A New Type of Signature Scheme Derived from a MRHS Representation of a Symmetric Cipher

Pavol Zajac, and Peter Špaček

***Abstract*— We propose a new concept of (post-quantum) digital signature algorithm derived from a symmetric cipher. Key derivation is based on a system of Multiple-Right-Hand-Sides equations. The source of the equations is the encryption algorithm. Our trapdoor is based on the difficulty of creating a valid transcript of the encryption algorithm for a given plaintext (derived from the signed message): the signer can use the encryption algorithm, because he knows the secret key, and the verifier can only check that the solution of the equation system is correct. To further facilitate the verification, we use techniques from coding theory. Security of the system is based on the difficulty of solving MRHS equations, or equivalently on the difficulty of the decoding problem (both are NP hard).**

***Index Terms*—Signature scheme, Substitution-Permutation Network, MRHS equation system, post-quantum.**

I. INTRODUCTION

We propose a new concept of (post-quantum) digital signature algorithm derived from a symmetric cipher. There are already some signature algorithms that use symmetric primitives as their basis: hash-based signatures, that use one-way property of the underlying hash function (e.g.,SPHINGS+ [1]), and generic schemes based on non-interactive proofs and multiparty computation (e.g., Picnic [2]).

Main innovation of our design is that it does not use underlying cipher as a black-box, but instead as a white-box. This might seem similar to white-box cryptography [3], but our goal is different. While white-box cryptography models the user as a potential attacker, we use white-box version of the cipher to provide a secret algorithm for signatures for a legitimate owner of a secret key. The recipient that verifies the signature does not have access to the white-box, but is instead provided a public key that is created from the cipher representation.

Our design is mostly related to multivariate signatures [4]: Public key is essentially a system of equations, that only the signer can solve (with the help of the secret key). Unlike multivariate case, we use a different representation of equation systems, so called Multiple-Right-Hand-Sides equations (MRHS, [5]). The source of our equations is the encryption algorithm. Our trapdoor is based on the difficulty of creating a valid transcript of the encryption algorithm for a given plaintext (derived from the signed message): the signer can use the encryption algorithm, because he knows the secret key, and the verifier can only check that the solution of the equation system is correct. To further facilitate the verification, we use techniques from coding theory.

In Section II, we summarize the notation, basic definitions and notions required to understand the proposed scheme. The scheme itself is specified in Section III. We provide a simplified example of some steps of the algorithm in Section IV. In Section V we discuss the correctness of the scheme, as well as its efficiency. Finally, in Section VI we analyse the security of the proposed scheme. The security of the system is based on the difficulty of solving MRHS equations, or equivalently on the difficulty of special decoding problem. Both of the problems are NP-hard (in generic version), and should be difficult to solve with quantum computer as well.

Our original goal was to base the signature scheme directly on the symmetric encryption standard AES. Main advantage would be prevalent existence of hardware and software implementations of AES on essentially any platform. As our security analysis shows, it is not clear, whether the scheme can be made secure with the underlying design of AES, which is strongly structured, and this structure might leak the used trapdoor. A more suitable underlying cipher might be LowMC [6] or similar designs. We believe that to construct a fully secure signature scheme, a new type of symmetric cipher should be designed in a way that will facilitate our trapdoor type. We leave these questions open for further research.

II. DEFINITIONS

We presume that the reader is acquainted with basic cryptographic definitions such as cryptosystem, (symmetric) cipher, public-key encryption, signature scheme, hash function, etc., as well as the related security notions. We also suppose that the reader is familiar with basic notions of coding theory, such as generator and parity-check matrix.

A. Notation

In this article we will use the following notation:

- All bit operations are represented in algebraic way over field $GF(2)$, shortened to $\mathbb{F}_2$. Note that standard operator $+$ in this field corresponds to a logic operation XOR.
- Sets are denoted by block form, e.g. $\mathbb{R} \subset \mathbb{F}_2^m$.
- Integers are represented by simple notation $i, j, n \in \mathbb{Z}$.
- Vectors of bits are denoted by bold variables such as $\mathbf{u}, \mathbf{x}$. The dimension of the vector depends on the context, and is introduced when defining the vector, e.g., $\mathbf{x} \in \mathbb{F}_2^n$. In our paper, all vectors are always row vectors.

Slovak University of Technology in Bratislava, Slovakia.
e-mail: pavol.zajac@stuba.sk

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.

DOI: 10.36244/ICJ.2019.4.4

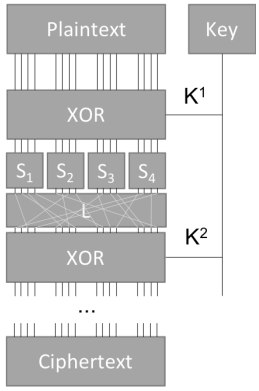

Fig. 1. An example of a subsitution-permutation network.

- Matrix is represented by a bold uppercase letter, with dimensions either depended on the context, or directly defined in the definition of the matrix: $\mathbf{M} \in \mathbb{F}_2^{(n\times m)}$.
- Functions are denoted by uppercase letters, such as F, H, R, S, e.g., $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$.
- Greek notation, such as π, is reserved for permutations of numbers $1, 2, \ldots, n$ (for some n defined in the context).

B. Substitution-permutation network

Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a vectorial Boolean function. In cryptographic context, F is called an S-box, if n is relatively small, and F is a highly non-linear function used in cipher design. In our paper, we will denote S-boxes always with S, or S_i if we need to number them.

Substitution-permutation network (SPN) is a type of symmetric cipher, in which the encryption algorithm consists of multiple rounds (number of rounds will be denoted by r). Each round consists of three basic steps:

1) key addition: $\mathbf{y} = \mathbf{x} + \mathbf{k}^i$, where $\mathbf{k}^i$ is a round subkey;
2) non-linear layer of S-boxes: $\mathbf{y}_{i\cdots j} = S(\mathbf{x}_{i\cdots j})$;
3) a linear diffusion layer (in basic case just a permutation of bits): $\mathbf{y} = \mathbf{x}\mathbf{M}$.

