itk AVtobvS Sàrl

AHX-Preliminary Security Analysis

Symmetric block cipher proposition for Post-quantum cryptography

KUDELSKI SECURITY

30.11.2018
Confidential

DOCUMENT PROPERTIES

Version:	01.00
File name:	2018-11-30-Kudelski-AHX-Analysis-v01.00.docx
Publication date:	30.11.2018
Confidentiality Status:	Confidential
Document Owner:	Benoît Gerhard
Document Recipient:	Stiepan Aurélian Kovac
Document Status:	Final
Client Company Name:	itk AVtobvS Sàrl

Recipients

Stiepan Aurélian Kovac
AVtobvS Sàrl
Route de Lullier 81
CH-1254 JUSSY(GE)
Switzerland

Kudelski Contact

In case of questions regarding this document please contact:
Benoît Gerhard
Head of Security Evaluation & Attacks
IoT Security Labs
Route de Genève 22-24
1033 Cheseaux-sur-Lausanne

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

EXECUTIVE SUMMARY

In a context where the amount of research on quantum computers is increasing, threats for classical cryptography appears.

For asymmetric cryptography and due to the efficiency of Shor's algorithm [15], NIST has launched a challenge to solicit, evaluate, and standardize one or more quantum-resistant public-key cryptographic algorithms.

Due to the quantum Grover's algorithm for hash functions with n bits input, the preimage resistance is reduced to $2^{\frac{n}{2}}$ ([12], [13]). It has the same impact on the key search [9], in case of symmetric block ciphers, thus doubling the key size can effectively enable to maintain security level.

The purpose of the AHX would be to offer a solution for embedded devices in the context of 5G and resistance to attacks possible on post-quantum computer.

It has been requested by itk AVtobvS Sàrl to Kudelski to provide some feedbacks regarding the security of the current proposal, that is the main goal of this report.

INTRODUCTION

The first high level security analysis presented in this report is focus on the block cipher called AHX.
The analysis has been conducted using only the source codes available on following links:

- https://github.com/Steppenwolfe65/CEX/tree/master/CEX/AXH.cpp
- https://github.com/Steppenwolfe65/CEX/blob/master/CEX/AHX.h
- https://github.com/Steppenwolfe65/CEX/blob/master/CEX/HKDF.cpp
- https://github.com/Steppenwolfe65/CEX/blob/master/CEX/HMAC.cpp

Others files might have been taken into account but due to short time constraint, it has not been possible.

Note that other similar block cipher proposals can be found in the CEX crypto library: THX which is based on the Twofish block cipher, and SHX which is based on the Serpent block cipher. The difference is the underlying block cipher. AHX is indeed the most interesting of such instances, because optimized hardware and processors are generally available for the AES block cipher.

1. AHX OVERVIEW

1.1. High level proposition

AHX seems to have a flexible design to enable the user to handle the required security level according to the context.

Key lengths from 128 to 2048[1] bits seem to be possibly handled, but for the 2048-bit use-case the hash function to use in the key schedule is not consistent to the security level expected or is missing.

The selection of the security level is also linked to time and before primitive selection following question should be answered: how long shall the data be safe?

One goal of this analysis will be to give first elements regarding the targeted security level.

In Figure 1 we have summarized the proposal.

[1] In comment the key length seems to be limited to 1024 bits but there no boundary check on key length.

