Bash-f: Another LRX Sponge Function

S. Agievich, V. Marchuk, A. Maslau, V. Semenov

Research Institute for Applied Problems of Mathematics and Informatics
Belarusian State University

Abstract. We present the Bash family of hashing algorithms based on the sponge paradigm. A core element of this family is the Bash-f sponge function which refers to the LRX (Logical-Rotation-Xor) class of symmetric cryptography schemes. We describe the components of Bash-f: a nonlinear mapping, linear diffusion mappings, a permutation of words of a hash state. For each component, we establish reasonable quality criteria as detailed as possible to make the choice of the component maximally objective and transparent.

Keywords: hash algorithm, sponge construction, LRX, S-box, bitslice technique.

1 Introduction

Bash = Bhash is a family of hashing algorithms that are being standardized in Belarus. The forthcoming specification STB 34.101.77 will continue the series of cryptography standards of our country [5].

The Bash algorithms are based on the sponge construction introduced by G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche in [3] and then studied in details in [2]. The base component of Bash is a step (or sponge) function named Bash-f. This function updates a hash state using a current block of input data. It uses bitwise logical operations NOT ($\neg$), OR ($\vee$), AND ($\wedge$) on 64-bit words, rotations and XORs ($\oplus$) of these words. Therefore, Bash-f refers to the LRX (Logical-Rotation-Xor) class of symmetric cryptographic schemes.

The chosen platform "sponge + LRX" was first used in the Keccak (SHA-3) family of hashing algorithms. Considering Keccak is smart, effective and sound, we nevertheless decided to design our own hash family. By our estimates, Bash is competitive to Keccak in performance and security guarantees expressed in terms of the number of active S-boxes.

The Bash algorithms differ by a security level $l \in \{16, 32, 48, \dots, 256\}$. The algorithm of level l processes data by blocks of $1536 - 4l$ bits and returns a hash value of length $2l$.

In Bash the sponge scheme is instantiated in the following manner:

1) the hash state has length 1536;

2) the initial state contains the encoding of l;

3) data blocks are processed in the so-called overwrite mode (instead of the usual xor mode as in Keccak);

4) the sponge-compliant padding rule $\{0,1\}^* \ni X \mapsto X \parallel 010^t$ is used (t is a minimal non-negative integer such that $|X| + t + 2$ is a multiple of $1536 - 4l$).

These settings conform with [2]. Here, as usual, 0^t is the word of t zeros, $\|$ denotes concatenation, and $|X|$ is the length of X.

To complete the instantiation of the sponge scheme, it is sufficient to define `Bash-f`, that is, a bijection over $\{0,1\}^{1536}$. We will specify and discuss it later in this paper. Section 2 outlines `Bash-f` in general, Sections 3 — 5 describe its components in details. We are planning to present security and performance estimates of `Bash-f` in a continuation of this paper.

We choose the components of our sponge function according to a principle, which is often called rigidity. This principle can be formulated as follows: With all the richness of choice there is no alternative. We subsequently exclude various configurations of the target components using reasonable quality criteria. As a result we keep only a few equivalent configurations and finally pick the first one of them. Sections 3 — 5 are organized in the same way: we specify quality criteria and describe the configurations that satisfy them.

2 The `Bash-f` sponge function

A hash state $S \in \{0,1\}^{1536}$ is divided into 24 words of length 64:

$$S = S_0 \,\|\, S_1 \,\|\, \ldots \,\|\, S_{23}.$$

The words are arranged into the 3×8 matrix (see Fig. 1). The columns (S_v, S_{v+8}, S_{v+16}), $v = 0, 1, \ldots, 7$, of this matrix are called *vertical planes*, and the rows $(S_{8i}, S_{8i+1}, \ldots, S_{8i+7})$, $i = 0, 1, 2$, are called *horizontal*. Each vertical plane includes 64 *vertical bit triples*.

