Towards Post-Quantum Secure Symmetric Cryptography: A Mathematical Perspective

Xenia Bogomolec, Quant-X Security & Coding, Hanover
John Gregory Underhill, itk AVtobvS SARL, Fribourg
Stiepan Aurélien Kovac, QRCrypto SA, Fribourg

xb@quant-x-sec.com
john.underhill@protonmail.com
contact@qrcrypto.ch

1 Introduction

We introduce an independent research project on symmetric cryptography with a focus on foreseeable industrial needs and higher post-quantum security compared to currently used symmetric algorithms. It was initiated by the independent IT-Security experts Kovac and Underhill. The result is the new symmetric cryptographic algorithm EAES*, which is intended to be a stronger brother of the widely used* AES *algorithm (Advanced Encryption Standard), the standardized version of the* RIJNDAEL *algorithm. In this analysis we show, that* EAES *offers an enhanced complexity by a factor* $\geq 2^{126}$ *regarding the quantum cryptanalysis Grover's search algorithm compared to* AES*. Furthermore we outline the basic facts and open questions regarding quantum algebraic attacks on* EAES.

It is known since 1995 that the security of currently used asymmetric cryptographic algorithms relying on the hardness of integer factorization and finding discrete logarithms (DLOG systems) will expire with the availability of potent enough quantum computers. By then, all private keys will be computable within reasonable time from the corresponding public keys. With the knowledge of those private keys, all encrypted data, which was collected and assigned to the relevant key exchanges, will no longer remain secret.

With the advent of 49 qubit processors quantum supremacy, the ability of quantum computing devices to solve problems that classical computers practically cannot solve, lies within reach. IBM's 14th quantum computer is its most powerful so far, a model with 53 of the qubits that form the fundamental data-processing element at the heart of the system [1]. Google participates in the race with their 72-qubit quantum processor Bristlecone [2]. Furthermore, successful discoveries through research for topological quantum computation [3] might create a verbatim "quantum leap" in quantum computation evolvement.

For this reason, the NIST launched a standardization process on asymmetric post-quantum cryptography, by definition, cryptography which is resisting quantum computer attacks. The evaluated algorithms rely on hardness of mathematical problems other than integer factorization and DLOG. They run effectively on currently used binary devices and also offer security against current and evolving threats performed on potent binary devices.

2 The importance of symmetric post-quantum secure algorithms

Hybrid encryption systems are used in all major crypto protocols: TLS, SSH and PGP. They combine the advantages of both cryptography classes. Symmetric protocols allow securely sharing a key via digital connections, and symmetric protocols are about $10^5\times$ faster than asymmetric ones. The secret session key SK, which is limited in time, is shared via asymmetric cryptography, and the message itself is symmetrically encrypted with the securely shared session key.

No asymmetric encryption within hybrid encryption systems can outbalance weaknesses of the symmetric part. ISO currently makes it possible to standardize new symmetric cryptography algorithms and to amend existing ones. Therefore, we strive to establish EAES as an ISO-Standard. Its predecessor AES has been standardized by both organizations.

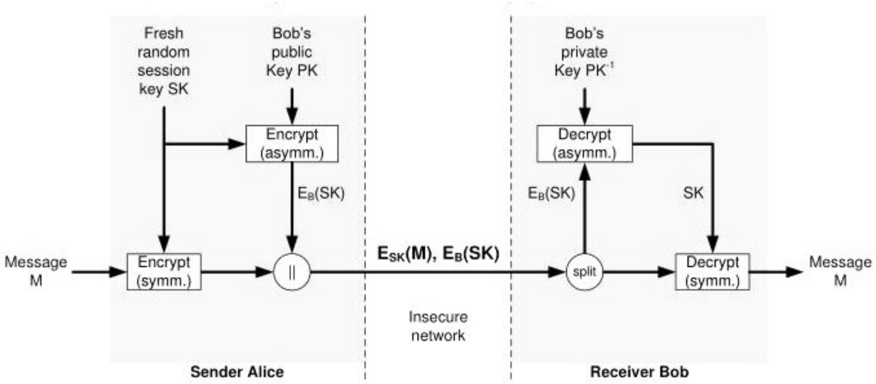

Figure 1: *Hybrid encryption.*

The algorithm EAES is designed by Kovac and Underhill. They published the e-print *Towards post-quantum symmetric cryptography* [4] on the motivation for their research as well as the design and implementation of EAES. Underhill had previously started the implementation in his CEX-NET project, within which he created the cryptographic $C++$ library CEX [5]. The library is open source and published under the GPL-license. Bogomolec is responsible for the mathematical analysis of the algorithm. All of us are independent IT-Security experts.

