Towards Post-Quantum Symmetric Cryptography

Stiepan Aurélien Kovac, Master's Degree, ICT Security, Algorithm Design
John Gregory Underhill, Cryptographic C-Programmer, Algorithm Design
Dipl. Math. Xenia Bogomolec, Mathematical Analysis

`stie@itk.swiss`
`john.underhill@protonmail.com`
`xb@quant-x-sec.com`

Introduction

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 [1]. 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 [2]. Google participates in the race with their 72-qubit quantum processor Bristlecone [3]. Furthermore, successful discoveries through research for topological quantum computation [4] 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.

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* [5]. 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* [6] and a *Quantum algorithms for boolean equation solving and quantum algebraic attack on cryptosystems* [7], 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 [6].

As a consequence of the mentioned facts and events, 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.

Here we focus on a modification of the RJINDAEL cipher, which adresses attacks on the reversibility of the original key schedule function [9, 5]. Furthermore we implemented more rounds within the block encipherments to counter scaled system attacks. The chosen round parameter options from RJINDAEL for AES relied on considerations about efficiency of binary devices used almost 20 years ago. the higher computation resources which come with our enhancements are easily compensated by performance of currently used devices.

Furthermore our cooperation is a proof of concept for the fact that holistic technical cryptographic security can only be achieved by the compbination of deep industrial and scientific experience. Those properties are usually not unfified in one person. The highest quality of all aspects can be achieved by working in a team of experts with dedicated knowledge.

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. "Reversability" 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

RJINDAEL relies on a substitution-permutation network. It operates on a 4×4 column-major order array of the 128 bits, called "the state". Columns of the matrix are also called "words". So each array entry consists of 8 bits. Furthermore RIJNDAEL uses the characteristic 2 finite field with 256 elements, the Galois field $GF(2^8) \cong \mathbb{F}_2[x]/(1 + x^8 + x^4 + x^3 + x + 1)$. For non-mathematicians, $\mathbb{F}_2$ represents the field represented by the set $\{0, 1\}$ with addition and multiplication. The so-called reducing polynomial for computations, $x^8 + x^4 + x^3 + x + 1$, equals 0 in $GF(2^8)$.

EAES inherits the block size of AES, the basic encryption functions and the high level algorithm architecture of RJINDAEL.

Our modifications affect the key schedule, which creates the series of round keys, as well as 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.

AES block size

The block size in EAES remains he same as in AES, 16 bytes, 128 bits respectively. The original RJINDAEL 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 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.

Basic RJINDAEL encryption functions

Furthermore, we kept the 4 basic functions of the RJINDAEL transformation within the encryption of each block:

1) *AddRoundKey* - addition in $(\mathbb{F}_2)^{128}$:
   XOR-ing each byte of the state with a block 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 $\alpha \in \mathbb{F}_2[x]/(1 + x^8 + x^4 + x^3 + x + 1)$, 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:
   Mutiplication of each column of the state with the following matrix M:

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

All those basic functions are invertible. Furthermore all of them, except the *SubBytes*, are linear.

RJINDAEL algorithm architecture

EAES also inherits the high level algorithm architecture of RJINDAEL which is described by:

Call the key schedule function *ExpandKey*, and then call per block:

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

PRNGs for key expansion, the essential modification

The original RJINDAEL key schedule expands the original key by applying the S-Box, left circular shifts and bitwise XOR-operations to the words in the state representation of the original key. We renounce on its exact description here, and will refer to other papers in the context of comparisons regarding quantum computing rescources. For now it is mostly important to note that the RJINDAEL key schedule function is invertible as well. This property opened 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. Including 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. We will look at the chosen PRNGs more closely in a dedicated sections.

More rounds and a 512-bit key version

Besides that, 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 RJINDAEL transformation function, which is 8 more rounds than AES-256, and twice the best known attack which breaks 11 rounds [8]. For EAES-512 we chose the number of 30 rounds.

Key schedule variants

We consider cryptographic key derivation functions as appropriate replacements for the invertible RJINDAEL 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 HKDF, a HMAC-based *extract-and-expand key derivation function*. As a second option, we implemented the SHA-3-derived and customizable CSHAKE-function. In both cases, we use SHA-hash-functions as underlying pseudo random functions. Due to the fact that within our algorithm, those functions expect an input which is cryptographically strong already, we renounce on the usage of a salt.