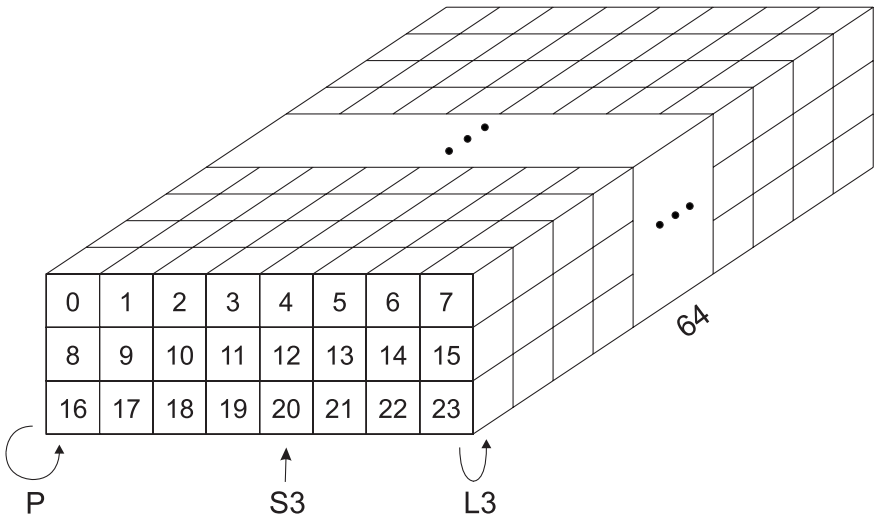

Figure 1: A hash state

In `Bash` the following conventions concerning 64-bit words are used. The lowest octet of a word is the first one (the little-endian rules), but octets have, as usual, the highest bit first. Before the shift, a word is loaded into a hypothetical register. The first (last) octet of the word is loaded into the lowest (highest) octet of the register. After the shift, octets of the register are unloaded in the same order. The cyclic shift towards high bits by d positions is denoted by $\texttt{RotHi}^d$. For example, $\texttt{RotHi}^1(\texttt{0123456789ABCDEF}_{16}) = \texttt{03468ACE12579BDF}_{16}$. The ordinary (without rotation) shift towards low bits by 1 position is denoted by `ShLo`.

The `Bash-f` function consists of 24 rounds, each of them is a composition of basic transformations described below.

$\underline{L3}$. Vertical planes are transformed by linear mappings $L3$. They utilize the operations $\oplus$ and $\texttt{RotHi}^d$. Cyclic shifts, which provide diffusion of bits in words, are used only here. There are 4 shifts in total. A tuple $[m_1, n_1, m_2, n_2]$ of their values refines $L3$ where necessary.

A triple $(W_0, W_1, W_2) = L3[m_1, n_1, m_2, n_2](w_0, w_1, w_2)$ is calculated by the following algorithm:

$$\begin{aligned}
&W_0 \leftarrow w_0 \oplus w_1 \oplus w_2,\\
&W_1 \leftarrow w_1 \oplus \texttt{RotHi}^{m_1}(w_0) \oplus \texttt{RotHi}^{n_1}(W_0),\\
&W_2 \leftarrow w_2 \oplus \texttt{RotHi}^{m_2}(w_2) \oplus \texttt{RotHi}^{n_2}\left(w_1 \oplus \texttt{RotHi}^{n_1}(W_0)\right).
\end{aligned}$$

It takes 6 additions, provided that a preimage of $\texttt{RotHi}^{n_2}$ is determined during the calculation of W_1.

$\underline{S3}$. All vertical bit triples are tied together by the same S-box $s\colon \{0,1\}^3 \to \{0,1\}^3$. The mapping $S3$ applies s to bit triples of a vertical plane (W_0, W_1, W_2). $S3$ uses the operations $\neg$, $\wedge$, $\vee$ and $\oplus$ (in decreasing order of precedence). It is the only nonlinear mapping in `Bash-f`.

The target plane is transformed as follows:

$$(W_0, W_1, W_2) \leftarrow (W_0 \oplus W_1 \vee \neg W_2, W_1 \oplus W_0 \vee W_2, W_2 \oplus W_0 \wedge W_1).$$

$\underline{P}$. The state words are shuffled according to a permutation P: $S_{P(u)}$ replaces S_u. P rotates the horizontal planes and simultaneously permutes the words within the planes:

$$P(u) = \begin{cases} \pi_0(u) + 8, & 0 \le u < 8,\\ \pi_1(u-8) + 16, & 8 \le u < 16,\\ \pi_2(u-16), & 16 \le u < 24. \end{cases}$$

Here π_i is the permutation of the i-th plane:

$$\begin{aligned}
\pi_0(v) &= (v + 2(v \bmod 2) + 7) \bmod 8,\\
\pi_1(v) &= v + 1 - 2(v \bmod 2),\\
\pi_2(v) &= (5v + 6) \bmod 8.
\end{aligned}$$

$\underline{\textit{Round constants}}$. In each round a constant $C \in \{0,1\}^{64}$ is added to the word S_{23}. The constants are not repeated and all the rounds become different.