In the last round of SPN, linear layer is typically replaced by another key addition. Round subkeys are derived from the main key $\mathbf{k}$ by a specific algorithm call a key schedule, i.e., $(\mathbf{k}^1, \mathbf{k}^2, \ldots, \mathbf{k}^{(r+1)}) = KS(\mathbf{k})$.

Sequence of internal bits that is used during the encryption (of required granularity, e.g., inputs to S-boxes) is denoted as a transcript of the encryption. If we know the encryption key and the cipher input, the whole transcript can be reproduced by following the computation steps of the SPN. For the attacker, the knowledge of the transcript is typically equivalent to the knowledge of the key. In SPN, if the attacker knows the transcript, he can easily compute round keys (using inverse of the key addition operation), and use the round keys instead of the original key (or the attacker can derive the original key if the key schedule allows it).

Advanced Encryption Standard (AES) is a specific instance of SPN-like cipher Rijndael [7], with linear layer defined by two operations ($ShiftRows$, $MixColumns$, with a specific last round). In our concrete instantiation of the signature scheme, we will use version AES-128, which has 128-bit key and block size, 10 rounds. Each round uses 16 bijective S-boxes on groups of 8 bits. Further details of the AES encryption process are not required to understand the paper.

C. MRHS equations

In our system, we create a non-linear Boolean equation system derived from the selected SPN. While such a system can be written in multiple forms (such as ANF for Gröbner basis method, or CNF for SAT solvers), for our purpose a specific form of a MRHS system [5] is preferable.

Definition 1: [8] Let $\mathbb{F}$ be a finite field. Let $\mathbf{M} \in \mathbb{F}^{(n\times m)}$ be an $(n \times m)$ matrix. Let $\mathbb{R}$ be a set of vectors from $\mathbb{F}^m$. MRHS equation is defined by an inclusion:

$$\mathbf{x}\mathbf{M} \in \mathbb{R}.$$

Vector $\mathbf{x} \in \mathbb{F}^n$ is a solution of MRHS equation, if the inclusion holds for this particular value of $\mathbf{x}$.

MRHS equations related to SPN are centered on S-boxes. Let $\mathbf{x}$ be a vector of variables (e.g. selected unknown transcript bits during encryption with SPN). Suppose that each vector of input bits of an S-box S can be expressed as a linear combination of unknown transcript bits: $\mathbf{u} = \mathbf{x}\mathbf{U}$. Similarly let $\mathbf{v} = \mathbf{x}\mathbf{V}$ be the output vector of the same S-box S. We can then write an MRHS equation for S-box S as

$$\mathbf{x}\,(\mathbf{U}|\mathbf{V}) \in \left\{(\mathbf{u}, S(\mathbf{u})); \mathbf{u} \in \mathbb{F}_2^k\right\}.$$

MRHS (equation) system is a set of MRHS equations, each of which must be satisfied simultaneously for some $\mathbf{x}$ (the particular $\mathbf{x}$ is then a solution of the MRHS system). MRHS system can be written in a similar form to a simple MRHS equation by using a Cartesian product:

$$\mathbf{x}\mathbf{M} \in \mathbb{R}_1 \times \mathbb{R}_2 \times \cdots \times \mathbb{R}_l,$$

where system matrix $\mathbf{M} = (\mathbf{M}_1|\mathbf{M}_2|\cdots|\mathbf{M}_l)$ is composed of matrices of individual MRHS equations.

MRHS system for an SPN is then a set of MRHS equations for each S-box in the system. Unknowns $\mathbf{x}$ in the system must be selected in such a way, that each input and output of the S-box can be expressed as a linear combination of bits of $\mathbf{x}$. Note that we can create a virtual "variable" $\mathbf{1}$ that can express the addition of a constant (0 or 1) when creating the system. After transcribing the system with variable $\mathbf{1}$:

$$(\mathbf{x}, \mathbf{1}) \cdot \begin{pmatrix} \mathbf{M} \\ \mathbf{c} \end{pmatrix} \in \mathbb{R}_1 \times \mathbb{R}_2 \times \cdots \times \mathbb{R}_l,$$

we can rewrite it in extended form as follows:

$$\mathbf{x}\mathbf{M} + \mathbf{c} \in \mathbb{R}_1 \times \mathbb{R}_2 \times \cdots \times \mathbb{R}_l.$$

Constant $\mathbf{c}$ can be transferred to the right-hand side by adding corresponding parts of $\mathbf{c}$ to each vector in $\mathbb{R}_i$.

If MRHS equation has a small number of right-hand sides (members of $\mathbb{R}$), it is easy to solve such a system by repeatedly solving a linear system of equations. Unlike individual equations, an MRHS system has exponentially many right-hand sides (if we try to express the Cartesian product directly). For a general MRHS system, a question of existence of a solution (MRHS problem) is an NP-hard problem [9].

III. ALGORITHM DESCRIPTION

Let us have a symmetric block cipher with encryption algorithm defined by function $Enc : \mathbb{F}_2^n \times \mathbb{K} \to \mathbb{F}_2^n$, which is itself an SPN network with r rounds. Let $S : \mathbb{F}_2^k \to \mathbb{F}_2^k$ be an S-box of the SPN network. We suppose that non-linear part of each of the r rounds of SPN consists of n/k applications of the same S-box S (in parallel). The condition that each S-box is the same is important for the security of the scheme. We will denote the total number of S-box applications by $m = rn/k$. Let $KS : \mathbb{K} \to \mathbb{F}_2^{n(r+1)}$ be a key schedule algorithm.

We define a signature scheme derived from this SPN with the following algorithms: $KeyGen$, $Sign$, and $Verify$.

A. Key generation algorithm

Let $\mathbf{k} \in \mathbb{K}$ be a randomly selected secret key (both of the symmetric cipher, and as a part of a private key of our signature scheme). Furthermore, let π be a randomly selected secret permutation of numbers $1, 2, \ldots, m$, which does not change the order of the initial n/k elements. *Private key* for creating signatures consists of the pair $(\mathbf{k}, \pi)$.