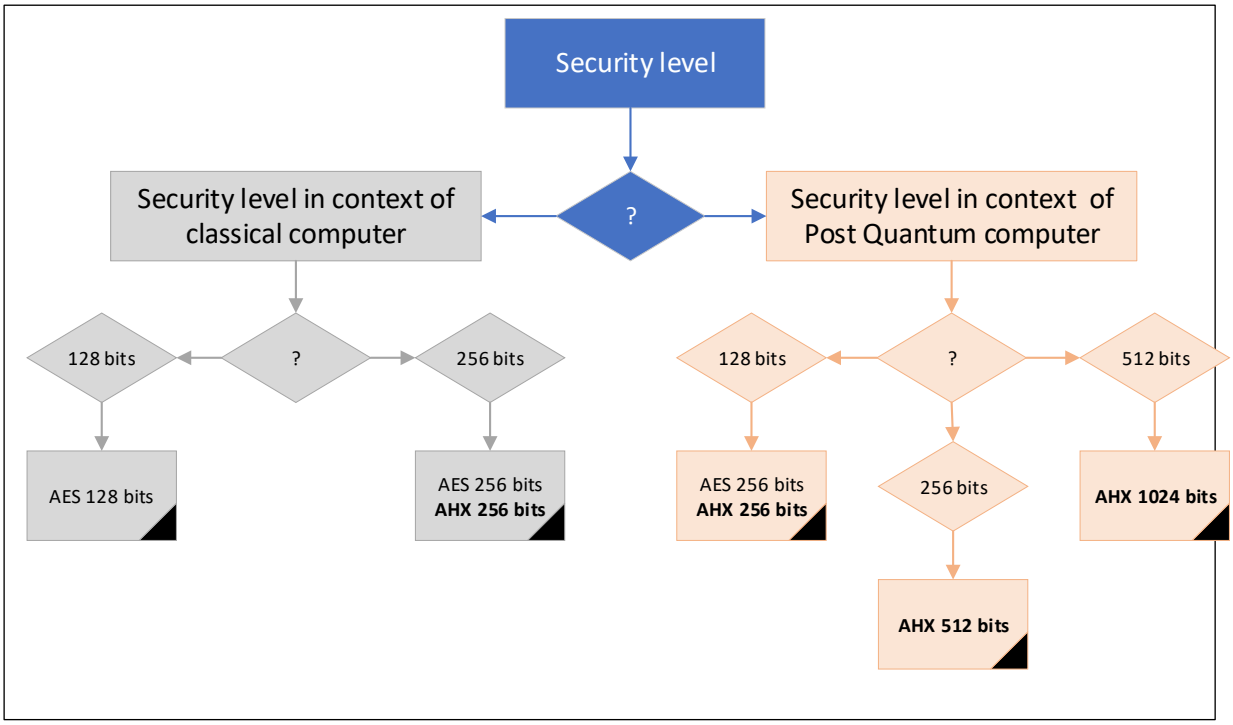

Figure 1: Security levels targeted according to algorithms

1.2. AHX Key schedule

AHX can use two different key schedules.

The first one is the standard AES key schedule, which is fully specified for AES for 128, 192 and 256-bit keys. AHX extends it to be able to process 512-bit keys. The length of the key determines the number of rounds (and round keys); given $k \leq 512$ the number of key bits, the number of rounds is given by $R_{nb} = \frac{k}{32} + 6$.

The second key schedule is a HKDF based on HMAC with selectable hash primitive. The chosen hash function can be any of those implemented by the CEX library (e.g. Blake2, Skein, SHA3). By looking at the code we can see that possible choices are SHA-2 (256 & 512) and the more recent SHAKE-256 and SHAKE-512 hash functions. The round keys are generated by calling iteratively the HKDF until all needed bytes are generated. The number of rounds in this case is given by $R_{nb} = Min\left\{\frac{k}{32} + 14, 38\right\}$.

Key length (in bits)	Hash function in the key schedule
256	SHA256
512	SHA256
1024	SHA512
2048 (?)	SHA512 (?)

Table 1: Hash functions used in the extended key schedule according to key length

1.3. AHX Block cipher

AHX block cipher round is fully equivalent to AES one.

As defined in the key schedule based on HKDF, the number of rounds is defined according to the input key size.
Given $k \leq 2048$ the number of key bits: $R_{nb} = Min\left\{\frac{k}{32} + 14, 38\right\}$.

Key length (in bits)	Rounds numbers
256	22
512	30
1024	38
2048 (?)	38 or 78 (?)

Table 2: Round numbers according to key length

Figure 2 below shows an overview of the AHX execution when using the HKDF-based key schedule with SHA256 underlying hash function and for key length that should not exceed 512 bits.