The constants are built as consecutive states of a linear feedback shift register with a primitive characteristic polynomial. The initial state of the register is composed of the first 8 octets of the S-box of the `Belt` block cipher [5]. The coefficients of the characteristic polynomial are also determined by 8 octets (the first appropriate) of this S-box.

Overall, `Bash-f` acts as follows.

Algorithm `Bash-f`

Input: $S = S_0 \parallel S_1 \parallel \ldots \parallel S_{23}$, $S_i \in \{0,1\}^{64}$.

Output: the updated state S.

Steps:

1 $C \leftarrow \texttt{B194BAC80A08F53B}_{16}$.

2 For $i = 1, 2, \ldots, 24$:

2.1 $[m_1, n_1, m_2, n_2] \leftarrow [8, 53, 14, 1]$;

2.2 for $v = 0, 1, \ldots, 7$:

(a) $(S_v, S_{v+8}, S_{v+16}) \leftarrow L3[m_1, n_1, m_2, n_2](S_v, S_{v+8}, S_{v+16})$;

(b) $(S_v, S_{v+8}, S_{v+16}) \leftarrow S3(S_v, S_{v+8}, S_{v+16})$;

(c) $[m_1, n_1, m_2, n_2] \leftarrow 7[m_1, n_1, m_2, n_2] \bmod 64$;

2.3 $S \leftarrow S_{P(0)} \parallel S_{P(1)} \parallel \ldots \parallel S_{P(23)}$;

2.4 $S_{23} \leftarrow S_{23} \oplus C$;

2.5 if the lowest bit of C is zero, then $C \leftarrow \mathtt{ShLo}(C)$;
else $C \leftarrow \mathtt{ShLo}(C) \oplus \mathtt{AED8E07F99E12BDC}_{16}$.

3 Return S.

The reference implementation of `Bash-f` and `Bash` in whole is available in [1].

3 The $S3$ nonlinear mapping

In $S3$ the bitslice technique is used: bits of $n = 3$ words are processed simultaneously by the same S-box s, this block is described by a simple boolean circuit which is extended naturally from bits to words.

In most known cases the bitslice technique is used with $n = 4$. These cases cover the block ciphers `Serpent` (where the technique was actually introduced), `Noekeon` and `Rectangle`, the hash functions `JH`, `Luffa` and others.

The complexity of a circuit that provides an acceptable cryptographic quality of the underlying S-box increases quickly with n. Even for $n = 5$, one should weaken s to simplify the circuit. Such a weakening has been used in `Keccak`. We have chosen another approach: use the strongest S-box of the smallest suitable dimension n. Since for $n = 2$ all bijective S-boxes are affine (and therefore not suitable), we have selected $n = 3$.

The target S-box s maps $x = x_0x_1x_2 \in \{0,1\}^3$ to $y_0y_1y_2 \in \{0,1\}^3$. The domain of s can naturally be transformed into $\mathbb{F}_2^3$ or $\mathbb{Z}_8 = \{0, 1, \ldots, 7\}$. In the latter case a binary word $w_0w_1w_2$ is interpreted as the number $w_0 + 2w_1 + 4w_2$. We specify s by the word $s(0)s(1)\ldots s(7) \in \mathbb{Z}_8^8$ of such numbers.

We can also specify s by its coordinate algebraic normal forms, that is, the polynomials $s_i \in \mathbb{F}_2[x_0, x_1, x_2]/(x_0^2 - x_0, x_1^2 - x_1, x_2^2 - x_2)$ such that $y_i = s_i(x_0, x_1, x_2)$, $i = 0, 1, 2$.

Finally, the action of s can be transferred to $\mathbb{F}_8$ and set up by the permutation polynomial $p(z) \in \mathbb{F}_8[z]$. Due to Hermite's criterion (see [8]), its degree ≤ 6. We represent $\mathbb{F}_8$ as the quotient ring $\mathbb{F}_2[\xi]/(\xi^3 + \xi + 1)$ or $\mathbb{F}_2[\xi]/(\xi^3 + \xi^2 + 1)$ and get two polynomials $p(z)$. In both cases words $w_0w_1w_2$ correspond to the elements $w_0 + w_1\xi + w_2\xi^2$ of the chosen ring.