To construct the public key, we must do the following:

1) *Expand the key.* Given $\mathbf{k} \in \mathbb{K}$, we compute the SPN's key schedule: $\hat{\mathbf{k}} = (\mathbf{k}^1, \mathbf{k}^2, \ldots, \mathbf{k}^{(r+1)}) = KS(\mathbf{k})$.
2) *Prepare the MRHS system*. Let $\mathbf{x} \in \mathbb{F}_2^{n(r+1)}$ denote a vector of unknowns corresponding to the inputs of the S-boxes during the SPN evaluation plus one set of S-box outputs in the last round. Inner outputs of the S-boxes can be expressed as linear combinations of variables from $\mathbf{x}$, cipher constants, and constants based on the expanded key $\hat{\mathbf{k}}$. Given linear expressions for the inputs and outputs of the S-boxes, we can construct a MRHS system (as described in section II-C) in the form
$$\mathbf{x}\mathbf{M} + \mathbf{c} \in \bigotimes \left\{ (\mathbf{u}, S(\mathbf{u})) ; \mathbf{u} \in \mathbb{F}_2^k \right\}. \quad (1)$$
Constant $\mathbf{c}$ is derived from the constants of the algorithm and bits of the expanded key $\hat{\mathbf{k}}$. Matrix $\mathbf{M}$ has dimensions $n(r+1) \times 2mk$.
Note that this step can be precomputed up to computation of the final constant $\mathbf{c}$ that depends on $\mathbf{k}$.
3) *Apply masking permutation*. Matrix $\mathbf{M}$ from equation (1) can be written as $\mathbf{M} = (\mathbf{M}_1 | \mathbf{M}_2 | \cdots \mathbf{M}_m)$. Blocks $\mathbf{M}_i$ of dimension $(n(r+1) \times 2k)$ correspond to linear functions that construct inputs and outputs of S-boxes from variables $\mathbf{x}$. Similarly, split $\mathbf{c}$ into blocks of size $2k$ denoted by $(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_m)$.
Apply the secret permutation π on the order of blocks of $\mathbf{M}$ and c. A permuted form of the system is given as:
$$\begin{aligned} &\mathbf{x} \cdot (\mathbf{M}_{\pi(1)} | \mathbf{M}_{\pi(2)} | \cdots \mathbf{M}_{\pi(m)}) \\ &\quad + (\mathbf{c}_{\pi(1)}, \mathbf{c}_{\pi(2)}, \ldots, \mathbf{c}_{\pi(m)}) \in \\ &\quad \bigotimes \left\{ (\mathbf{u}, S(\mathbf{u})) ; \mathbf{u} \in \mathbb{F}_2^k \right\}. \end{aligned} \quad (2)$$
Let us denote the joint matrix of system (2) by $\mathbf{M}_\pi$, and similarly let us denote be permuted vector $\mathbf{c}$ by $\mathbf{c}_\pi$.

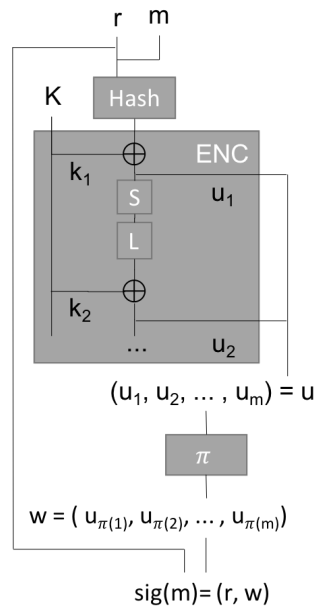

Fig. 2. An overview of the signature algorithm.

4) *Parity check matrix*. Compute **systematic**[1] parity check matrix $\mathbf{H} = (\mathbf{I}|\mathbf{Q})$, such that $\mathbf{M}_\pi \mathbf{H}^T = \mathbf{0}$.
5) *Syndrome*. Compute $\mathbf{q} = \mathbf{c}_\pi \mathbf{H}^T$.
6) *Final public key* The public key consist of the pair $(\mathbf{Q}, \mathbf{q})$.

When applicable, public key should be augmented by "domain parameters" describing the used SPN. These consist of a triplet (n, m, k, S), in order: the block size, the number of S-boxes, the S-box size, and the S-box itself.

B. Signature algorithm

Let $m \in \mathbb{F}_2^*$ be a message we want to sign. Let $H : \mathbb{F}_2^* \to \mathbb{F}_2^n$ be a cryptographically secure hash function[2]. To sign message m do the following:

1) Generate random (nonce) $\mathbf{r} \in \mathbb{F}_2^n$.
2) Let $\mathbf{p} = H(\mathbf{r}|\mathbf{m}) + \mathbf{k}^1$. Here $\mathbf{k}^1$ is the first subkey derived by KS. Note that value $\mathbf{p}$ is constructed in such a way that $H(\mathbf{r}|\mathbf{m})$ is the vector of inputs to the first layer of S-boxes in the first round of the SPN.
3) Compute $c = Enc(\mathbf{p}, \mathbf{k})$ using SPN algorithm. During encryption, store a sequence of S-box inputs as vector $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m)$.
4) Apply secret permutation π to the order of blocks of $\mathbf{u}$, and compute vector $\mathbf{w} = (\mathbf{u}_{\pi(1)}, \mathbf{u}_{\pi(2)}, \ldots, \mathbf{u}_{\pi(m)})$.
5) Signature of m is pair of vectors $(\mathbf{r}, \mathbf{w})$.

[1]We can compute systematic parity check matrix by linear algebra. Simple algorithm is to use (modified) Gaussian elimination to get $\mathbf{M}_\pi$ to form $(\mathbf{Q}^T|\mathbf{I})$. Note that we cannot change the order of columns during the Gaussian elimination (if no pivots are available, we need to restart the algorithm, or change π to swap blocks as required).

[2]We require that H is one-way and collision resistant.