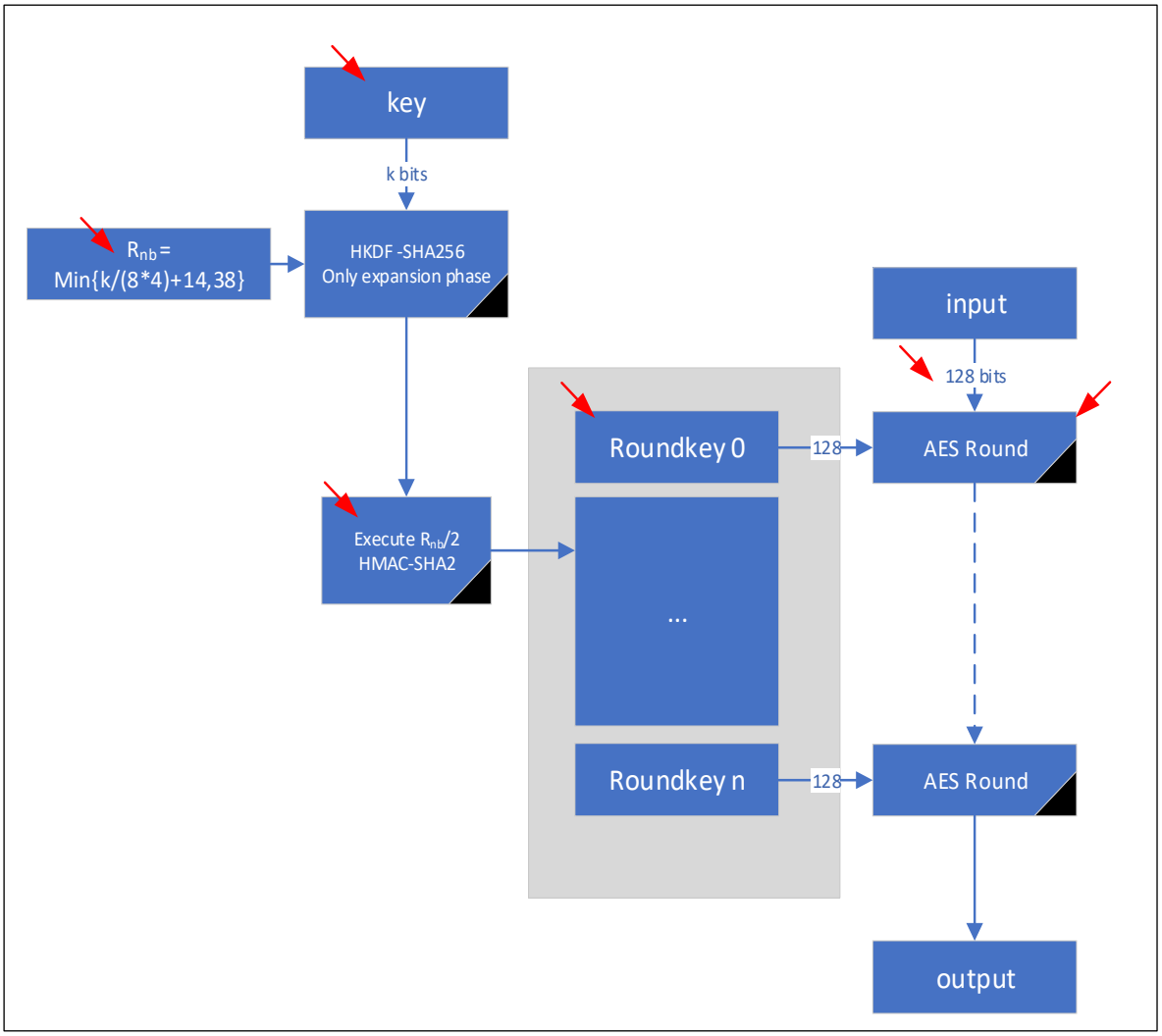

Figure 2: AHX global scheme

2. SECURITY ANALYSIS

In Figure 2 we have pointed in red the elements that must be handled carefully to guarantee the security level of the whole algorithm.

2.1. Consistency of the primitives

In the library CEX, it is proposed to manage the security consistency of the cryptographic primitives by the LegalRounds() and LegalKeySize() functions in the code, it might be interesting to enforce that.

The construction proposes others hash functions for the key schedule and should include the alternatives for the block cipher rounds construction. Indeed, if a weakness is found on the AES structure in the future, it might be possible to have it whatever the key size.

2.2. Key

Regarding the key element and according to security level expected, a specific attention must be paid to following elements and which are partially or not covered by the proposal:

- The enforcement of the key size in bits
- The quality of the random number generator used to guarantee the expected entropy of the key.
- The secure storage of the key including confidentiality and integrity.

2.3. Input Block size

A block cipher input block size should be such that no complete dictionary can be built for a given key. Nowadays it is quite common to set the block size to 128 bits, which is considered to completely protect from such brute force attacks.

Moreover, if we consider the input/output size of the block cipher, there are $(2^{128})!$ possible permutations. Whatever the key lengths considered -256, 512 or 1024 bits (even 2048 bits) this is much larger than the key spaces, so collisions are not expected.

However, some academic results have alerted that it might not be enough to just double the key length of the symmetric primitives to block the attackers from the post-quantum world [8]. And given that the AHX is proposed to handle 512-bit security level (and maybe more) - considering quantum resistance, the block size bit length should also be taken into account and solution to handle larger block size should be investigated.

We argue that maybe the choice could be to switch to 256 bits block size, although a thorough analysis of attack scenarios should be produced to justify the choice.

