# Towards Post-Quantum Secure Symmetric Cryptography; a Cryptographic Analysis

Xenia Bogomolec - xb@quant-x-sec.com

John Gregory Underhill - john.underhill@protonmail.com

Stiepan Aurélien Kovac - stie@itk.swiss

# Introduction

It has been 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 [1]. By then, all private keys will be computable within a 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 [6]. Google participates in the race with their 72-qubit quantum processor Bristlecone [7], and have publicly stated that their goal is to double the number of qubits each year. State sponsored intelligence agencies are also sprinting towards quantum supremacy, with vast resources, and capabilities that remain publicly unknown. Furthermore, successful discoveries through research for topological quantum computation [8] might create a verbatim "quantum leap" in quantum computation evolvement.

For this reason, the National Institute of Standards and Technology (NIST) launched a standardization process on asymmetric post-quantum cryptography, by definition, cryptography which is resistant to quantum-based computer attacks. The evaluated algorithms rely on the 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.

**Post-quantum security for symmetric cryptography**

Currently used symmetric cryptographic algorithms were generally still considered secure under the assumption of impracticality of formerly published attacks, e.g. the *related-key attack* [2]. On the other hand, published articles about the security of symmetric block ciphers with regards to quantum computing, such as *Applying Grover’s algorithm to AES: quantum resource estimates* [4] and a *Quantum algorithms for boolean equation solving and quantum algebraic attack on cryptosystems* [5], demand new considerations regarding post-quantum security of symmetric ciphers such as AES. *Grover’s search algorithm* on AES-256 only requires 6681 logical qubits according to the analysis in [4].

As a consequence of the aforementioned facts and events and the rapidly evolving threat landscape, we see the necessity of moving towards post-quantum security in symmetric cryptography as well. No asymmetric encryption within hybrid encryption systems can outbalance weaknesses of the symmetric part of that system.

Here we focus on a modification of the Rijndael cipher, which addresses attacks on the reversibility of the original key schedule function [9, 2]. Furthermore, we implemented an increased number of rounds within the block encipherments to counter scaled system attacks. The chosen round parameter options from Rijndael for AES relied on considerations about efficiency of binary devices used almost 20 years ago. Our enhancements come with higher computation resources, which are easily compensated by the high performance of currently used devices.

**Data units**

We mainly talk about bits in terms of data units. It will make interrelations more obvious. Furthermore, we can assume that 1 byte equals 8 bits in all algorithmic and digital contexts of this paper. So, the synonymous usage of 1 byte as 8 bits is innocuous.

**Invertibility vs. reversibility**

"Invertible" stands for invertibility of a function in a mathematical sense, meaning that we have a bijective function. "Reversibility" refers to computational aspects and can also include a brute force attack which can be executed successfully within reasonable time, such as finding the input data of a given hash value by trying all possible inputs. This can be done under suitable conditions with additional knowledge of the input data, such as its size, even though the hashing function is not even injective.

**Inheritance and modiﬁcations**

Rijndael relies on a substitution-permutation network. It operates on a 4 × 4 column-major order array of the 132 bits, called "the state". Columns of the matrix are also called "words" and expressed as 32-bit unsigned integers. So, each array entry consists of 8 bits. Furthermore, Rijndael uses the characteristic 2 ﬁnite ﬁeld with 256 elements, the Galois ﬁeld ).

For non-mathematicians, F2 represents the ﬁeld represented by the set {0,1}, with addition and multiplication. The so-called reducing polynomial for computations, x8 + x4 + x3 + x + 1, equals 0 in *GF*(28).

The eAES cipher design inherits the block size of AES, the transformation and rounds functions and the high-level algorithm architecture of Rijndael.

Rijndael is an iterative rounds-based block cipher. Our modiﬁcations primarily affect the key-schedule, which generates the set of round keys used in the rounds function, as well as increasing the number of rounds of transformation applied to the state. We have implemented an extended AES implementation using both a 256-bit key, and a 512-bit input-key, which is no outlier amongst post-quantum secure algorithm key sizes.

**AES block size**

The block size; the size of the internal state in eAES remains the same as in the AES family of cryptosystems, 16 bytes, or 128 bits respectively. 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-bit block in eAES for compatibility reasons with existing hardware implementations, even though several published 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 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.

