore, the subkey addition depends on ($r \mod 4$).
To forme the subkey while $r \mod 4 = 1$ the RC_s are XORed with the master key, otherwise they are XORed with $ROT_{20}(MasterKey)$

As the subkey addition is a simple XOR applied on the internal state, we have 256 bits to XOR with 16-bit constants on 16-bit registers it gives us 16 instructions. In a hardware setting we would need 256 XOR gates.

3.2.5 Rounds Constants

The key schedule is ignored in the original paper and not taken into account while assessing the efficiency of Saturnin.
RC_s are updated by clocking 16 times two LSFR in Gallois mode with independent feedback polynomial. They can be implemented in C macro as such:

$$\#define\ LOCK\_LFSR(x) \quad x = ((x)\ \&\ 0x8000)\ ?\ ((x)\ <<\ 1) \wedge 0x002d\ :\ (x)\ <<\ 1$$

This transformation gives us 3 or 4 instructions to be repeated 16 times for each RC.

3.2.6 Summary

Our total number of instructions or gates /round = S-Box + Permutations + MDS + Subkey addition.
Which gives us : $48 + (18 \times 2) + 38 + (16/2) = 130$ instructions or $768 + 608 + (256/2) = 1504$ gates.

Our computations match exactly the computation of the paper :

	our #instructions	reported #instructions	our #gates	reported #gates
1 round	130	130	1504	1504
20 rounds	2616	2616	30336	30336

For the key schedule we have between 48 and 64 instructions per round, depending on the state of the LSFR.

3.3 Practical comparison with AES effectiveness

3.3.1 Bias

First of all, we assume that implementation of Saturnin and AES isn't optimized because tests are made with code written in C and not in assembly for a specific processor. Performance could be even increased using a dedicated FPGA (Field-Programmable Gate Array). We choose this implementation because it's simpler to implement and more portable if we want to try the code on several processors. Moreover, a lot of CPUs optimize AES implementations and it can lead to important improvements of performance.

Second, another assumption is that the Saturnin code is not optimized as much as it could be. Saturnin is not a standard yet and it's a new primitive, so there isn't much work on it. This way, it's really probable that the code can be improved. For AES, there exists a lot of implementations which are optimised in C code.

Third, we assume that modes of operation are different because the CTR-Cascade used in Saturnin is not implemented in AES. This mode of operation was designed to respect as much as possible the post-quantum resistance. To demonstrate a general comparison, we will take some implementation of AES with different relevant modes of operations but we must keep in mind that it is not the same mode of operation.

Lastly, we assume that in general the operating system as well as material (CPU, GPU, RAM,..) and many other elements might affect our tests in an unpredictable way.

3.3.2 Code implementation

For this part, we used the code implementation of Saturnin's designers proposed for the NIST competition. To compare Saturnin, we decided to choose two implementations of AES for authenticated encryption namely, AES-GCM and AES-CCM. Both these implementations are accessible on the OpenSSL wiki and available here. For our tests, we use an Intel Core I7-7700HQ processor and we run implementations on an Ubuntu 20.10. For the compilation we use GCC. To obtain better reliability in our tests we try each implementation encrypting two different text sizes. The first text has a size of 10 kilobytes and the second one has a size is 1 megabyte. To determine which one has the best performance, we compare all encryption and decryption in terms of time, more precisely in milliseconds. To optimize our results, we made a loop to encrypt and decrypt 1000 times plaintexts and ciphertexts respectively, then we took the average time for each encryption/decryption.

We can notice that the key, IV, additional data and plaintexts are always the same for each implementatiosn and we did not generate them randomly, all is hardcoded. Our tests and configurations are available on github.

After the tests has been performed, results given in table show that Saturnin-CTR-Cascade implementation takes more time to encrypt and decrypt than AES-CCM or AES-GCM. The results also show that AES-GCM has the best performance. These results can be explained by the several bias clarified just before. However, the results seem coherent for AES because AES-GCM has better performance than AES-CCM.

3.3.3 Comparative table

	10KB Message		1MB Message	
Compiled with GCC	Encryption(ms)	Decryption (ms)	Encryption (ms)	Decryption (ms)
Saturnin-CTR-Cascade	3.090274	3.136383	290.427258	293.453288
AES-GCM (256)	0.014091	0.015645	1.649558	1.623063
AES-CCM (256)	0.021118	0.023108	2.526605	2.558232

4 Conclusion

We saw that Saturnin is relatively new in the world of post-quantum algorithms for symmetric encryption and current knowledge in quantum does not allow us to be totally confident that Saturnin will be resilient in front of possibly new quantum algorithms. However, at this time, Saturnin is theoretically resistant against known attacks if we respect the minimum properties claimed according to the Bellare and Namprempre theorem. Even if the definition of qIND-qCPA security is recent, we can conclude that it's preferable to use the QCB mode with Saturnin to avoid the possible problem related to the CTR mode which can be exploited by a quantum attack based on the Deutsch-Josza's algorithm but not on QCB.

Moreover, Saturnin has the needed properties of a block cipher, it contains confusion, diffusion and has also the avalanche effect, even if this last is slightly lower than in AES. It also has tools to break symmetry and linearity.

Furthermore, the comparison with other block ciphers with the same mode of operation, and more precisely with AES, is a bit difficult because Saturnin has not been implemented in a lot of applications and AES has not been implemented with this type of mode of operation. The only concrete comparison is based on the theoretical comparison like the number of operations in each primitive, where we can say that Saturnin is better. Still, in our biased practical tests AES-GCM is the better one.

We can conclude that Saturnin is promising as a post quantum algorithm in the field of symmetric cryptography and in terms of weight. We should wait to see whether it will be standardized by the NIST institution in the coming years or not. We should continue to perform research on the proprieties of Saturnin and other primitives relatively to quantum attacks.It is also important to continue research on both CTR-cascade and QCB relatively to post-quantum techniques.