EAES mitigates threats by formerly published attacks on AES from binary devices [6, 7, 8] and additionally offers enhanced security against attacks performed by moderate quantum computers [9, 10].

3 Published attacks on AES

There are various types of attacks on cryptographic algorithms, amongst which side-channel attacks, implementation attacks, brute-force attacks and cryptanalysis attacks on AES are of importance in our context. In some cases, those attack types can be combined to achieve a more efficient decryption of a ciphertext.

Side-channel attacks use unintended side effects of cryptographic operations to glean information about the plaintext and/or secret key being processed. **Implementation attacks** use weaknesses in implementation of an encryption scheme, e.g. weak key generation. **Brute force attacks** attempt every possible combination for a key. **Cryptanalysis attacks** rely on alternative algorithms for finding a secret key of an encrypted text. The latter can be applied to stationary data. Therefore it is interesting for data collectors who are patient enough to wait for the availability of potent enough machines to perform such an attack.

3.1 Mathematical structure of AES

All RIJNDAEL functions are linear. Only the basic encryption function *SubBytes* (see section Basic RIJNDAEL encryption functions) is often referred as the non-linear part of AES, but in fact it is linear as well. Well chosen linear layers with very strong diffusion properties protect against conventional attacks using statistical properties of a cryptosystem. In the case of the quantum algebraic attack [10], this effect is limited.

3.2 Attacks performed on binary devices

Various published attacks on AES [6, 7, 8] take advantage of the invertibility of the original RIJNDAEL key schedule besides other properties such as the relatively low number of rounds, implementation weaknesses and the same block size for standardized all key sizes.

3.3 Grover's search

AES was generally still considered post-quantum secure for key sizes larger than 192. Even Grover's search algorithm [11] is not regarded a threat for AES-256 in the near future. It can be used to extract the key from a small number of AES plaintext-ciphertext pairs (5 for AES-256). Grover's search is a alternative algorithm for an exhaustive key search (brute force attack).

3.4 Quantum algebraic attack

A harder impact on the security of AES-256 is posed by the quantum algebraic attack on cryptosystems which can be reduced to Boolean equation solving [10]. this attack reduces security level of AES-256 from 256 to 78.53. The quantum algebraic attack is a classical cryptanalysis attack.

3.5 Consequences

Both Grover's search algorithm and the quantum algebraic attack can be applied to collected data without the corresponding key exchanges as soon as potent enough quantum computers are available. Therefore we propose eAES as a symmetric alternative for AES, with higher security against all previously mentioned attacks [6, 7, 8, 9, 10].

4 Rijndael, the base of eAES

Here we only give a high level overview of Rijndael in order to classify our modifications. For a detailed description of its standardized version AES, we refer to the Federal Information Processing Standards Publication 197 [12].

4.1 Computation grounds

Rijndael is an iterative rounds-based block cipher. It relies on a substitution-permutation network. In AES, the network is operating on a 4×4 column-major order array of the 128 bits (block size), called the „state“. Each input data block is intially transformed into a „state“. The term „state“ refers to the digital representation as well as to the fact of the continuous transformation during the encryption. So each array entry consists of 8 bits. Columns of the matrix are also called „words“. All operations are performed in the Galois field $GF(2^8) \cong \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$.

4.2 Rijndael algorithm architecture

Rijndael is a block cipher. That means that the basic encryption functions are iterated a certain number of rounds with dedicated round keys over the state. With $r =$ number of rounds, the encryption process is:

(1) **Derive round keys**
Call *ExpandKey* (key) to derive $r + 1$ round keys from the secret key. The $+1$ stands for the fact that *AddRoundKey* is called additionally before each block encryption.

(2) **Encryption**
Perform on each block of data represented by the „state“:

```
a) Initial round:
       Add the plaintext to the state
       AddRoundKey (state, 1st round key)
b) For ($i = 2$; $i \leq r - 1$, $i$++):
       SubBytes (state)
       ShiftRows (state)
       MixColumns (state)
       AddRoundKey (state, $i$-th round key)
   (4 basic Rijndael encryption functions)
c) The final round:
       SubBytes (state)
       ShiftRows (state)
       AddRoundKey (state, last round key)
```

The decryption process is composed by the inversed chain of the encryption functions. A symmetric encryption algorithm needs to be composed by invertible functions, but this property wouldn't be necessary for the key expansion. In fact, the invertibility of *ExpandKey* opened the door for various side-channel attacks.

