Code-based signature scheme derived from a MRHS representation of an AES encryption

Pavol Zajac *
Slovak University of Technology in Bratislava, Slovakia

Extended Abstract

We propose a new post-quantum signature system that is based on MRHS equations related to the design of symmetric encryption standard AES (or any block cipher in general). In this extended abstract we first describe concrete basic steps and parameters of the algorithm based on parameters of AES-128. In the following part, we discuss the efficiency and security of the system.

Note: This is the second version of the scheme. We thank anonymous reviewers for pointing out some of the security problems of the original version.

Algorithm description

Key generation algorithm. The secret key of the algorithm consists of an AES key k, and a permutation π on $9 \cdot 16 = 144$ numbers (which is the number of S-boxes used in AES encryption excluding the first round). To construct the public key, we must do the following:

1. Expand the key $k \in \mathbb{Z}_2^{128}$ with shortened AES key schedule to $K = (k_0, k_1, \ldots, k_9) \in (\mathbb{Z}_2^{128})^{10}$ (last subkey is unused).
2. Express AES computation as a system of MRHS (see e.g. [2]) equations $(X+K, Y)\cdot\mathbf{M} \in S_1 \times S_2 \times \cdots \times S_{160}$. Each S_i is a set of 256 vectors $S_i = \{(x, S(x)); x \in \mathbb{Z}_2^8\}$, where S is a standard AES S-box. Matrix $\mathbf{M}$ corresponds to linear parts of the AES encryption (AddRoundKey, ShiftRows, MixColumns). Variables $(X + K)$ represent combinations inputs of each internal S-box of AES encryption, and Y is the output of the last round S-box layer. Symbolic addition $X + K$ can be replaced by a constant addition using precomputed round keys. Next, we apply permutation π on the order of S-boxes 17 to 160 (and corresponding blocks of $\mathbf{M}$). After extracting the constants, we get a final form of the system:
$$X \cdot \mathbf{M}_\pi + c \in S_1 \times S_2 \times \cdots \times S_{16} \times S_{16+\pi(1)} \times \cdots \times S_{16+\pi(144)} \quad (1)$$
3. Compute **systematic** (parity check) matrix $\mathbf{H}$, such that $\mathbf{M}_\pi \cdot \mathbf{H}^T = \mathbf{0}$.
4. Compute $q = c \cdot \mathbf{H}^T$.
5. The public key consist of a pair $(\mathbf{H}^T, q)$.

Signature algorithm. Let $m \in \mathbb{Z}_2^{128}$ be a message we want to sign. Let h be a cryptographically secure hash function, with 128-bit output, and r a random value (128 bits). Encrypt $h(m, r) + k$ with AES using key k, and store vector of 144 bytes s_i. Note

*This research was sponsored in parts by the NATO Science for Peace and Security Programme under grant G5448, and by Slovak Republic under grant VEGA 1/0159/17.

that $h(m, r)$ will become input to the first S-box layer of the AES encryption. The 144 s_i's represent inputs to S-boxes from rounds 2-10. Signature of m is then sequence of bytes $s_{\pi(i)}$, for $i = 1, 2, \ldots, 144$, plus the 16 bytes of random value r.

Verification algorithm. Construct vector v consisting of 160 blocks of 16 bits. Each block of bits consists of pair $(x, S(x))$, where S is an AES S-box. For the first 16 blocks, x bytes are taken from $h(m, r)$, then bytes of the signature in sequence. Signature is valid, if $v \cdot \mathbf{H}^T = q$.

Correctness. During signature, we select pairs $(x, S(x))$ such that system equation (1) holds for this concrete vector v. Because $X \cdot \mathbf{M}_\pi \mathbf{H}^T = \mathbf{0}$ for any X, we get equation $c \cdot \mathbf{H}^T = q = v \cdot \mathbf{H}^T$.

Security and efficiency of the system

Efficiency. Signature size is 160 B = 1280 bits, which corresponds to a standard RSA signature size. Public key size is $(2560 + 1) \times 1152$ bits, which is 360 kB. This is similar to the standard McEliece cryptosystem [1]. We can compress the public key size to 198 kB, if $\mathbf{H}$ is in a systematic form and we only store only the relevant part of $\mathbf{H}$.

All operations used in the signature scheme are very simple and fast. Key generation requires a simple linear algebra (system $\mathbf{M}$ can be precomputed). Signature generation algorithm requires a single one AES encryption (most of the AES implementations can be modified to provide us inputs of S-boxes used). Verification algorithm requires a single vector-matrix multiplication.