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 [5] 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.

On the other hand, 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 mental 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 globally widely used. 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 cooperation 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 keys. Each 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 $(n+1) \cdot 128$ bit, where n represents the number of rounds. The $+1$ stands for the fact that *AddRoundKey* is called initially and finally at each block encryption.

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. Naturally, we use HKDF(SHA-256) for EAES-256 and HKDF(SHA-512) for EAES-512.

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 chosen hash function, SHA in our case, to generate a pseudo-random 128-bit block.

So SHA-256 has to be called $\lceil \frac{(22+1)\cdot 128}{256} \rceil = 12$ times for EAES-256 and $\lceil \frac{(30+1)\cdot 128}{256} \rceil = 16$ times for EAES-512.

If SHA-512 is used for a HKDF key expansion within EAES-512, it is called $\lceil \frac{(30+1)\cdot 128}{512} \rceil = 8$ times.

CSHAKE, flexible hashing architecture

We implemented the SHA-3-derived and customizable cSHAKE-function as a second option for the key schedule. cSHAKE is designed for 128- and 256-bit security strength [11]. cSHAKE128 uses SHA3-256, and cSHAKE256 is based on SHA3-512.

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 cryptoanalytic breakthroughs might compromise the security of both hash function families.

KECCAK is a so called *sponge function*. That means it is an algorithm with finite internal state, taking an input bit stream of any length and producing an output bit stream of any desired length. Its standardized version SHA-3 has four versions with fixed output lengths: SHA3-224, SHA3-256, SHA3-384 and SHA3-512. The algorithm's low-level primitives are boolean bitwise operations over 64-bit words excluding additions. Avoiding operations which involve carry propagation, enhance performance on hardware implementations such as FPGA and ASIC.

cSHAKE returns SHA-3 if only an input string and an output length are given, which is the case in our implementation. The initial state of SHA-3 is a 5×5 array of 64-bit entries, 1600 zero bits it total. The input is absorbed into the initialized state, and with each call to the inner permutation function of KECCAK, rates of $(1600 - 2 \cdot n,\ n \in \{128, 256\})$ bits are returned. So cSHAKE128 returns 168, and cSHAKE256 returns 136 pseudo-random bytes per call. We choose cSHAKE256 for EAES-256 and EAES-512.

For EAES-256 we have $\lceil \frac{(22+1)\cdot 128}{136 \cdot 8} \rceil = 3$ calls to the permutation function, $+1$ call for the initial state.

For EAES-512 we have $\lceil \frac{(30+1)\cdot 128}{136 \cdot 8} \rceil = 4$ calls to the permutation function, $+1$ call for the initial state.

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 [7]. The complexity of finding a solution to the systems of KECCAK-256 and KECCAK-512 comes down to $2^{78.25} c\kappa$, where c is a constant and κ the condition number. We will look a little closer at the impact of [7] 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 [6] 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 represent the complexity of an algorithm.

1) 128 qubits to hold the current internal state
2) *ExpandKey*: $> 50'000$ gates and a depth of $> 24'000$ 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$> 20'000$ 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 $200'000$, and $1'336$ qubits to achieve the output of each AES-256 system. Quantum resource estimates for Grover's algorithm to attack AES-256 are $3.24 \cdot 2^{151}$ gates, a depth of about $1.64 \cdot 2^{145}$, and $6'681$ 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 $1.27 \cdot 10^{44}$ T-gates, a depth of about $3.76 \cdot 10^{43}$, and 3615 qubits. For SHA3-256 the costs are $2.71 \cdot 10^{44}$ T-gates, a depth of about $2.31 \cdot 10^{41}$, and 3615 qubits. For both functions, the total cost comes down to approximately 2^{162} basic operations.

Both SHA-256 and SHA3-256 are considerably more cost intensive in this context than the original RJINDAEL 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 [6] 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 [7] 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. 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 $\mathbb{C}$ by 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 $\emptyset$ 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. 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 [7] 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.

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 effectively on currently used devices, is compatible with existing hardware implementations and offers higher security under attacks within reach than 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/quant-ph/9508027*, 1995.

[2] 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.

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

[4] 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.*

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

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

[7] 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.

[8] 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.*

[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.