4.3 Basic RIJNDAEL encryption functions

The 4 basic functions of the RIJNDAEL encryption are:

1) *AddRoundKey* - addition in $(\mathbb{F}_2)^{128}$:
Bitwise addition of the state and the correspondent round key.
2) *SubBytes* - non-linear substitution:
Each byte is replaced by another according to the specified substitution table (S-Box). A more resource friendly option is to treat a state byte as an element $\alpha \in \mathbb{F}_2[x]/(x^8+x^4+x^3+x+1)$, where the multiplicative inverse of α needs to be found.
3) *ShiftRows* - transposition for diffusion:
The second, third and fourth row of the state are shifted to the left, by 1, 2 and 3 steps.
4) *MixColumns* - mixing for diffusion:
Multiplication of each column of the state with the following matrix M:

$$M = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

As a chain of invertible algebraic functions, RIJNDAEL and AES can be represented as a polynomial system [13]. This polynomial system can re reduced to a Boolean equation system. We will look at this property in the context of the quantum algebraic attack [10].

5 Modifications for EAES

Our modifications affect the key schedule *ExpandKey* and the number of rounds per version. Furthermore we implemented a version for a 512-bit key, which is no outlier amongst post-quantum secure algorithm key sizes.

5.1 RIJNDAEL inheritance

The original RIJNDAEL algorithm includes a block size option of 256 bits, which was not admitted for the AES standard. We decided to keep the 128 bits for EAES inorder to ensure compatibility with existing hardware implementations [1]. Several mentioned attacks take advantage of the fact that AES-256 runs with the same block size as AES-128. Those vulnerabilities are at least mitigated in our algorithm by the higher number of rounds. We also kept the 4 basic RIJNDAEL encryption functions and the RIJNDAEL algorithm architecture.

5.2 More rounds and a 512-bit key version

We increased the number of rounds taking in account recommendations of renowned cryptographers and cryptanalysts such as Bruce Schneier. The 256-bit key variant EAES-256 runs 22 rounds of the original RIJNDAEL transformation function, which is 8 more rounds than AES-256, and twice the best known attack which breaks 11 rounds [7]. Based on the same considerations, we fixed the number of 30 rounds for EAES-512 .

5.3 Say goodbye to 128- and 192-bit keys

Organizations such as the NIST and the EU quantum flagship [14] recommend to use the maximal key lengths of currently used cryptographic algorithms whenever possible. But for operational staff in IT, the options very often simply come down to the question about what choices they have for establishing a confidential communication with a business or cooperation partner. EAES doesn't offer the 128- and 192-bit key options anymore.

5.4 Replacing *ExpandKey*

The invertibility of the original RIJNDAEL key schedule *ExpandKey* opened the door for various published attack schemes. Furthermore the output of the original RIJNDAEL key schedule does not offer cryptographic quality. In

[1] We have also implemented an authenticated stream cipher (RCS) using the 256-bit block transform along with a cryptographically strong key-schedule, and an increase in transformation rounds that parallels eAES.

the simple cases, round keys could be extracted within encryption or decryption processes, and the original key computed by applying the inverted key schedule function. For cryptanalysis attacks, this is just another convenient property for finding a replacement of the chain of RIJNDAEL functions.
Therefore we replace *ExpandKey* by cryptographically secure Pseudo Random Number Generators (PRNGs) with strong diffusion properties in EAES.

5.5 Pseudo Random Number Generators

PRNGs are built for deriving keys of a fixed size for further cryptographic operations by using an underlying pseudo random function. In our case those pseudo random functions are hashing algorithms. They are not even injective, but produce a well defined and collision resistant output.

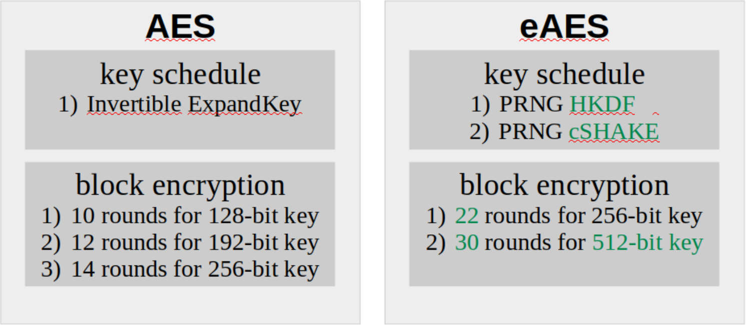
AES
key schedule
1) Invertible ExpandKey
block encryption
1) 10 rounds for 128-bit key
2) 12 rounds for 192-bit key
3) 14 rounds for 256-bit key
eAES
key schedule
1) PRNG HKDF
2) PRNG cSHAKE
block encryption
1) 22 rounds for 256-bit key
2) 30 rounds for 512-bit key

Figure 2: *High level comparison of AES and eAES.*

Furthermore, it is very cost intensive under foreseeable technical developments to perform brute force attacks on the chosen PRNGs. So they are also hardly reversible apart from the non-existant mathematical invertibility.