Security. Informally, a signature system is secure, if no (poly-time) attacker is able to forge signatures on new messages. The main attack vectors in our design are as follows:

- Verification algorithm consists of verifying the identity $v \cdot \mathbf{H}^T = q$. There is an exponentially large number of solutions v, but only some of them represent a valid signature. Moreover, from a specific solution $c \cdot \mathbf{H}^T = q$ that corresponds to $X = 0$, we can compute the AES secret key. Computing a low-weight vector v given the pair $(\mathbf{H}, q)$ corresponds to a an NP-hard decoding problem, which is believed to be hard to solve even on quantum computer. In our case, there are two principal differences from a generic case:

  1. We want to find a very specific subset of solutions, or a special solution c, which are not guaranteed to be low-weight solutions. In a case with random $\mathbf{H}$ we suppose that this should be at least as hard as finding a low-weight solution (but we do not have a proof).
  2. Matrix $\mathbf{H}$ has a special structure given from the construction of the system. It is a parity-check matrix of a permuted code generated by $\mathbf{M}$, which is derived from linear layers of AES. It is not clear whether the permutation in this case is sufficient to create a hard decoding instance.

- Suppose that the attacker cannot recover c from $(\mathbf{H}, q)$, but can recover permutation π. He can then compute the secret key by matching outputs of round i of the AES encryption (vector u) with inputs to round $i + 1$ (vector w). Clearly $k = w \oplus \mathbf{L} \cdot u$, where $\mathbf{L}$ expresses the linear part of AES round transform.

  It is not clear, whether the attacker can recover π from analysing $\mathbb{H}$. Original matrix $\mathbf{M}$ is very sparse and have a specific structure that can help to recover π from $\mathbf{M}_\pi$. However, recovering sparse $\mathbf{M}_\pi$ from $\mathbb{H}$ should be as difficult as a general decoding problem above. Moreover, to further mask the structure (and for efficiency reasons) we hide the structure of the code by publishing the systematic form of $\mathbb{H}$.

- Due to the structure of AES rounds, attacker does not need whole π, but might be able to separately reconstruct the 32-bit parts of the key from finding matching 8 S-boxes. If the attacker tries to do this randomly, he can four times select 8 correct

S-boxes (out of 144) with probability 2^{-167}. However, as anonymous reviewer pointed out, a statistical analysis of known signatures can provide more information to the attacker. To partially address the problems with statistical analysis, we have added a requirement to apply a hash function, and a random value r (salt), to the input of encryption. Thus attacker cannot directly control what is being computed, although he can make statistical observations from known signatures. A more research is required to fully address this problem.

- Given valid signature v, such that $v \cdot \mathbf{H}^T = q$, attacker can compute any other v' by adding a codeword w of the code generated by $\mathbf{M}_\pi$, with $v' \cdot \mathbf{H}^T = q$. If the signature system was based on the Niederreiter-like code-based system, attacker could just use any prefix $h(m', r')$ and find valid $v' \cdot \mathbf{H}^T = q$ by solving a linear system of equations (or even find a different signature for the same message).

  However, our scheme has an additional property: only a half of the vector v is provided in the signature, second half is computed using AES S-boxes. This means that valid signatures are forming only a (very small and non-linear) subset of the possible code cosets. Each valid signature is a solution of the system equation 1, which represents an AES encryption. As AES is a permutation for each key, there is a unique sequence of state bits and thus a unique signature for each message.

  To summarize: A signature oracle for our scheme provides a permuted internal state of the AES (for an unknown plaintext and key, but known input to first layer of S-boxes). Signature forgery means that we can find a new related AES state given this information from multiple previous signatures.

Concluding remarks

The proposed design should be not be considered a secure signature scheme, as our assumptions are heuristic. The signature scheme relies on hardness of the decoding problem / MRHS problem. However, it is not clear whether the specific decoding problem is as hard as a random decoding instance. Furthermore, it is not clear whether the linear trapdoor used to mask an easy problem (AES encryption with known key) is sufficient to hide the structure of the system and secret parameters.

The proposed system opens an interesting research area and provides a different (and more efficient) way to provide a signature scheme from a secure blockcipher. Note that the scheme can be instantiated with essentially any secure symmetric cipher (not only AES)[1]:

- the key generation algorithm of the signature scheme is based on the corresponding MRHS representation of the selected cipher;
- the signature generation and verification is similar to the scheme presented here.

It is an open problem of what is the effect of the design of the symmetric cipher on the security of the corresponding signature scheme. It is also an open question whether we can also provide a public encryption scheme based on similar symmetric cipher representation.

References

[1] McEliece, R. J.: *A public-key cryptosystem based on algebraic coding theory.* DSN progress report 42/44 (1978), 114–116.

[2] Zajac, P.: *MRHS equation systems that can be solved in polynomial time.* Tatra Mountains Mathematical Publications, 67/1 (2014), 205–219.

[1] A demo version based on simple SPN will be available at `https://github.com/zajacpa/SPNsig`