In Table 1 all the possible forms of the finally selected S-box are listed. The 3rd row of the table corresponds to the representation $\mathbb{F}_8 = \mathbb{F}_2[\xi]/(\xi^3 + \xi + 1)$ and the last one to $\mathbb{F}_8 = \mathbb{F}_2[\xi]/(\xi^3 + \xi^2 + 1)$.

The optimal S-boxes. There are $8! = 40320$ appropriate S-boxes. We call a box s optimal if it satisfies the standard cryptographic criteria **R1** — **R3**.

R1. The equation $s(x \oplus \alpha) \oplus s(x) = \beta$ has no more than 2 (the minimum value) solutions for any nonzero $\alpha, \beta \in \{0,1\}^3$.

R2. The nonlinearity of s is equal to 2 (the maximum value).

R3. The coordinate functions of s and their nontrivial linear combinations have degree 2 as algebraic normal forms.

Table 1: The final S-box

12346750
$(x_1x_2 + x_2 + x_0 + 1, x_0x_2 + x_2 + x_1 + x_0, x_0x_1 + x_2)$
$1 + (\xi + \xi^2)z + (\xi^2)z^2 + (1 + \xi^2)z^3 + (1 + \xi + \xi^2)z^5 + (1 + \xi)z^6$
$1 + (1 + \xi^2)z^2 + (1 + \xi + \xi^2)z^3 + (1)z^4 + (\xi)z^5 + (\xi)z^6$

We have found that there are 10752 optimal 3-bit S-boxes (every 4th of 15), **R1** yields **R2** and **R3**, **R2** yields **R1** and **R3**, but **R3** does not necessary yield either **R1** or **R2** (there are $38976 > 10752$ S-boxes satisfying **R3**).

It was proved in [6, 9] that any 3-bit substitution s is affine equivalent to one of the 4 canonical: 01234567, 01234576, 01234675 or 01243675. Affine equivalence means that one substitution can be transformed into another by invertible affine transformations of preimages and images. For such transformations the optimality of s is preserved. Since the first 3 canonical substitutions are not optimal, all the optimal S-boxes are affine equivalent to the 4th one and therefore to each other.

The golden S-boxes. Due to **R1**, precisely 4 output differences $\beta = \beta_0\beta_1\beta_2$ correspond to a non-zero input difference $\alpha = \alpha_0\alpha_1\alpha_2$. The output differences form an affine plane of dimension 2. Criterion **R4** requires this plane to satisfy the following equation:

$$\alpha_0\beta_0 + \alpha_1\beta_1 + \alpha_2\beta_2 = 1. \tag{1}$$

This equation has been chosen for reasons of simplicity and symmetry.

R4 can facilitate the estimation of resistance of `Bash-f` against differential attacks. For example, if $\alpha = 100$, then β_0 is necessarily non-zero. In other words, a single error in the first bit of a difference is preserved. It is clear, that other single errors are preserved too.

The next criterion relates to linear attacks. There always exist non-zero $\alpha, \beta \in \{0, 1\}^3$ such that the relation

$$\alpha \cdot x = \beta \cdot s(x)$$

holds with probability $\neq 1/2$ for a random x. The dot here denotes the scalar product of words-as-vectors. The word β above is called the *input mask*, and α is the *output mask*. Due to **R2**, there are 4 acceptable α for each non-zero β, and they again form an affine plane of dimension 2. Criterion **R5** requires that this plane is also described by the equation (1).

R5 implies that acceptable input-output differences are also acceptable output-input masks and vice versa. Hence, since all `Bash-f` components except $S3$ are linear, any acceptable differential characteristic is also an acceptable linear one. Moreover, both characteristics have the same cryptanalytic quality expressed in the number of active S-boxes. In fact, **R5** means that analysis of the strength of `Bash-f` against differential and linear attacks are the same.

By direct calculations, **R4** and **R5** are equivalent: **R4** yields **R5** and vice versa.

Criterion **R6** requires that the mappings $x \mapsto s(x)$ and $x \mapsto s(x) \oplus 001$ have no fixed points. Informally, a fixed point is an idle cycle of s. It is desirable to eliminate such cycles. The mapping $x \mapsto s(x) \oplus 001$ is examined because a bit of a round constant can be added to the last bit of $s(x)$.

The optimal S-boxes satisfying **R4** – **R6** are called golden. All 16 golden S-boxes are listed in Table 2.