6 Key schedule variants

Our first choice is HKDF, a HMAC-based extract-and-expand key derivation function, with SHA-2 as the underlying hash function. As a second option, we implemented the KECCAK-derived and customizable CSHAKE-function. KECCAK, the SHA-3 competition finalist, was chosen by the NIST for its algorithmic unrelatedness from SHA-2, while offering flexibility and a comparable speed in computation.

6.1 Common features

Both HKDF and CSHAKE produce arrays of bytes which are converted to big-endian ordered 32-bit integers for the round subkeys. Each round key has the same size as the cipher's block size, namely 128 bits. So the output of our key derivation functions needs to be of the size $(r + 1) \cdot 128$ bit, where r represents the number of rounds. We renounce on the usage of a salt due to the fact that within our algorithm, the input is cryptographically strong already.

6.2 HKDF

We consider the HMAC-based HKDF as a sensible intermediary solution for the derivation of the round keys, because it is already widely available. We use HKDF(SHA-256) for EAES-256 and HKDF(SHA-512) for EAES-512 to align with expected security strengths.

SHA-256 and SHA-512 are members of the SHA-2 function family. They use the Merkle-Damgard construction, a method of building collision-resistant cryptographic hash functions from collision-resistant one-way compression functions. The compression function for SHA-2 uses the Davies–Meyer structure from a (classified) specialized block cipher.

HKDF generates cryptographically strong output of any desired length by repeatedly generating hash-blocks, concatenating them, and finally truncating the result to the desired length. Each call to HMAC involves two calls to the SHA-2 hash function to generate a pseudo-random 256-bit or 512-bit output block.

So if the number of rounds is r and the SHA-2 output length is n, SHA-2 has to be called $\lceil\frac{(r+1)\cdot 128}{n}\rceil$ times. This results in 12 times with SHA-256 for EAES-256 and 8 times with SHA-512 for EAES-512.

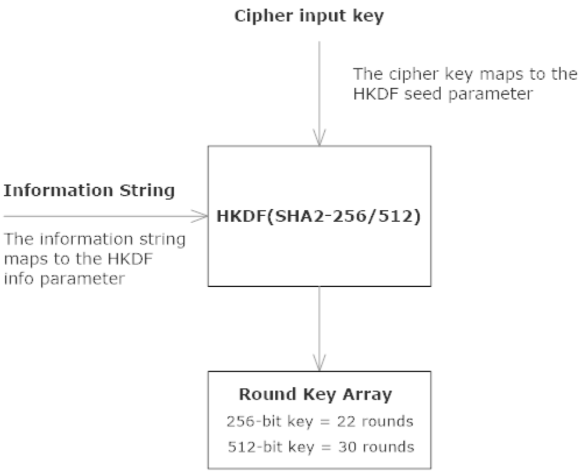

Figure 3: *Rijndael HKDF eXtension.*

6.3 CSHAKE

We implemented the SHA-3-derived SHAKE-function as a second option for the key schedule. The SHA-3 competition finalist KECCAK is a so called *sponge function*. Those are functions with finite internal state, taking an input bit stream of any length and producing an output bit stream of any desired length. CSHAKE is the customizable version of SHAKE. It is originally designed for 128- and 256-bit security strength [15]. Underhill has additionally created a 512-bit security implementation, which mirrors the internal block-size, squeeze and permutation settings of SHA3-512. CSHAKE-256 is used for EAES-256, and CSHAKE-512 is used for EAES-512.

The initial state of SHAKE is a 5×5 array of 64-bit unsigned integers, 1600 zero bits it total. The first call is to initialize the custom state. Then the original EAES-key is absorbed into the previously intialized state, and with each call to the inner permutation function of KECCAK, rates of $(1600 - 2 \cdot n,\ n \in \{256, 512\})$ bits are returned. So CSHAKE-256 returns 136, and CSHAKE-512 returns 72 pseudo-random bytes per call.