**Basic Rijndael encryption functions**

We have kept the 4 basic functions of the Rijndael transformation within the encryption of each block: 1) AddRoundKey – A key-whitening step, the addition in (F2)128: XOR-ing each byte of the state with a byte of the 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 α ∈F2[x]/(1 + x8 + x4 + x3 + x + 1), where the multiplicative inverse of α (leaving 0 invariant) 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: the multiplication of each column of the state with the following matrix M:

M = |2, 3, 1, 1|

|1, 2, 3, 1|

|1, 1, 2, 3|

|3, 1, 1, 2|

All those basic functions are invertible, and all of them, excepting the SubBytes step, are linear.

**Rijndael algorithm architecture**

The eAES cipher also inherits the high-level algorithm architecture of Rijndael which is described by:

Call the key-schedule function ExpandKey, and the application of the rounds function call per block:

1. the initial round key addition AddRoundKey
2. the number of rounds of SubBytes, ShiftRows, MixColumns and AddRoundKey
3. the ﬁnal round SubBytes, ShiftRows and AddRoundKey

The rounds function first adds the plaintext to the state, then XOR-ing the first round-key with the state in the AddRoundKey step. The Rijndael rounds transformation is then applied to the state using the SubBytes, ShiftRows, MixColumns steps, with each round executed iteratively on the state using a unique round-key. The rounds function concludes with a SubBytes SBox substitution, a ShiftRows, and an additional AddRoundKey key-whitening step.

**PRNGs for key expansion, the essential modiﬁcation**

The original Rijndael key schedule expands the input cipher-key by a series of steps; applying the S-Box, left circular shifts, and bitwise XOR-operations to the words in the state representation of the original key. The majority of round-keys are derived by an XOR of two previous keys, and the output of this simple expansion function can not be considered of cryptographic quality. We forego its exact description here, and will refer to other papers in the context of comparisons regarding quantum computing resources. For now, it is mostly important to note that the Rijndael key schedule function is invertible as well. This property can open the door for various attacks on the encryption and decryption processes, within which subkeys could be extracted, and the original key computed by applying the inverted key schedule function.

Our chosen replacements are cryptographically secure PRNGs with strong diffusion properties. Our choice of PRNGs use hashing algorithms as ingredients, they are not even injective, but produce a well deﬁned output. These properties ensure security against all attacks on gained knowledge of round keys from round 2. Furthermore, they are hardly reversible under foreseeable technical developments within the next decades.

**More rounds and a 512-bit key version**

We have increased the number of rounds taking into 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 [3]. We chose to use ***2n*** the best-known break of 11 rounds as the threshold for the 256-bit implementation, and 30 rounds for the 512-bit key version, because we feel this should be adequate for the lifetime of the AES cryptosystem, and conservative enough, so that we can avoid the need to revise the standard again.

**Key schedule variants**

We consider suitable for key expansion, cryptographic extended output functions (XOF) and key derivation functions (KDF) as appropriate replacements for the invertible and differentially-weak Rijndael key schedule. Those functions are built for deriving keys of a ﬁxed size for further cryptographic operations by using a chosen underlying pseudo-random function. Our ﬁrst choice is the Keccak based SHAKE extended output function (XOF). A customized SHAKE implementation was chosen in part because of its standardization as the replacement for older SHA2 hashing functions, it is widely accepted as a strong pseudo-random generator within the cryptographic community, and that it is a more efficient key expansion function in this context. We also provide a version using the HKDF-Expand function to generate the round key array, this is an HMAC(SHA2) based key expansion function provided for backwards compatibility on systems that may not yet have adopted the new standard. These two variants are formally named Rijndael HKDF eXtension (RHX), and Rijndael Shake eXtension (RSX), the only difference between the two being the type of key expansion mechanism.

Both versions make use of an information string; this is an input secondary to the input cipher-key, that can either be a user-supplied string used to create a tweakable cipher implementation that generates unique ciphertext output by using a shared secondary key, or as default, a string derived internally from an expression of the ciphers formal string name, which is the ciphers variant name, the key size, and the expansion functions name, for example ‘RSX256-SHAKE256’. When using the cSHAKE version of the cipher, this name string is used to create a unique output by first adding the string to the state, and then calling the Keccak permutation function, initializing SHAKE’s internal state to non-zero initial values. When using the HKDF-Expand function (RHX), this name string is used as the HKDF information string, passed to HMAC along with the cipher input-key and a monotonic counter to generate blocks of pseudo-random output.