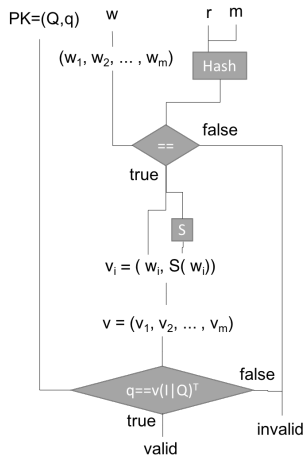

Fig. 3. An overview of the verification algorithm.

C. Verification algorithm

Let $m' \in \mathbb{F}_2^*$ be a message with a supposed signature $(\mathbf{r}, \mathbf{w})$. Let $(\mathbf{Q}, \mathbf{q})$ be the corresponding public key of the signer. To verify whether the signature is valid, perform the following steps.

1) Split $\mathbf{w}$ into m blocks of size k, $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m)$.
2) Let $\mathbf{h} = H(\mathbf{r}|\mathbf{m})$. Verify that $\mathbf{h} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{n/k})$. If not, signature is invalid.
3) Construct m vectors $\mathbf{v}_i$ by using specified S-box S to compute $\mathbf{v}_i = (\mathbf{w}_i, S(\mathbf{w}_i))$.
4) Concatenate $\mathbf{v}_i$ to get vector $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m)$.
5) Verify that $\mathbf{q} = \mathbf{v}(\mathbf{I}|\mathbf{Q})^T$. If not, signature is invalid.

IV. EXAMPLE

In this section, we provide (simplified) examples for some of the critical steps of the algorithm. In our example, we will work with SPN introduced in Section II-B (see Figure 1). To demonstrate MRHS equation building step, imagine first a simple SPN with 2-bit "S-boxes" given by permutation 1230. Suppose we denote S-box inputs by x_1, x_2, and outputs by y_1, y_2. MRHS equation with solutions corresponding to valid I/O pairs for S-box can be written as

$$(x_1, x_2, y_1, y_2) \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \in \left\{ \begin{array}{l} (0,0,0,1), \\ (0,1,1,0), \\ (1,0,1,1), \\ (1,1,0,0) \end{array} \right\}$$

If we wanted to demonstrate an MRHS equation related to a 4-bit S-box, we would need four input variables, four output variables, 8×8 identity matrix, and a set of sixteen 8-bit vectors on the right-hand side.

Now let us construct an MRHS system for Figure 1. We have selected that our unknowns are S-box inputs. We do not use variables of type $y_1, y_2, \ldots$ as in the previous example. Instead, we express y variables as linear combinations of x variables and subkey bits. We use SPN network as denoted in Figure 1. Its linear layer is represented as a multiplication with matrix $\mathbf{L}$. We can write

$$(\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8) = (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4) \cdot \mathbf{L} + (\mathbf{k}_5, \mathbf{k}_6, \mathbf{k}_7, \mathbf{k}_8),$$

which leads to

$$(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4) = (\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8) \cdot \mathbf{L}^{-1} + (\mathbf{k}_5, \mathbf{k}_6, \mathbf{k}_7, \mathbf{k}_8) \cdot \mathbf{L}^{-1}$$

We can extract equations for separate y_i variables (one for each S-box) by splitting matrix $\mathbf{L}$ into blocks $\mathbf{L}_{i,j}$ of size 4×4 bits ($k \times k$ in general). We get the following system (simplified for the first two S-boxes only):

$$(\mathbf{x}_1, \ldots, \mathbf{x_8}, 1) \cdot \left(\begin{array}{cc|cc|c} \mathbf{I} & 0 & 0 & 0 & \cdots \\ 0 & 0 & \mathbf{I} & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ 0 & \mathbf{L}_{11}^{-1} & 0 & \mathbf{L}_{12}^{-1} & \cdots \\ 0 & \mathbf{L}_{21}^{-1} & 0 & \mathbf{L}_{22}^{-1} & \cdots \\ 0 & \mathbf{L}_{31}^{-1} & 0 & \mathbf{L}_{32}^{-1} & \cdots \\ 0 & \mathbf{L}_{41}^{-1} & 0 & \mathbf{L}_{42}^{-1} & \cdots \\ 0 & \mathbf{c}_1 & 0 & \mathbf{c}_2 & \cdots \end{array} \right) \in \mathbb{R} \times \mathbb{R},$$

where $\mathbb{R}$ is a set of sixteen 8-bit vectors $\mathbb{R} = \{(\mathbf{x}, S(\mathbf{x})), \mathbf{x} \in \mathbb{F}_2^4\}$. Constants are computed from subkeys:

$$(\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4) = (\mathbf{k}_5, \mathbf{k}_6, \mathbf{k}_7, \mathbf{k}_8) \cdot \mathbf{L}^{-1},$$

and can be extracted from MRHS equation matrix to get an extended form $\mathbf{x} \cdot \mathbf{M} + \mathbf{c} \in \mathbb{R}$, as described in Section II-C.

The system is expanded for each round in a similar way. However, for the last round, we need to add variables that denote S-box outputs, as there are no other S-box inputs. In four round SPN, we get a system in form

$$(\mathbf{x}_1, \ldots, \mathbf{x}_{20}) \cdot \begin{pmatrix} \mathbf{A}_1 & 0 & 0 & 0 \\ \mathbf{A}_2 & \mathbf{A}_3 & 0 & 0 \\ 0 & \mathbf{A}_4 & \mathbf{A}_5 & 0 \\ 0 & 0 & \mathbf{A}_6 & \mathbf{A}_7 \\ 0 & 0 & 0 & \mathbf{A}_8 \end{pmatrix} + \mathbf{c} \in \mathbb{R} \times \cdots \times \mathbb{R},$$

where each matrix $\mathbf{A}_i$ is 16×16 bit matrix, each $\mathbf{x}_i$ is an unknown 4-bit vector ($\mathbf{x}_1$ to $\mathbf{x}_{16}$ are S-box inputs, $\mathbf{x}_{17}$ to $\mathbf{x}_{20}$ are last-round S-box outputs). Constant c is an 80-bit vector (computed from subkeys), and there are 16 (identical) right-hand side sets $\mathbb{R}$ in the Cartesian product.