So if the number of EAES-rounds is r and the number of returned bytes per rate are n, we have $\lceil \frac{(r+1) \cdot 128}{n \cdot 8} \rceil$ calls to the inner permutation functions of CSHAKE.For EAES-256 we have 3 and for EAES-512 we have 7 calls to the inner permutation function, $+1$ call for the initial state.

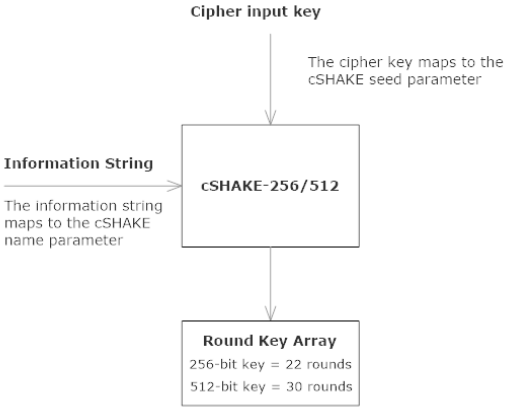

Figure 4: *Rijndael SHAKE eXtension.*

Note that here we are talking about calls to the inner permutation function of KECCAK, while we are talking about calls to SHA-2 itself in the previous section. Fewer calls of the permutation function lead to higher efficiency of CSHAKE compared to the one of HKDF. On the other hand, KECCAK is prone to quantum algebraic attacks [10].

7 Impact of Grover's Search Algorithm

Grover's algorithm offers a $\sqrt{k}$ speed-up [11] over a classical exhaustive search (brute force attack) on the set of keys of size k. It seems to be one of the most relevant quantum cryptanalytic impact for the study of block ciphers. The authors of [9] present quantum circuits to implement the key search for AES and analyze the quantum resources required to carry out such an attack for key sizes of 128, 192 and 256 bits.

The number of required logical qubits is relatively low, $6,681$ for AES-256. On the other hand, the large circuit depth of enrolling the entire Grover iteration poses a challenge to an implementation on an actual physical quantum computer, even if the gates (basic quantum circuit operating on qubits) are not error corrected. The key schedule *ExpandKey* causes much of the circuit cost within each Grover iteration. Replacing it by our chosen PRNGs will even be considerably more cost intensive.

Here we only summarize the basic prerequisites of the involved procedures described in [9] for AES-256 and compare them to EAES for $k \in \{256, 512\}$.

7.1 Algorithm architecture

A quantum circuit implementing a Boolean function f is the input of the Grover's search algorithm:

$$f : \{0,1\}^k \to \{0,1\}$$

The basic algorithm finds an element x_0 such that $f(x_0) = 1$. This is realized by repeatedly applying the operation G to measure an element x_0 such that $f(x_0) = 1$ with constant probability.

$$G = U_f \left((H^{\otimes k}(2|0\rangle\langle 0| - 1_{2^k})H^{\otimes k}) \otimes 1_2 \right)$$

H denotes the 2×2 Hadamard transform and U_f involves the computation of the cipher functions. To ensure uniqueness of the solution (the found key), a small number of plaintext-ciphertext pairs are needed. This number is 5 for a 256-bit key and 9 for 512-bit key.

7.2 Quantum resources and graph theory

Quantum resources are represented by logical qubits, gates and circuit depth. In the model of a Boolean circuit as a directed acyclic graph, the circuit depth is the maximal length of a path from an input gate to the output gate. The sum of required gates represent the complexity of an algorithm.

Reversible circuits are needed for the application of Grover on AES to provide invertibilty of the operations. Therefore the proposed solution in [9] relies on a set of fault-tolerant reversible logical gates. It consists of called Toffoli gates, controlled NOT gates and NOT gates. A Toffoli gate is a universal reversible logic gate, i.e. any reversible circuit can be constructed from Toffoli gates.

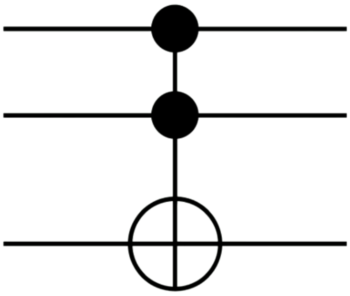

Figure 5: *Toffoli gate (controlled-controlled-not gate).*

For our key schedule replacements, we will base our comparisons on the results of „Estimating the cost of eneric quantum pre-image attacks on SHA-2 and SHA-3s" [16]. The authors present an implementation including Grover's algorithm on reversible circuits on on a surface code based fault-tolerant quantum computer.