The platinum S-boxes. Turning back to the description of `Bash-f`, we see that $S3$ can be implemented with 7 logical operations. It is the minimal number of operations needed to

Table 2: The golden S-boxes

#	S-box	#	S-box	#	S-box	#	S-box
1	12346750	5	30526714	**9**	52743601	13	70123645
2	16307245	**6**	34567201	**10**	56702314	14	72013456
3	23145670	7	36015274	**11**	67143052	15	72451630
4	23507416	**8**	36457012	**12**	67501234	16	74162350

implement an optimal S-box and this minimum is required by Criterion **R7**. We have found that there are 660 optimal S-boxes satisfying **R7** and 8 of them are golden. The ordinal numbers of the suitable golden boxes are highlighted bold in Table 2.

Criterion **R8** provides complexity of the description of s by polynomials over $\mathbb{F}_8$: s satisfies **R8** if in its both permutation polynomials $p(z) = a_0 + a_1 z + \ldots + a_6 z^6$ each coefficient (including the zero one) is repeated no more than three times.

Only the 1st and 10th golden S-boxes satisfy **R7** and **R8**. We call them platinum. The platinum S-boxes differ only in the output y_2 by the constant term 1. We have selected the first platinum S-box: 12346750.

4 The $L3$ linear mapping

The structure. Both the mappings $S3$ and $L3$ transform vertical planes. But whereas $S3$ acts on vertical bit triples separately, $L3$ establishes connections between them. These connections are built using the operations $\oplus$ and $\texttt{RotHi}^d$.

Let (w_0, w_1, w_2) be a preimage of $L3$ and (W_0, W_1, W_2) be an image. In this section we write w^d for $\texttt{RotHi}^d(w)$, $w \in \{0,1\}^{64}$, and $+$ for $\oplus$.

We only consider bijective mappings $L3$, which have the following structure:

$$\begin{aligned} W_0 &\leftarrow w_0 + w_1 + w_2, \\ W_1 &\leftarrow w_1 + a^{m_1} + b^{n_1}, \\ W_2 &\leftarrow w_2 + c^{m_2} + d^{n_2}. \end{aligned}$$

Here a, b, c, d are words constructed from w_0, w_1, w_2. They are either the base words w_i or their sum $w_i + w_j$ or, in the case of c and d, elements of the set $\{w_1 + A, w_1 + B, A + B\}$, where $A = a^{m_1}$, $B = b^{n_1}$.

To calculate W_j, we have to use 4 rotations and at least 6 additions. Criterion **R1** requires the number of additions to be minimal. That means that the words a, b, c, d can be computed without extra cost during the main calculations.

Let us illustrate **R1**. If $L3$ is represented by a tuple (a, b, c, d), then $L3 = (w_0, W_0, w_2, w_2{+}B)$ satisfies **R1**, but $L3 = (W_0, w_0 + w_1, W_1, w_1 + w_2)$ does not. Indeed, in the first case the sum $w_2 + B$ can be calculated along with W_1. In the second case the sums $w_0 + w_1$, $w_1 + w_2$, $W_0 = w_0 + w_1 + w_2$ must be calculated with only 2 additions, which is impossible.

In the next criteria we assume that $L3$ has a *reasonable* shift tuple $[m_1, n_1, m_2, n_2]$. It means that the residues

$$0, m_1, n_1, m_2, n_2, m_1 + m_2, m_1 + n_2, n_1 + m_2, n_1 + n_2 \bmod 64$$

are pairwise distinct.

Criterion **R2** requires that any bit of any of the words w_i affects several (at least one) bits of each of the words W_j.

Criterion **R3** relates to the inverse mapping $L3^{-1}$. It requires that any bit of any of the words W_j affects approximately half of the bits in each of the words w_i. In other words, **R3** requires that the matrix of $L3^{-1}$ contains about equal numbers of ones and zeros.

The characteristic

$$\text{BranchNumber}(L3) = \min_{x \in \{0,1\}^{192} \setminus \{0^{192}\}} (\text{wt}(x) + \text{wt}(L3(x)))$$

describes the diffusion properties of $L3$. Here $\text{wt}(x)$ is the Hamming weight of x. Criterion **R4** requires that $\text{BranchNumber}(L3) \geq 5$. The threshold 5 is the maximum which can be achieved with the chosen structure of $L3$.

There are 20 structures satisfying **R1** — **R4**. They are listed in Table 3.