The system matrix and the constant $\mathbf{c}$ can be rewritten into 16 blocks of 8 bits, corresponding to each S-box used during the encryption. I.e., we understand the system to be

$$\mathbf{x} \cdot (\mathbf{M}_1 | \cdots | \mathbf{M}_{16}) + (\mathbf{c}_1, \ldots, \mathbf{c}_{16}) \in \mathbb{R} \times \cdots \times \mathbb{R}.$$

Select a secret 16-element permutation with first 4 elements fixed, e.g. $\pi = (1, 2, 3, 4, 12, \cdots, 7)$. To get the public key, we construct

$$\mathbf{M}_\pi = (\mathbf{M}_1 | \cdots | \mathbf{M}_4 | \mathbf{M}_{12} | \cdots | \mathbf{M}_7),$$

and compute the corresponding systematic parity check matrix $\mathbf{H}$. Size of the matrix $\mathbf{M}_\pi$ is 80×128 bits, thus $\mathbf{H}$ has dimensions 48×128. Finally, we compute 48-bit public vector

$$\mathbf{q} = (\mathbf{c}_1, \ldots, \mathbf{c}_4, \mathbf{c}_{12}, \ldots, \mathbf{c}_7) \cdot \mathbf{H}^T.$$

Having constructed private and public keys for the signature scheme, following the steps of the signature and verification algorithms is relatively simple. We do not provide concrete bit values, as the number of bits involved is quite large even for the SPN-based demo. To provide a working demonstration of the system, we have prepared a proof of concept implementation based on simple SPN with 16-bit block and 4-bit S-boxes. The source code is available on GitHub at https://github.com/zajacpa/SPNsig. The demonstration code requires SAGE linear algebra system [10] to run.

Note that for a simpler implementation we store inputs and outputs of S-boxes in different order: first we store sequence of all S-box inputs, and then the sequence of S-box outputs. While not exactly corresponding to the theoretical MRHS instance, it is easy to see that the algorithm still works regardless of the bit order. We only need to ensure that the corresponding columns of $\mathbf{M}$ (with correct $\mathbf{H}$) and $\mathbf{c}$ are arranged in the same way as the desired order of the bits in the transcript of the encryption.

V. SIGNATURE SCHEME PROPERTIES

After the description of the signature scheme, we devote this short section to the discussion of the correctness and efficiency of the proposed scheme.

A. Correctness

Verification algorithm has two steps where it can reject the signature: verification of the hash $\mathbf{h}$, and verification of the syndrome of $\mathbf{v}$.

During the signature, inputs to SPN encryption were chosen in such a way that $\mathbf{h}$ is the input to the first layer of S-boxes in the first round of SPN. Recall that during the key generation we require that the order of these S-boxes remains fixed by permutation π. This means the verifier in the second step just checks whether plaintext $\mathbf{p}$ was constructed from message $\mathbf{m}$ and provided randomness $\mathbf{r}$.

Signature process is based on SPN encryption modelled by MRHS system (1). The pairs $(\mathbf{u}_i, S(\mathbf{u}_i))$ are corresponding right-hand sides providing a valid solution of this system. They are published in modified order, which is obtained as vector $\mathbf{v}$ by the verifier. The following (overdetermined) linear equation system thus has a solution:

$$\mathbf{x}\mathbf{M}_\pi + \mathbf{c}_\pi = \mathbf{v}.$$

If we multiply this by $\mathbf{H}^T = (\mathbf{I}|\mathbf{Q})^T$ we get:

$$\mathbf{x}\mathbf{M}_\pi\mathbf{H}^T + \mathbf{c}_\pi\mathbf{H}^T = \mathbf{v}\mathbf{H}^T,$$

or equivalently

$$\mathbf{q} = \mathbf{v}(\mathbf{I}|\mathbf{Q})^T.$$

B. Efficiency

The performance of the scheme depends on the chosen underlying SPN. However, in comparison with number theoretic or other post-quantum signature schemes the algorithm is extremely simple. In the following we consider the part of the algorithm without message hashing (which is required regardless of the signature algorithm).

The signature algorithm is equivalent to a single symmetric encryption accompanied with a simple permutation of a vector of transcript bits. Similarly, the verification algorithm requires evaluation of S-boxes plus vector-matrix multiplication. The number of nonlinear operations is the same as the number of operations required during a single SPN encryption. While the complexity of the linear part can be higher than required in SPN encryption, this operation is quite simple and can be efficiently implemented.

The size of the public key and signature is related to the chosen SPN. Private key consists of an l-bit symmetric key and a permutation of m numbers. Public key is a pair consisting of a vector $\mathbf{q}$ of size $2mk - (r+1)n$ bits and a matrix $\mathbf{Q}$ of size $(2mk - (r+1)n) \times mk$ bits. Each signature is also a sequence of mk bits.

Let us consider instantiation of the scheme with AES algorithm. In this case, we use $m = 160$ S-boxes of size $k = 8$ bits. This means that each signature is 1280 bits long (160 bytes). This is comparable to a standard RSA signature. Public key contains vector $\mathbf{q}$ of size 1280 bits, and matrix $\mathbf{Q}$ of size 1638400 bits. Together, this is approximately 200kB.

VI. SECURITY

As mentioned in the introduction, the proposed signature scheme is a provided as a new (and hopefully interesting) concept. Security of the scheme is related to the difficulty of solving MRHS equations and the decoding problem, but also to the security of the underlying symmetric encryption scheme. To be able to fully instantiate the (modified version of the) scheme in a provably secure manner requires a deeper research of the proposed scheme, and the related security questions. In this section we focus on security aspects of the scheme and the potential attacks that can compromise the security of the scheme.

Informally, a signature system is secure, if no (poly-time) attacker is able to forge signatures on new messages. To formally prove the security we would have to provide suitable security reduction to some computationally difficult problem. While we believe this might be possible with a suitable instantiation of the SPN and further tweaks of the design, the current scheme as described is not (provably) secure.