7.3 Quantum resources for the search

H represents the Hadamard operation on a single qubit. A single solution to the equality function f can be found by applying $H^{\otimes k+1}$ to the initial state, where k is the size of the key. The next step is applying

$$G^{\lceil \frac{\pi}{4}\sqrt{2^k}\rceil},$$

followed by a measurement of the entire quantum register which will yield a solution x_0 with high probability.

The exact decomposition of the search is outlined in [9], section 2. For our comparison we only look at the cost for the operation $(2|0\rangle\langle 0| - 1_{2^k}))$, which is determined by the reduction to the implementation of a k-fold NOT gate, where $k \in \{256, 512\}$ is the key size. In terms of Toffoli gates we have the formula $8k - 24$ [17] and if we count only T-gates, we have the formula $32k - 84$ as an upper bound for a k-fold controlled NOT gate [18]. This results in:

Table 1: Cost for the operation $(2|0\rangle\langle 0| - 1_{2^k}))$**:**

key size	Toffoli gates	T-gates
256	2,024	8,108
512	4,072	16,300

Grover's search will have to be performed on the small number n of plaintext-ciphertext pairs. The equality function f can be implemented by a multiple controlled NOT gate that has $128 \cdot n$ controls. With above formulas, the needed number of Toffoli counts and T-counts to compute the equality function f come down to:

Table 2: ost for the equality function f:

key size	Toffoli counts	T-counts
256	5,096	20,396
512	9,192	36,780

The search resources only depend on the key sizes and not on the ciphers and their options. In this regard we don't have to include further differing considerations.

7.4 Quantum resources for the ciphers

The number of rounds depends on the specific key length k, while the four basic RIJNDAEL functions are independent of the key length k for both AES and EAES. The realization of all RIJNDAEL functions is analyzed in dedicated sections in [9]. Here we only present the results and comparisons per function.

Table 3: Cost for SHA-2 and -3 functions (256 bit):

function	gates	depth	qubits
SHA-256	$1.42 \cdot 2^{146}$	$1.69 \cdot 2^{144}$	$2^{12.6}$
SHA3-256	$1.52 \cdot 2^{147}$	$1.33 \cdot 2^{137}$	2^{20}

The values for SHA-256 and SHA3-256 are results from the resource estimates done in [16], converted to a representation in powers of 2 instead of 10. For both functions, the total cost comes down to approximately 2^{166} basic operations. The estimation of the resources for the according 512-bit versions will be done in a further step of our research.

For our EAES-options we have to multiply the gate values by the number of calls to the hash and inner permutation functions.

Table 4: Cost for AES and EAES key expansion (256 bit):

function	gates	depth	qubits
ExpandKey	54,331	23,896	512
HKDF	$1.07 \cdot 2^{150}$	$1.26 \cdot 2^{148}$	$2^{12.6} + 3072$
CSHAKE	$1.52 \cdot 2^{149}$	$1.33 \cdot 2^{139}$	$2^{20} + 1024$

The numbers of gates and depths take the number of calls to the according hash and permutation functions into account. Additional qubits are needed to store the output of all calls to the hash functions.

As mention before, we cannot refer to values for 512-bit keys at this time. But we can see that the 256-bit versions already offer considerable advantages over the original RIJNDAEL key schedule *ExpandKey*.

Table 5: Cost for holding the state and initial round:

function	gates	depth	qubits
state	0	0	128
initial round v_1	64 NOT	0	128
initial round v_2	128 CNOT	1	256

Initial round v_1 represents the realization with flipping bits. These values are the same for all ciphers and options.

Table 6: Cost for basic functions

function	gates	depth	qubits
SubBytes	22,326	1	9
ShiftRows	0	0	0
MixColumns	277 CNOT	39	0
AddRoundKey	128 CNOT	1	0

The current round key is available on 128 dedicated wires. *SubBytes* is realized by finding the inverse of the byte in $GF(2^8)$, seen as permutation in $\mathbb{F}_{256}$. No extra gates are necessary to implement *ShiftRows*, as it corresponds to a permutation of the qubits. The position of the subsequent gates is simply adjusted to the correct input wire.

Cost for RIJNDAEL rounds

To compute all rounds of RIJNDAEL, 536 qubits are needed for 10 rounds in the 128 bit key version and 664 qubits are required for any round number $r \geq 12$. In this regard, the higher number of rounds in EAES does not add complexity to the algorithm. But the number of rounds do add complexity to the number of gates and to the depth.