**Resolving the AES key length dilemma in the face of evolving technologies**

With these replacements, we implement resistance against known attacks on the original key schedule. Even though attacks such as the related-key attack and the related-subkey attack [2] weren’t considered practical due to possible countermeasures within an integrated key generation, they were often cited and put the advantage of AES-256 over AES-128 in question. These attacks do however, demonstrate two things conclusively; that the differentially-weak native Rijndael key-schedule is a serious vulnerability in the cipher design, and that the number of transformation rounds is currently set too low.

Organizations such as the NIST and the EU quantum ﬂagship [10] recommend to increase the key lengths of currently used cryptographic algorithms whenever possible. The resulting dilemma is sometimes resolved by focusing on other symmetric algorithms. But in many cases, it takes time to establish newer algorithms in the IT-landscape. DES for example is still being used for international ﬁnance transactions in VPNs, even though it is ofﬁcially declared insecure. AES is still widely used globally. For operational staff in IT the algorithm options very often simply come down to the question about what choices they have for establishing a conﬁdential communication with a business or business partner.

**Common features of the chosen variants**

Both HKDF and cSHAKE produce arrays of bytes which are converted to big-endian ordered 32-bit integers for the round subkeys. Each full round-key has the same size as the ciphers block size, namely 128 bits. So, the output of our key derivation functions needs to be of the size (n + 1) · 128-bit, where *n* represents the number of rounds. The additional round-key is used in the key-whitening AddRoundKey stage.

**HKDF, a widely available PRNG**

We consider the HMAC-based HKDF as a sensible intermediary solution for the derivation of the round keys, because it is already widely available and runs effectively on currently used devices. We use HKDF(SHA2-256) for eAES-256 and HKDF(SHA2-512) for eAES-512 to align with expected security strengths.

Both SHA-256 and SHA-512 are members of the SHA-2 function family. They use the Merkle-Damgard construction around an internal permutation which is an extended Feistel network. Merkle-Damgard is a method of building collision-resistant cryptographic hash functions from collision-resistant one-way compression functions.

HKDF generates cryptographically strong output key material 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 SHA2 hash function to generate a pseudo-random 256-bit or 512-bit block. So SHA-256 has to be called d(22+1)·128 256 e = 12 times for eAES-256, and SHA2-512 is called (30+1)·128 512 e = 8 times for eAES-512 .

**cSHAKE, ﬂexible hashing architecture**

We implemented the Keccak derived and customizable SHAKE function as the primary option for the key expansion generator. cSHAKE is designed for 128- and 256-bit security strength [11]. We have also created a cSHAKE-512 implementation, which mirrors the internal block-size, squeeze and permutation settings of SHA3-512. The 256-bit version is used when a 256-bit cipher input key is chosen, the 512-bit version of the generator is selected when using the 512-bit key implementation of eAES.

Keccak, the SHA-3 competition ﬁnalist, was chosen by the NIST for its algorithmic unrelatedness from SHA-256, while offering ﬂexibility and a comparable speed in computation. Additionally, the different algorithm architectures of SHA-2 and SHA-3 make it less likely that potential future cryptanalytic breakthroughs might compromise the security of both hash function families.

The initial state of SHAKE is an array of 25 64-bit unsigned integers, 1600 bits it total. The information string used by RSX is absorbed into the state then permuted, setting the initial SHAKE state to non-zero values in a customized SHAKE configuration (cSHAKE). The cipher-key input is absorbed into the state, and with each call to the inner permutation function of Keccak, rates of (1600 − 2 · n, n ∈ {256,512}) bits are returned. cSHAKE256 returns 136 pseudo-random bytes per call, and cSHAKE-512 returns 72 bytes. We choose cSHAKE-256 for eAES-256 and cSHAKE-512 to correspond to the expected security strength of eAES-512. For eAES-256 we have d(22+1)·128 136·8 e = 3 calls to the permutation function, +1 call for the initial state. For eAES-512 we have d(30+1)·128 72·8 e = 7 calls to the permutation function, +1 call to initialize the custom state.

Note that here we are talking about calls to the inner permutation function of Keccak, while we are talking about calls to SHA2 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 [5]. The complexity of ﬁnding a solution to the systems of Keccak-256 and Keccak-512 comes down to 278.25cκ, where c is a constant and κ the condition number. We will look a little closer at the impact of [5] on our complete algorithm in section 5.

**Impact of Grover’s Search Algorithm**

The square root speed-up offered by Grover’s algorithm [12] over a classical exhaustive key search seems to be one of the most relevant quantum cryptanalytic impact for the study of block ciphers. The authors of [4] present quantum circuits to implement an exhaustive 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. For AES-256 their identiﬁed approximate quantum resources are summarized in the list below. The exact computations are presented in their paper.