The *security level* of the scheme is limited to $n/2$, where n is the block size of the used SPN. This is due to the use of initial hashing step: If the attacker can find a hash collision in the form $h(r|m_1) = h(r|m_2)$, he can use the same signature for two different messages, breaking the non-repudiation property of the signatures. Thus, for our scheme instantiated with AES we can guarantee at most 64-bit security. It is possible to use general Rijndael algorithm with 256-bit block, and with 256-bit hash function (e.g., SHA-2) to extend the presumed security level to 128-bits. In this case $m = 448$ (Rijndael with 256-bit block and 128-bit key, 14 rounds, 32 S-boxes in each), the signature size is 3584 bits, and public key size is 1568 kB. If we take into account Grover's algorithm and extend the key to 256-bits as well, we get $m = 576$ (18 rounds), i.e., 576B signature, and 2592kB public key, respectively.

Regardless of generic attacks on the hash function, there are multiple other ways that attacker can try to attack the signature algorithm:

1) Try to submit false vector $\mathbf{w}'$ that will satisfy the verification algorithm. First part of the verification (hash of message with nonce r should be the prefix of $\mathbf{w}'$) is easily satisfied. Thus the attacker is only concerned with providing false $\mathbf{w}'$ that will satisfy $\mathbf{q} = \mathbf{v}(\mathbf{I}|\mathbf{Q})^T$.
2) Try to derive the private key from the public key.
3) Try to derive the private key, or a new signature from the public key and (chosen) signatures.

To prevent these attacks we rely on the following (presumably) difficult problems:

1) Decoding problem/MRHS problem. These problems are in general NP-hard, and both are related as discussed in [11]. Parameters of the proposed scheme (if instantiated by AES/Rijndael) are comparable (or even stronger) to parameters of proposed code-based cryptosystems with the same expected security level. Main concern for our scheme is whether our specific type of decoding/MRHS instances derived from SPN representation by MRHS system are still sufficiently hard (in relation to random instances of the problem).
2) Given a set of permuted transcripts of the SPN encryption (with known, but not chosen, input to the first layer of S-boxes), can the attacker find the key, or at least provide any new transcript (permuted in the same way)? This question is related to the security of the underlying SPN, and the efficiency of the proposed masking (random permutation of S-box inputs).

We will now discuss these attack vectors in more details.

A. Decoding/MRHS attacks

Verification algorithm consists of verifying the identity $\mathbf{v} \cdot \mathbf{H}^T = \mathbf{q}$. There is an exponentially large number of solutions $\mathbf{v}'$, but only some of them represent a valid signature. Given valid signature $\mathbf{v}$, such that $\mathbf{v}\mathbf{H}^T = \mathbf{q}$, attacker can compute any other valid $\mathbf{v}'$ by adding a codeword $\mathbf{u}$ of the code generated by $\mathbf{M}_\pi$ (which can be computed from public $\mathbf{Q}$). If our signature system was just based on the Niederreiter-like code-based system [12], attacker could just use any prefix $h(\mathbf{m}', \mathbf{r}')$ and find a valid $\mathbf{v}' \cdot \mathbf{H}^T = \mathbf{q}$ by solving a linear system of equations.

However, our scheme has an additional property: only one half of the vector $\mathbf{v}$ is provided in the signature, second half is computed using SPN's non-linear S-boxes. This means that valid signatures form only a (very small and non-linear) subset of the code coset. Each valid signature is a solution of the MRHS system given by equation 2. If the attacker can forge signatures, he can solve this non-linear MRHS system, and vice-versa.

Note that underlying SPN is a block cipher, and thus a permutation for each secret key. This means that there is a unique transcript of the encryption and thus a unique signature for each message hash.

As mentioned, the MRHS problem problem is related to the decoding problem. We can use a specific decoding algorithm to solve MRHS problem [11]. In our system there is also a specific case where the attacker can apply the decoding algorithm: given public key $(\mathbf{q}, \mathbf{Q})$, attacker tries to find the original constant $\mathbf{c}_\pi$ which was used to define code coset defined by syndrome $\mathbf{q}$. If the attacker obtains $\mathbf{c}_\pi$ it might be possible to reconstruct the original key: Attacker knowns a permuted set of subkeys (except the first and the last one), how difficult it is obtain original key?

However, to get $\mathbf{c}$ attacker needs to solve the decoding problem first. Depending on the structure of the cipher he can presume that $\mathbf{c}_\pi$ is a sparse vector (in contrast to signatures $\mathbf{v}$) with a specific structure (i.e., subkeys are only used to compute outputs of S-boxes from the S-box inputs of the next round). It is not clear whether this information can be sufficient to simplify the decoding problem, as the expected weight of $\mathbf{c}$ is still too large (compared to expected minimum code weight).

B. Structural attacks on signatures

We believe that the security of the signature scheme with respect to decoding/MRHS problem is sufficient with respect to generic solving methods. However, there are more critical attacks that exploit the internal structure of the used MRHS problem to circumvent the need to apply generic solution methods, or to assist the generic methods. We will call these attacks structural attacks (similar to terminology used in code-based crypto).

First, let us note that it is necessary that attacker cannot recover permutation π. If the attacker knowns π and a single signature $\mathbf{w}$, he can reconstruct the exact sequence of S-box inputs in SPN. As S-boxes are public, he also knows the corresponding S-box outputs. Let $\mathbf{x}$ denote a vector of outputs of S-boxes in round $i-1$, and let $\mathbf{y}$ denote a vector of inputs of S-boxes in round i. Let $\mathbf{L}$ represent a (known) matrix of the linear diffusion layer of SPN. Then $\mathbf{k}^i = \mathbf{y} + \mathbf{L}(\mathbf{x})$. This means that knowledge of π leads to a knowledge of the sequence of subkeys. This sequence is sufficient to forge signatures (regardless of the key schedule).