Cipher output for 256 bit keys

The number of gates and depth for EAES is derived as follows:

Values from the rounds row in table 4, AES-256 in [9] :

g_{14} = sum of number of gates in [9]
d_{14} = sum of depths in [9]

Values from the key expansion cost table in this article:

g_{hkdf} = number of gates for HKDF-256
d_{hkdf} = depths for HKDF-256
q_{hkdf} = number of qubits for HKDF-256
g_{shake} = number of gates for CSHAKE-256
d_{shake} = depths for CSHAKE-256
q_{shake} = number of qubits for CSHAKE-256

Then we compute the values for EAES(HKDF) from the formulas:
$gates = g_{hkdf} + \frac{22}{14} \cdot g_{14}$ $depth = d_{hkdf} + \frac{22}{14} \cdot d_{14}$

The values for EAES(CSHAKE) from the formulas:
$gates = g_{shake} + \frac{22}{14} \cdot g_{14}, \; depth = d_{shake} + \frac{22}{14} \cdot d_{14}$

And the number of qubits for is the simple addition $664 + q_{sha}$, $664 + q_{sha3}$ respectively.

Table 7: Cost for each cipher output (256 bit)

cipher	gates	depth	qubits
AES-256	$1.65 \cdot 2^{21}$	$1.46 \cdot 2^{17}$	1,336
EAES(HKDF)	$1.07 \cdot 2^{150}$	$1.26 \cdot 2^{148}$	$1.21 \cdot 2^{13}$
EAES(cSHAKE)	$1.52 \cdot 2^{149}$	$1.33 \cdot 2^{139}$	$1.002 \cdot 2^{20}$

Note that the gates and depth values for EAES equal the values in the key expansion cost table. This comes from the fact that the contribution of the gates and and depth from the rounds is in much lower power of 2 than the cost for the key expansions HKDF and cSHAKE. So we see that this replacements are the essential change and contribution to a higher algorithm complexity.

Grover algorithm

For the overall T-count for Grover on Rijndael we have an estimate of $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor \cdot (6t_k + f_k)$, where k is the key size, t_k is the number of T-gates and f_k is the cost for the equality function. For the overall circuit depth we obtain with $r =$ number of rounds: $2 \cdot r \cdot$ total T-depth. Here we are ignoring some of the gates which do not contribute significantly to the bottom line. Both formulas are from [9], section 3.4.. So by a moderate estimation we get the following complexities for Grover's search on the compared ciphers:

Table 8: Grover's search on ciphers

cipher	gates	depth	qubits
AES-256	$3.24 \cdot 2^{151}$	$1.71 \cdot 2^{145}$	6,681
EAES(HKDF)	$> 2^{278}$	$> 2^{274}$	$> 2^{14}$
EAES(cSHAKE)	$> 2^{277}$	$> 2^{257}$	$> 2^{21}$

Both HKDF and cSHAKE have a considerable impact on the increased cost of Grover's search. The higher number of rounds for EAES-256 and the 512-bit key version additionally contibute to the complexity of the rounds by factors of at least $\frac{r}{14}$. The authors of [9] recommended in 2015 to move away from AES-128 when expecting the availability of at least a moderate size quantum computer. Our implementation excludes that option and offers an increased complexity of at least 2^{126} compared to AES-256.

8 Impact of Quantum Algebraic Attacks

The authors of [10] present an algorithm which leads to new considerations of the security of systems which can be reduced to solving Boolean equations. A solution a for the equation $\mathcal{F} \cdot a = 0$ with a set of polynomials $\mathcal{F} \subset \mathbb{C}[X]$ is called *Boolean* if each coordinate of a is either 0 or 1.

The resulting quantum algebraic attack algorithm includes quantum-monomial solving of polynomial systems over $\mathbb{C}$ by applying a Macauly linear system. Like this they constructed a Boolean equation solving algorithm, with the following properties:

1) It decides if there exists a Boolean solution.
2) It returns a Boolean solution with a given probability if there are such solutions to the system.
3) It returns $\emptyset$ if no Boolean solution exists.

The runtime complexity of the resulting quantum algebraic attack is considerably lower than the one of Grover's Search, but it depends on two factors: a constant c and a condition number κ. The complexity of $2^{78.53} c \kappa^2$ for AES-256 is not much higher than the complexity of $2^{73.30} c \kappa^2$ for AES-128 due to the same block size of 128 bit. Therefore we can assume that the complexity won't be much higher for 512-bit key sizes, and it will also not considerably increased by the higher number of rounds in EAES.