We note that the original Rijndael proposal already specifies the possible choice of 256-bit block size, possibility that has not been retained (or judged valuable) for the AES standard.

2.4. Key schedule

It appears important here to recall security breach published on AES 256 bits key. Indeed, it has been proved by researchers that AES-192 and AES-256 are weaker than AES-128 against some classes of attacks [6].

This can be explained at very high level by the fact that AES-256 tries to squeeze two times the key information into a construction that was essentially built for 128-bit keys, and this has serious side effects.

The attacks are related-key attacks which requires the cryptanalyst to have access to plaintexts encrypted with multiple keys that are related in a specific way -a scenario which is debatable. They are also very far to be practical.

However, this work casts serious doubts on the possibility to extend the AES to bigger key sizes, at least on theoretical grounds.

This can have serious implications for the first key schedule of AHX, namely the one fully based on AES but extended to accept 512-bit keys. In our opinion the author of AHX would have to show or at least mention that such attacks are not feasible against its proposal of extended AES key schedule.
It is not possible for us to validate the hypothesis, given the limited time available for this preliminary analysis.

The related key attack scenario is certainly not applicable to the second proposed AHX key schedule, namely the one based on HKDF. In this case round keys are derived with a one-way cryptographic function, which has higher computational cost, but certainly renders the proposal more attractive.

HKDF is well known and studied. If the hash function is correctly chosen and implemented, it enables to produce many more round-keys keeping maximal entropy for each round key.

The choice of the KDF used to generate the round keys essentially determines the upper bound on the security of the scheme. The HKDF is built on HMAC primitive, which in turn is based on a hash function.

In the code comments, we can clearly see the following choices mentioned:

- SHAKE256
- SHAKE512
- HKDF-SHA2-256
- HKDF-SHA2-512

The choice poses constraint on the actual key space of AHX.

SHA2-256 and SHAKE256 have claimed security of 256 bits against pre-image attacks (collision attacks are not meaningful in the KDF scenario) [Note that SHAKE256 should be used in this case with at least 512-bit output].

This is coherent with the fact that the main key can be up to 512 bits long; in fact, Grover's algorithm [9] run on a quantum computer, would break such keys with a 2^{256} effort, which is a value coherent with the choice of either SHA2-256 or SHAKE256.
Therefore, such choice is possible if the key of AHX is no longer than 512 bits.

For longer keys, one could choose either SHA2-512 or SHAKE512, but knowing that for the same motivations outlined above, in this case the key space of AHX cannot exceed 1024 bits.

Thus, in the context of post quantum computer and considering an attacker could exploit one round key leakage to get the original key using pre-image attack, we would recommend the following:

- SHA256-SHAKE256 for key up to 512 bits to guarantee a security level up to 256 bits
- SHA512-SHAKE512 for key up to 1024 bits to guarantee a security level up to 512 bits

It is at this moment unclear how one would attain a key space for AHX longer that 1024 bits.
We think it would be desirable to limit AHX key length to 1024 bits.

The info and salt elements that are provided to the HKDF are less critical but could bring additional entropy if needed.

The one-way property of the second proposed key-schedule is an asset to protect the implementation regarding attacks; an attacker would need to get all the round-keys to break the full encryption function (if the original key is well protected and does not leak during first HMAC execution).

2.5. Rounds number

The formula for choosing round number should be justified. It is in general quite dangerous for a block cipher to let the user decide the number of rounds, because that would lead to trivial attacks (differential analysis of n and n+1 rounds).

Examining AHX code, it seems that round number is implicitly determined by the key length, so it seems correct (not freely selectable by user), and via the KDF a different key length would in any case lead to different KDF output. We only mention that in source code comments it is reported that number of rounds is selectable by user, which seems to contradict what has been said.

We think that a more desirable method would be to let the number of rounds enter the KDF function as ancillary data. This would enforce the security.

Regarding the number of rounds, we would like to state the following: the amount of (round) key bits that can be introduced into a round of AES is 128. If the AES standard proposed 10 rounds for 128-bit keys, we would conservatively assume that AES with 256-bit keys should feature 20 rounds.

The AES standard dictates 14 rounds for 256-bit keys, and in fact we have seen that its security is far from ideal under certain scenario (related key attacks).

If we conservatively assume that 10 rounds would be needed for every 128 bits of key material, this would mean 40 rounds for 512-bit keys.

The formulas chosen by the AHX author are less conservative, as they prescribe:

- 30 rounds for 512-bit keys using the KDF key schedule
- 22 rounds for 512-bit keys using the extended AES key schedule

The choice seems particularly non-conservative for the extended AES key schedule. As we already discussed, the KDF key schedule seems a more robust proposal overall.