Table 3: The optimal structures of $L3$

#	(a, b, c, d)	#	(a, b, c, d)
1	(w_0, W_0, w_0, W_1)	11	(w_0, W_0, A, W_1)
2	(w_0, W_0, w_1, W_1)	**12** (III)	$(w_0, W_0, W_1, w_1 + B)$
3 (I)	$(w_0, W_0, w_2, w_1 + B)$	13	$(w_0, W_0, W_1, A + B)$
4	$(w_0, W_0, w_2, A + B)$	14	$(w_0, w_0 + w_2, W_1, w_1 + B)$
5	$(w_0, W_0, w_0 + w_1, w_1 + B)$	15	$(w_0, w_1 + w_2, W_1, A + B)$
6	$(w_0, W_0, w_0 + w_1, A + B)$	**16** (IV)	(w_2, W_0, w_0, W_1)
7 (II)	$(w_0, W_0, w_0 + w_2, W_1)$	17	$(W_0, w_0 + w_1, w_0, W_1)$
8	$(w_0, W_0, w_0 + w_2, w_1 + A)$	18	$(W_0, w_1 + w_2, w_0, W_1)$
9	$(w_0, W_0, w_1 + w_2, W_1)$	19	$(W_0, w_1 + w_2, W_1, w_1 + A)$
10	$(w_0, W_0, w_1 + w_2, w_1 + A)$	20	$(W_0, w_1 + w_2, W_1, A + B)$

S-box activation. Consider the mappings $L3$ and $S3$ in the context of differential attacks. Assume that they transform the differences in pairs of preimages rather than the preimages themselves. An input difference $w_0 \parallel w_1 \parallel w_2$ of $L3$ is transformed into an output difference $W_0 \parallel W_1 \parallel W_2$, which is the input difference of $S3$. The difference $w_0 \parallel w_1 \parallel w_2$ *activates* the v-th S-box, if the v-th vertical bit triple $W_{0v}W_{1v}W_{2v} \neq 000$. To protect against differential attacks, it is important that differences of small Hamming weight or, in other words, errors in a small amount of positions activate as many S-boxes as possible.

Let $\min[i \to ?]$ be the minimal number of S-boxes that are activated by i errors in the input of $L3$. Criterion **R4** requires the characteristics $\min[i \to ?]$ with $i = 1, 2, 3$ to be maximal possible:

$$\min[1 \to ?] = 4, \quad \min[2 \to ?] = 3, \quad \min[3 \to ?] = 3.$$

There are 4 structures in Table 3 satisfying **R4**. They are marked with the additional indices I — IV.

Backward activation. Let $\min[? \to j]$ be the minimal number of errors in an input of $L3$ needed to activate exactly j S-boxes. Criterion **R5** requires that $L3$ has large enough characteristics $\min[? \to 1]$ and $\min[? \to 2]$ under certain parameters $[m_1, n_1, m_2, n_2]$.

The upper bounds of the target characteristics are presented in Table 4. Each bound can be reached for some $[m_1, n_1, m_2, n_2]$. The table shows that the type III mappings do not satisfy **R5**.

Table 4: The upper bounds for $\min[? \to 1]$, $\min[? \to 2]$

Type	$\min[? \to 1]$	$\min[? \to 2]$
I	103	74
II	109	49
III	109	3
IV	109	51

Note that there is no need to calculate $\min[? \to 3]$: it equals to 2 due to the fact that $\min[1 \to ?] > 3$ and $\min[2 \to ?] = 3$. Instead of $\min[? \to 3]$ we use the characteristic $\min[?^{>3} \to 3]$: the smallest greater than 3 number of errors needed to activate 3 S-boxes. Criterion **R6** requires this characteristic to be large enough under certain shift tuples. For the mappings of types II and III the target characteristic equals to 4, for the type I it can reach 51. After **R6** we only keep the mappings of the latter type.

The criteria **R5**, **R6** suppress (lock) getting errors of small weight in backward iterations of `Bash-f`. Such locks will play a crucial role in our estimation of the number of active S-boxes.

Inversion. The "backward locks" are caused by complexity of the inversion of $L3$. Let us analyze this inversion, i.e. the determination of (w_0, w_1, w_2) from (W_0, W_1, W_2).