The conclusion of [10] is, that systems which can be solved by Boolean equation solving, are only secure under quantum algebraic attack, if the condition number κ is large. The construction of such systems is a topic for further research. Besides AES and Keccak, stream ciphers such as Trivium an the multivariate public key cryptosystem MPKC are affected by the attack.

9 Conclusion

Considering the lower exponent of the highest power of 2 in the total algorithm complexity, eAES offers a higher complexity of a factor $\geq 2^{277-151} = 2^{126}$ regarding Grover's search algorithm compared to AES, even if only the 256-bit version is used. Regarding the quantum algebraic attacks, we can say that there is no attack outlined yet for the version with HKDF. Regarding the version with cSHAKE, the quantum algebraic attack has to be adapted to two phases: Keccak and then on the rounds functions. The complexity of such a solution remains to be investigated.

We consider eAES a sensible candidate for a first generation post-quantum secure symmetric encryption. It runs effectively on currently used devices and is compatible with existing hardware implementations. We hope to be able to contribute to a smooth transition into the new post-quantum cryptographical era.

References

[1] S. Shankland, MSN News, *https://www.msn.com/en-us/news/technology/ibms-new-53-qubit-quantum-computer-is-its-biggest-yet/ar-AAHtPaW*, September 2019.

[2] J. Kelly, Google AI Blog, https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html, March 2018.

[3] S. Ran, C. Eckberg, Q.-P. Ding, Y. Furukawa, T. Metz, S.R. Saha, I-L. Liu, M. Zic, H. Kim, J. Paglione and N.P. Butch, NIST events on paper of above authors, *https://www.nist.gov/news-events/news/2019/08/newfound-superconductor-material-could-be-silicon-quantum-computers*.

[4] S. Kovac and J. Underhill, Towards post-quantum symmetric cryptography, *https://eprint.iacr.org/2019/553*.

[5] J. Underhill, The CEX Cryptographic library in C++, *https://github.com/Steppenwolfe65/CEX*.

[6] A. Biryukov and D. Khovratovich, Related-key Cryptanalysis of the Full AES-192 and AES-256, *https://eprint.iacr.org/2009/317.pdf*.

[7] A. Biryukov, O. Dunkelman, N. Keller, D. Khovratovich and A. Shamir Key Recovery Attacks of Practical Complexityon AES-256 Variants With Up To 10 Rounds *http://www.wisdom.weizmann.ac.il/ orrd/crypt/PracticalAES256.pdf*.

[8] S. Lucks, Attacking Seven Rounds of Rijndael under 192-bit an 256-bit Keys, *https://madoc.bib.uni-mannheim.de/10615/*, 2000.

[9] M. Grassl, B. Langenberg, M. Roetteler, R. Steinwandt, Applying Grover's algorithm to AES: quantum resource estimates, *https://arxiv.org/abs/1712.06239*, 2018.

[10] Y. -A. Chen, X. -S. Gao, Quantum Algorithms for Boolean Equation Solving and Quantum Algebraic Attack on Cryptosystems, *https://arxiv.org/abs/1712.06239*, 2018.

[11] L. K. Grover, A fast quantum mechanical algorithm for database search, *Gary L. Miller, editor, Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC 1996), pages 212–219* ACM, 1996.

[12] Federal Information Processing Standards Publication 197, NIST, Announcing the ADVANCED ENCRYPTION STANDARD (AES), *https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf*, 2000.

[13] C. Cid, Information Security Group, University of London, Algebraic Analysis of AES, *https://www.cosic.esat.kuleuven.be/ecrypt/AESday/slides/AES-Day-CarlosCid.pdf*, October 2012.

[14] J. Baloo, KPN, Everything is Quantum – The EU Quantum Flagship, *https://2017.pqcrypto.org/conference/slides/baloo.pdf*, 2017.

[15] J. Kelsey, S. -J. Chang, R. Perlner, SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash, *https://nvlpubs.nist.gov/nistpubs/SpecialPublications /NIST.SP.800-185.pdf*, December 2016.

[16] M. Amy, O. Di Matteo, V. Gheorghiu, M. Mosca, A. Parent, J. Schanck, Estimating the cost of eneric quantum pre-image attacks on SHA-2 and SHA-3s, *https://eprint.iacr.org/2016/992.pdf* QCrypt, 2016.

[17] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. W. Shor, T. Sleator, J. Smolin, H. Weinfurter, Elementary gates for quantum computation, *Physical Review A, 52(5):3457–3467*, 1995.

[18] N. Wiebe and M. Roetteler, Quantum arithmetic and numerical analysis using Repeat-Until-Success circuits, *https://arxiv.org/abs/1406.2040*, 2014.