Note that specific key schedules can have an adverse effect on the security. In AES (and many other ciphers), main encryption key can be derived from any single subkey (by running the key schedule algorithm in reverse). This means that the attacker only needs to find matching S-box outputs/inputs between arbitrary rounds $i-1$ and i. The easiest case is to find the correct inputs to S-boxes in round 2 (as round 1 is known, due to fixed part of π). This means that attacker needs to correctly place a sequence of $b = n/k$ blocks out of $(r-1)b$ blocks. Complexity of exhaustive search in this case is

$$N = \frac{((r-1)\,b)!}{((r-2)\,b)!}.$$

In case of AES-128, we get $N \approx 2^{113}$, which is much higher than expected security of 64 bits. Similarly for Rijndael-256-256, we get $N \approx 2^{289}$, which is again much higher than security level of 128 bits.

On the other hand, attacker is not limited to an exhaustive search. In the first version of the signature scheme, message $\mathbf{m}$ (of fixed length given by the block size) was used directly

as an input of the first layer of S-boxes. Suppose that SPN was AES-128, and attacker used some signature oracle to obtain signatures for two messages that differ in a single byte. From the properties of AES diffusion layer we know that in second round there are exactly 4 non-zero differences and 12 zero differences. Attacker marks those bytes that are unchanged between signatures. Repeating this with different bytes, he can disclose the positions of the round-2 S-box inputs (in 4 byte groups, whose order can be quickly searched to completly disclose the subkey).

To prevent the class of chosen plaintext attacks on SPN, we have added a randomized hashing[3] as the first step of the algorithm. The attacker still knows the inputs to the first layer of S-boxes, but cannot select them in arbitrary way. E.g., if the attacker was to reproduce the previous attack, he requires two hashes that coincide in each byte except one. This is slightly easier than the security level (in AES-128 expected complexity is 2^{60} hashes), but it is then followed by an attack with a difficulty higher than the reduction obtained while looking for collisions.

While the randomized hashing restricts the efficiency of some of the structural attacks, it is not clear whether it is sufficient to prevent all attacks of this type (and how serious is the security level reduction). Efficiency of these types of attacks are related to the concrete structure of the used SPN. It is not sufficient that the cipher under consideration has a sufficient number of rounds to resist some type of cryptanalysis: as we have seen the goal of the attacker can be different, as in some cases he only needs to distinguish which blocks are used in some specific round.

C. Structural attacks on public key

In previous section we have discussed some ways that the attacker can try to recover secret permutation π from some known or chosen signatures. There is still another attack vector related to the public key, namely parity check matrix $\mathbf{H} = (\mathbf{I}|\mathbf{Q})$.

System matrix $\mathbf{M}$ reflects the structure of the used SPN, and is very sparse. Essentially, each block $\mathbf{M}_i$ can be expressed as

$$\begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{L} \\ \mathbf{0} & \mathbf{0} \end{pmatrix},$$

where $\mathbf{I}$ is an identity matrix corresponding to S-box inputs in round i. Matrix $\mathbf{L}$, along with constant $\mathbf{c}$ reflects the linear layer of SPN, and represent the affine transformation applied to S-box inputs in round $i+1$ to compute S-box outputs in round i.

This structure of the matrix is in essence preserved when π is applied. It is however not clear, whether this structure is always present in matrix $\mathbf{H}$. Note that there are multiple possible matrices $\mathbf{H}$: any base of the dual space is suitable for signature verification. For efficiency reasons, we have chosen to restrict $\mathbf{H}$ to have a systematic form (so we only need to store $\mathbf{Q}$). Another advantage of the systematic form is that during the computation of systematic $\mathbf{H}$ we hope to hide the sparsity and structure of the original $\mathbf{M}$, similarly to LDPC- and MDPC-based cryptosystems [13].

[3]We thank one of the anonymous reviewers of the Central European Conference on Cryptology 2019 for the general idea.

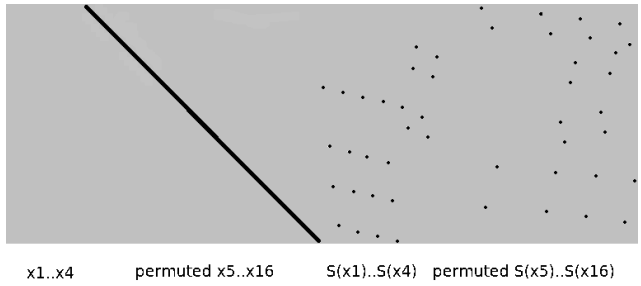

Fig. 4. Graphical depiction of the systematic parity check matrix obtained in one of the experiments with small 16-bit SPN.

Our experiments with the small SPN show that in this case a computation of the systematic parity check matrix is not enough to mask the structure of the system. In Figure 4, we show an example of the matrix $\mathbf{H}$ obtained from the SPN demo. Structure of the original $\mathbf{M}_\pi$ is clearly preserved, and by matching ones on corresponding lines attacker can obtain permutation π.

We have not implemented a full version of AES-based scheme, thus it is not clear whether this problem is also affecting this larger scheme. While AES-based matrix is less sparse due to MixColumns operations, it is still significantly structured. We believe that in this case, systematic form of $\mathbf{H}$ would still not be enough to hide the structure sufficiently.

We might ask, whether the signature scheme be ever made secure in view of this structural attack? Unfortunately, we do not have a definitive answer, but we propose some possible solutions to hide the system structure:

- Change the basis of variables in the system. Instead of defining $\mathbf{x}$ as a vector of S-box inputs, define $\mathbf{x}$ as a vector of arbitrary linearly independent variables $\mathbf{y}$, from which S-box inputs can be computed with linear algebra as $\mathbf{x} = \mathbf{y}\mathbf{A}$. In the key generation, we get modified system $\mathbf{y}\mathbf{A}\mathbf{M}$, and we compute parity check matrix for the code generated by $\mathbf{A}\mathbf{M}_\pi$. Note that this is actually the same code as the one generated by $\mathbf{M}_\pi$. Our hope is to obtain different matrix $\mathbf{H}$ that is less sparse and structured as before. It is not clear whether this matrix can really be obtained, and even if it is obtained, whether attacker cannot find different $\mathbf{H}$ that will show him the system structure.
- Use SPN with secret random linear layer. We are inspired by cipher family LowMC [6], but in the signature scheme we do not publish the linear layer, but make it part of the secret key. In this case we lose some efficiency and flexibility in symmetric encryption: linear layer is now asymmetric secret, thus the cipher cannot be reused for symmetric encryption between network participants. This can be remedied by using two versions of the cipher: one for encryption, with fixed (simple) linear layer, and one

(complex/random) linear layer for signatures with secret linear layer. Note that while this eliminates some of the sparsity from the system, the block structure related to cipher rounds remains the same.