A type I mapping can be inverted as follows: Solve the equation

$$w_2 + w_2^{m_1} + w_2^{m_2} + w_2^{m_1+m_2} + w_2^{m_1+n_2} = W_0^{m_1+n_2} + W_0^{m_1+n_1+n_2} + W_1^{n_2} + W_2 + W_2^{m_1}$$

for w_2, then determine

$$w_0 \leftarrow W_1^{-m_1} + W_2^{-m_1-n_2} + w_2^{-m_1-n_2} + w_2^{m_2-m_1-n_2},$$
$$w_1 \leftarrow W_0^{n_1} + W_2^{-n_2} + w_2^{-n_2} + w_2^{m_2-n_2}$$

(the operations in the exponents are done modulo 64).

In general, and in particular in this case, the main part of the inversion is finding a solution w_i of a equation

$$f(w_i) = F(W_0, W_1, W_2).$$

The left side of this equation is determined by a characteristic polynomial f that belongs to the quotient ring $R = \mathbb{F}_2[x]/(x^{64}+1)$. In order to solve the equation, the inverse (in R) polynomial f^{-1} should be found and applied to the both sides:

$$w_i = f^{-1}(F(W_0, W_1, W_2)).$$

The polynomial f is invertible in R if and only if it contains an odd number of monomials, i.e. $f(1) = 1$. Under invertibility, $f^{-1}(x) = (f(x))^{63}$. It is interesting that the type I mappings have characteristic pentanomials, while the mappings of types II — IV have characteristic trinomials. Perhaps, this is the reason for the advantages of the type I.

Shifts. $L3$ has such a structure that the rotation of input words w_i by d bits implies the rotation of the output words W_j also by d bits. This means that the differences $w_0 \parallel w_1 \parallel w_2$ and $w_0^d \parallel w_1^d \parallel w_2^d$ activate the same number of S-boxes. We consider these differences are equivalent and choose a canonical representative in each equivalence class. The choice is made in the lexicographical manner: a canonical difference $w_0 \parallel w_1 \parallel w_2$ should contain ones as early to the left as possible.

Table 5: The lower bounds for $\#[i \to j]$

$i \backslash j$	3	4	5	6
2	3	0	3	10
3	3	3	22	
4	0	18	44	
5	0	12		
6	0	7		

Let $\#[i \to j]$ be the number of canonical differences with i errors needed to activate j S-boxes. Some lower bounds for $\#[i \to j]$ are presented in Table 5. All the bounds are tight.

Criterion **R7** requires to use such shift tuples $[m_1, n_1, m_2, n_2]$ that provide the lower bounds of the table along with the following constraints:

$$\min[? \to 1] \geq 70, \quad \min[? \to 2] \geq 50, \quad \min[?^{>3} \to 3] \geq 40, \quad \#[3 \to 6] \leq 64.$$

Appropriate shift tuples are presented in Table 6.

The characteristics of $L3[m_1, n_1, m_2, n_2]$ do not change, if we multiply the elements of $[m_1, n_1, m_2, n_2]$ by an odd number modulo 64. Such a multiplication establishes the equivalence relation among shift tuples. The table presents only one tuple from each equivalence class, such one that contains 1 as left as possible.

Table 6: The optimal shift tuples

#	$[m_1, n_1, m_2, n_2]$	$\min[? \to 1]$	$\min[? \to 3]$	$\min[?^{>3} \to 3]$
1	$[1, 4, 55, 16]$	83	50	40
2	$[1, 10, 25, 58]$	91	56	40
3	$[1, 13, 47, 57]$	93	60	40
4	$[1, 14, 53, 44]$	87	61	40
5	$[1, 49, 29, 55]$	98	61	45
6	$[1, 51, 17, 6]$	88	60	45
8	$[1, 52, 45, 15]$	88	52	40
9	$[1, 53, 14, 5]$	85	66	41
10	$[1, 53, 50, 45]$	94	58	40
11	$[1, 54, 57, 26]$	81	58	41
12	$[1, 57, 18, 30]$	75	57	43
13	$[1, 60, 36, 47]$	72	54	43
14	$[8, 53, 14, 1]$	87	59	43
15	$[18, 10, 40, 1]$	89	67	40

Eventually, the tuple $[8, 53, 14, 1]$ was chosen. We explain the choice in the following section.

5 The P permutation

Digraphs. The permutation P rotates the horizontal planes and simultaneously applies the underlying permutation π_i to the words of the i-th plane, $i = 0, 1, 2$. The horizontal planes

are rotated upward and therefore we call P an *up-permutation*. We also use the notion of *down-permutations* which rotate planes downward.

