From:	pqc-forum@list.nist.gov on behalf of Markku-Juhani O. Saarinen <mjos.crypto@gmail.com>
Sent:	Saturday, September 2, 2023 9:02 AM
To:	pqc-forum
Subject:	[pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: AIMer

Summary

We observe that the bit complexity of recovering x in AIM-1/3/5 is lower than the key search circuit for AES-128/192/256. Since the inversion of AIM is equivalent to solving the AIMer signature scheme secret key from its public key, the security of AIMer does not appear to meet its security claims.

Similar (or quantitatively somewhat better) observations have also been made in [1], which presents an algebraic attack that sheds 2^13,2^14,2^15 from operations against AIM-1/3/5. Based on their analysis, one can estimate that the security of AIMer falls short of the target by at least 16 bits.

[1] Fukang Liu, Mohammad Mahzoun, Morten Øygarden, Willi Meier, "Algebraic Attacks on RAIN and AIM Using Equivalent Representations." IACR ePrint 2023/1133 https://eprint.iacr.org/2023/1133

The extremely simple structure of AIM encourages one to look for significantly faster attacks; AIM would be almost revolutionary as a symmetric primitive if it proves to have even this level of security. I've made a simple Python implementation of AIM-1/3/5 (with some test vectors) available at https://github.com/mjosaarinen/aim-sym-py/blob/main/aim.py to encourage further analysis by cryptographers.

Background

AIMer is a first-round PQC on-ramp candidate that builds a signature scheme from a symmetric one-way function (named AIM) and a non-interactive zero-knowledge proof of knowledge (NIZKPoK) system based on the MPC-in-the-Head paradigm.

AIMer Public key consists of a pair pk = (iv, c) where "c" is the output of the one-way function c = AIM(x, iv). The secret key is the quantity x (slightly confusingly, "plaintext" in the context of the AIM function.).

A direct way of breaking AIMer is to attack its one-way function: Solving the secret key x, given (iv, c) from the public key. There is no need to consider the NIZKPoK scheme in this type of attack; one determines the secret key and then constructs forgeries using the standard signing procedure.

Brief description of AIM

Let n in {128,192,256} be the security level for AIM-I/III/V. This is also the secret key "x" size. AIM uses binary fields size of F = GF(2^n).

The finite field exponentiation function Mer[e](x) = x^(2^e-1) is the only nonlinear component in the algorithm. Internally, SHAKE is used to derive affine transforms from "Ai": two or three invertible n*n binary matrices A1, A2(, A3) and an n-bit vector/field element "b." However, the complexity of SHAKE does not affect the complexity of the search for secret key "x" -- the affine transform can be considered a constant in this task.

In the following, + is a field addition (XOR), ^ is exponentiation, * is a binary vector-matrix multiplication. The AIM one-way functions are:

AIM-1: c = ((x^(2^3-1))*A1 + (x^(2^27-1))*A2 + b)^(2^5-1) + x
AIM-3: c = ((x^(2^5-1))*A1 + (x^(2^29-1))*A2 + b)^(2^7-1) + x
AIM-5: c = ((x^(2^3-1))*A1 + (x^(2^53-1))*A2 + (x^(2^7-1))*A3 + b)^(2^5-1) + x

There is no iteration. In terms of symmetric cryptography, AIM could be characterized as a 1-round or 2-round SPN PRF.

Simple Key Search Manipulations

Recall that in a binary field, we have (x+y)^2 = x^2 + y^2, and squaring is bitwise linear. Hence, repeated squaring (i.e., computing x^(2^n)) is also linear. We use "Ei" matrices to represent power-of-2 exponentiations; x*Ei = x^(2^i) and x*EMi = x^(-2^i) = x^(2^(n-i)).

Substitute z = x^-1. If this variable is available, we can turn the nonlinear exponentiations into a single multiplication: Mer[e](x) = Ee*z.

In case of AIM-1, the search of "x" satisfying

    c == ((x^(2^3-1))*A1 + (x^(2^27-1))*A2 + b)^(2^5-1) + x

.. can be transformed to remove all exponentiation operations ..

    u = x*E3*z*A1 + x*E27*z*A2 + b
    u*E5 == (c + x)*u

In a search for "x" satisfying the condition, one generates a sequence of x1,x2,x3.. and corresponding z1,z2,z3.. z=x^-1 at the same time using a generator and its inverse. The generator for z is not much more complex than x if LFSRs are used.

We can observe that this circuit consists of 7 multiplications and some n-bit xors. Collapsing the ANDs and XORs of constant binary matrix A1,A2,E3,E5,E27 multiplication into netlist optimization should allow significant optimizations. It is much simpler than an AES-128 key search circuit (which contains AES subkey expansion and trial encryption functions.) Handling the "z" inverse sequence in a quantum oracle may be trickier than in a classical setting, however.

Best Regards,
- markku
--

In the paper, we extensively strengthen algebraic cryptanalysis part as 3 out of 4 analyses are related to the algebraic characteristic.

As we recognized [4] very recently, our paper contains [1, 2, 3], but not [4].

Nevertheless, we found that the constant addition included in our update effectively mitigates the attack in [4].

This update does not affect efficiency much.

The signature size will remain same, and signing and verifying time will be slightly increased (expected ~10%).

We plan to incorporate AIM2 to AIMer, which will be dubbed AIMer2.

The specification document and implementation results will also be updated.

We truly thank all the authors above for pointing out the vulnerabilities of AIM.

Third-party analysis is always welcome!

Best regards,

Seongkwang Kim on behalf of the AIMer team