- Decompose larger S-boxes to individual AND gates, and call these AND gates new S-boxes. In this case we create a MRHS system with right-hand sides consisting of sets related to AND gates, consisting of triples: $\{000, 010, 100, 111\}$. Signer produces a vector of bits of length $2m$, and verifier just appends bits, that are products of successive pairs, and verifies the coset. System matrix in this case is more complex, as individual I/O bits on individual AND gates need to be expressed as linear combinations of variables. This is also the main disadvantage of the system: system matrix becomes excessively large, increasing the size of the public key.
- Use a different cipher design. E.g., a wide cipher with a large branch number could have a complex enough linear layer in each round. If we study Figure 4, we can see that the main problem is that S-box inputs from one round are only connected to S-box outputs from the previous rounds. If we do not need the encryption algorithm to be reversible (e.g. for a use in stream cipher mode), we can propagate some internal bits to multiple rounds. Note that this must be done carefully to avoid enabling linear or differential attacks on the underlying cipher.
- Would it be possible to hide some information? E.g., we might consider, what would happen if we only include odd numbered rounds in the signature. The MRHS system is still present, and the signature system works correctly. Main difference is that here are now fewer blocks of $\mathbf{M}$, and parity check matrix has smaller dimension (if we remove half of blocks, dimension becomes zero!). This means there would be false solutions, and multiple signatures per message hash. It is not clear whether there is a suitable trade-off when removing selected blocks would actually improve the security.

Implementing some of these solutions can also solve problems with structural attacks based on known signatures. However, we believe that provably secure scheme can only be obtained with a carefully designed block cipher with a goal of providing signatures (through our general scheme) along with symmetric encryption.

VII. Conclusions

In this paper we have presented a new concept of signature scheme based on symmetric cipher design, whose signature and verification algorithm are comparable in complexity to symmetric encryption. Parameters of the system, and its connection to symmetric ciphers, are quite favourable to consider it for future use.

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. Moreover, if the signature system, as presented here, is instantiated by current cipher designs (such as AES), it would presumably not attain the required security due to structural attacks. We have proposed some options on how to further hide the inner structure of the encryption system, but all of these options require further research. We believe that the most promising direction is to design a specific symmetric cipher that will support solid security arguments for the proposed scheme.

Acknowledgment

The authors would like to thank anonymous reviewers of the Central European Conference on Cryptology for the comments on the first proposal of this scheme, that led to identification of some types of attacks on the scheme.

References

[1] A. Hulsing, D. J. Bernstein et al., "Sphincs+," 2018. [Online]. Available: https://sphincs.org/

[2] G. Zaverucha, M. Chase *et al*., "The Picnic signature algorithm," 2018. [Online]. Available: https://microsoft.github.io/Picnic/

[3] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot, "White-box cryptography and an aes implementation," in *International Workshop on Selected Areas in Cryptography*. Springer, 2002, pp. 250–270. **DOI:** 10.1007/3-540-36492-7_17

[4] J. Ding and B.-Y. Yang, "Multivariate public key cryptography," in *Postquantum cryptography*. Springer, 2009, pp. 193–241. **DOI:** 10.1007/978-3-540-88702-7_6

[5] H. Raddum and I. Semaev, "Solving multiple right hand sides linear equations," *Design, Codes and Cryptography*, vol. 49, no. 1, pp. 147–160, 2008. **DOI:** 10.1007/s10623-008-9180-z

[6] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner, "Ciphers for mpc and fhe," in *Advances in Cryptology–EUROCRYPT 2015*. Springer, 2015, pp. 430–454. **DOI:** 10.1007/978-3-662-46800-5_17

[7] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[8] H. Raddum and P. Zajac, "MRHS solver based on linear algebra and exhaustive search," *Journal of Mathematical Cryptology,* vol. 12, no. 3, pp. 143–157, 2018. **DOI:** 10.1515/jmc-2017-0005

[9] P. Zajac, "MRHS equation systems that can be solved in polynomial time," *Tatra Mountains Mathematical Publications*, vol. 67, no. 1, pp. 205–219, 2016. **DOI:** 10.1515/tmmp-2016-0040

[10] The Sage Developers, *SageMath, the Sage Mathematics Software System (Version 7.2)*, 2016, http://www.sagemath.org.

[11] P. Zajac, "Connecting the complexity of MQ-and code-based cryptosystems," *Tatra Mountains Mathematical Publications*, vol. 70, no. 1, pp. 163–177, 2017. **DOI:** 10.1515/tmmp-2017-0025

[12] N. T. Courtois, M. Finiasz, and N. Sendrier, "How to achieve a McEliece-based digital signature scheme," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2001, pp. 157–174. **DOI:** 10.1007/3-540-45682-1_10

[13] M. Baldi, *QC-LDPC code-based cryptography*. Springer Science & Business, 2014.

Pavol Zajac received PhD. in applied mathematics from Slovak University of Technology in Bratislava in 2008. His research interest is mathematical cryptography, see https://scholar.google.com/citations?user=kutD0ZsAAAAJ for a list of publications. Currently he is a full professor at Slovak University of technology working on problems related to post-quantum cryptography.

Peter Špaček is a PhD. student under supervision of Pavol Zajac. His research focus is on post-quantum cryptography and its adaptation into real world applications.