**Required quantum resources**

Quantum resources are represented by logical qubits (circuits), gates (elementary quantum operations) and depth (repetition of operations). The sum of required gates also represents the complexity of an algorithm.

1) 128 qubits to hold the current internal state

2) ExpandKey: > 500000 gates and a depth of > 240000 on 512 qubits for storage and ancillae.

3) AES rounds: 64 uncontrolled NOT gates for addition by ﬂipping bits or else 128 CNOT gates and 128 qubits for the initial round.

a) AddRoundKey: Current round key on 128 wires and 128 CNOT gates for parallel bitwise XOR-ing

b) SubBytes: The computation of a byte substitution requires> 200000 gates using only 9 qubits. This is performed 16 times per round, requiring 384 auxiliary qubits for all to be done simultaneously. Otherwise 24 auxiliary and 640 storage qubits with a maximum depth of 8 are required to compute 14 rounds.

c) ShiftRows: No extra gates are necessary to implement this operation as it corresponds to a permutation of the qubits. 664 qubits are needed to compute 14 rounds.

d) MixColumns: 277 gates and a total depth of 39 to operate on an entire column of the state on 32 qubits at a time.

These resources result in the costs of > 3.5 million gates, a depth of about 2000000, and 10336 qubits to achieve the output of each AES-256 system. Quantum resource estimates for Grover’s algorithm to attack AES-256 are 3.24 · 2151 gates, a depth of about 1.64·2145, and 60681 qubits. The identiﬁcation of required quantum computing resources for a brute force Grover’s search on a fault tolerant surface code-based architecture on SHA-256 and SHA3-256 has been done in [13]. It costs 1.27·1044 T-gates, a depth of about 3.76 · 1043, and 3615 qubits. For SHA3-256 the costs are 2.71·1044 T-gates, a depth of about 2.31 · 1041, and 3615 qubits. For both functions, the total cost comes down to approximately 2162 basic operations.

Both SHA2-256 and SHA3-256 are considerably more cost intensive in this context than the original Rijndael key schedule. The higher number of rounds for eAES-256 and the 512-bit key version additionally mitigate the threat of Grover’s search on eAES. The authors of [4] 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 higher security than AES-256.

**Impact of Quantum Algebraic Attack**

The authors of [5] 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 F · a = 0 with a set of polynomials F ⊂ C[X] is called boolean if each coordinate of a is either 0 or 1. Philosophically spoken we can say that ﬁnding a binary solution of a system with a by complex polynomials is the mathematical expression of the quantum principals applied to binary computations.

The resulting quantum algebraic attack algorithm includes quantum-monomial solving of polynomial systems over Cby applying a Macauly linear system. Like this they constructed a boolean equation solving algorithm which decides if there is boolean solution, returns a boolean solution with a given probability, if there are such solutions to the system, and returns ∅ 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 278.53 cκ2 forAES-256 is not much higher than the complexity of 273.30 cκ2 for AES-128. We can assume that the complexity won’t be much higher for 512-bit key sizes, and it will also not considerably be increased by the higher number of rounds in eAES.

The conclusion of [5] 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 the multivariate public key cryptosystem MPKC are affected by the attack.

**Conclusion**

The exact impact of our changes regarding a quantum algebraic attack remains to be analyzed. But eAES is intended to be a transitional solution, a step towards post-quantum security. We consider eAES a sensible candidate for this purpose. It runs efficiently on currently used devices, is compatible with existing hardware implementations and offers higher security than standard AES.

eAES will be proposed for an ISO-Standard. We hope to be able to contribute to a smooth transition into a new cryptographical era.

References

[1] P. W. Shor, Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, https://arxiv.org/abs/quantph/9508027, 1995.

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

[3] 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.

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

[5] Yu-Ao Chen, Xiao-Shan Gao, Quantum Algorithms for Boolean Equation Solving and Quantum Algebraic Attack on Cryptosystems, https://arxiv.org/abs/1712.06239, 2018.

[6] S. Shankland, MSN News, https://www.msn.com/en-us/news/technology/ibmsnew-53-qubit-quantum-computer-is-its-biggestyet/ar-AAHtPaW, September 2019.

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

[8] 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/newsevents/news/2019/08/newfound-superconductormaterial-could-be-silicon-quantum-computers>.

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

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

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

[12] 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.

[13] 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.