Deeper analysis should be conducted for 1024-bit and 2048-bit keys use cases.

2.6. Implementations

The AHX cipher can be implemented in software and dedicated hardware (ASIC and/or FPGA). It is based on well-known cryptographic primitives, and from a functional point of view should not pose problems for a skilled coder/designer.
According to the context, we want to mention that implementations should be secure against active (fault) and passive (side-channel) physical attacks (exploiting for example cache and/or timing differences).
While techniques for doing that are quite known in the literature, especially for software implementations, the addition of more cryptographic primitives (KDF, HMAC, hash functions) that manipulate key material means that every one of these primitives should be secured against such attacks, making the complexity and therefore the cost of the implementation higher.
In addition, for pure software implementation specific attention must be paid to keys, round keys storage and entropy generation.

It is certainly possible to implement the AHX block cipher with KDF key schedule in a dedicated chip (ASIC) and programmable logic (FPGA). It would require deeper analysis regarding memory constraints, performances requirements and physical attacks. of the whole block cipher.

DOCUMENT HISTORY

Version	Status	Date	Authors	Comments
00.01	Draft	**28.11.2018**	**M. Macchetti- K. Villegas**	Preliminary analysis
01.00	Final	**30.11.2018**	**M. Macchetti- K. Villegas**	Final document

Reviewer	Position	Date	Document Version
Benoit Gerhard	Head of Security Evaluation & Attacks	**30/11/2018**	01.00

Approver	Position	Date	Document Version

DOCUMENT RECIPIENTS

Name	Position	Contact Information
Stiepan A. Kovac		**stie@itk.swiss**

KUDELSKI SECURITY CONTACTS

Name	Position	Contact Information
Benoît Gerhard		**benoit.gerhard@nagra.com**

REFERENCES

#	References Descriptions
[1]	https://github.com/Steppenwolfe65/CEX/tree/master/CEX
[2]	https://tools.ietf.org/pdf/rfc5869.pdf HMAC-based Extract-and-Expand Key Derivation Function (HKDF)
[3]	https://csrc.nist.gov/projects/post-quantum-cryptography
[4]	D. J. Bernstein (2009). "Introduction to post-quantum cryptography". (Introductory chapter to book "Post-quantum cryptography").
[5]	D.J. Bernstein, Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete?, 2009
[6]	A. Biryukov and D. Khovratovich, Related-key Cryptanalysis of the Full AES-192 and AES-256, 2009
[7]	Joan Daemen and Vincent Rijmen, The Design of Rijndael, AES - The Advanced Encryption Standard, Springer-Verlag 2002 (238 pp.)
[8]	Xiaoyang Dong, Bingyou Dong, Xiaoyun Wang,Quantum Attacks on Some Feistel Block Ciphers, Journal of latex class files, VOL. 14, NO. 8, 2015
[9]	Grover L.K.: A fast quantum mechanical algorithm for database search, Proceedings, 28th Annual ACM Symposium on the Theory of Computing, (May 1996) p. 212
[10]	Grover L.K.: From Schrödinger's equation to quantum search algorithm, American Journal of Physics, 69(7): 769-777, 2001. Pedagogical review of the algorithm and its history.
[11]	Grover L.K.: QUANTUM COMPUTING: How the weird logic of the subatomic world could make it possible for machines to calculate millions of times faster than they do today The Sciences, July/August 1999, pp. 24–30.
[12]	Grover L.K, Quantum mechanics helps in searching for a needle in a haystack, Physical Review Letters 79 (1997), 325–328.
[13]	Grover L.K, Terry Rudolph, How significant are the known collision and element distinctness quantum algorithms ? Quantum Information & Computation 4 (2003), 201–206. MR 2005c:81037. URL: http://arxiv.org/abs/quant-ph/0309123
[14]	Peter W. Shor, Algorithms for quantum computation: discrete logarithms and factoring., in [7] (1994), 124–134.
[15]	Peter W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Journal on Computing 26, 1997
[16]	Paul C. van Oorschot, Michael Wiener, Parallel collision search with application to hash functions and discrete logarithms, in [2] (1994), 210–218
[17]	Paul C. van Oorschot, Michael Wiener, Parallel collision search with cryptanalytic applications, Journal of Cryptology 12 (1999), 1–28
[18]	John Underhill, CEX++ 1.0- An Introduction to the CEX Cryptographic Library, owner@vtdev.com,July 03, 2017

Table 3: References