Let G be the digraph with the vertices $0, 1, \ldots, 7$ and the arcs

$$\{(u, v) \colon v \in \{\pi_0(u), \pi_1(u), \pi_2(u)\}\}.$$

An arc (u, v) shows that P maps some word from the u-th vertical plane into the v-th one. If the word is mapped into the c-th horizontal plane, then the arc is labeled by c.

Throughout the remainder of this section we will consider G as a primary configuration and P as a secondary one. It means that G determines P, not vice versa.

We can determine P from G only up to the direction of rotation of the horizontal planes. Among the suitable up- and down-permutations we choose the one having the maximal order. In case the orders are equal we favor the up-permutation.

Strong regularity. Each vertex of G has in- and out-degrees of 3, i.e. G is the 3-regular digraph. Let M be the adjacency matrix of G. Criterion **R1** requires that all elements of M^2 are positive or, in other words, there is a walk of length 2 between any two vertices.

Criterion **R1** provides fast diffusion between words: after only 3 rounds of `Bash-f` each output word depends on each of the input words.

Due to the 3-regularity, the sum of elements of any row in M^2 equals to 9. Since the entries of M^2 are positive, the row contains exactly one element equal to 2. Criterion **R2** requires that all the elements 2 are on the main diagonal of M^2. Under **R2**, there exists exactly one walk in G of length 2 connecting any two distinct vertices and exactly 2 walks of length 2 from any vertex to itself.

Criterion **R2** implies that G is strongly regular. We recall that a k-regular directed graph with ν vertices is called *strongly regular* if all diagonal elements of its adjacency matrix equal to zero and there exist integers t, λ and μ such that

$$M^2 = tI + \lambda M + \mu(J - I - M).$$

Here I is the identity matrix of order ν, J is the all-ones square matrix of the same order.

In our case $(\nu, k, t, \mu, \lambda) = (8, 3, 2, 1, 1)$. The strongly regular digraph with these parameters is registered in the database [4]. It is constructed according to the scheme proposed by Jorgensen in [7]: for $\mu \mid (k-1)$, $\nu = (k+1)(k-1)/\mu$, $t = \mu + 1$, $\lambda = \mu$, the digraph G has an arc (u, v) if and only if $u + kv \equiv 1, 2, \ldots, k \pmod{\nu}$.

Jorgensen's digraph is isomorphic to the digraph shown on Figure 2. We have verified that this digraph is isomorphic to any other strongly regular digraph with the required parameters. The isomorphism does not change the characteristics of P which are of interest to us, therefore we have chosen the digraph of Figure 2 as G.

Labeling the arcs. The base digraph of Figure 2 has no labels assigned to its arcs. When assigning the labels, it is only required that no two arcs with the same label are directed either to or from the same vertex.

We make the additional requirement **R3**: the digraph should remain the same if we shift all vertices clockwise by two positions. This requirement provides the *almost-symmetry* which allows us to make fast error propagation throughout the rounds of the sponge function. The full symmetry of the digraph means that it remains the same when the vertices are shifted by one position. Unfortunately, fully symmetric digraphs do not exist.

All almost-symmetric digraphs can be derived from the two canonical digraphs shown on Figure 3 by renumbering arc labels. The left digraph of the figure is denoted by G_1 and the right one by G_2. A renumbering is described by a permutation abc over the set $\{0, 1, 2\}$: the

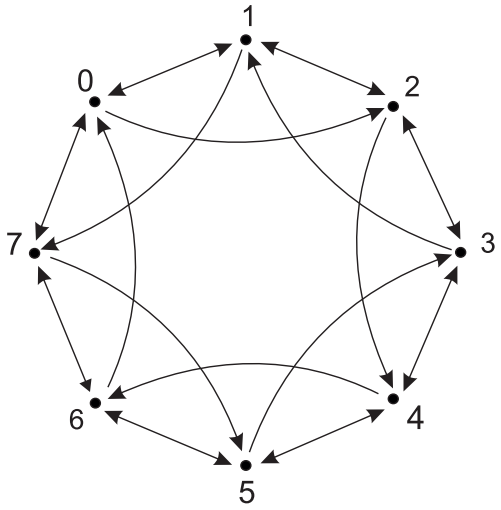

Figure 2: The base digraph

labels 0, 1 and 2 are replaced by a, b and c, respectively. The renumberings are sorted in the natural order

$$012, 120, 201, 210, 102, 021$$

and are indexed from 0 to 5. Let G_i^j be the digraph derived from G_i by the j-th renumbering, $i = 1, 2$, $j = 0, 1, \ldots, 5$.

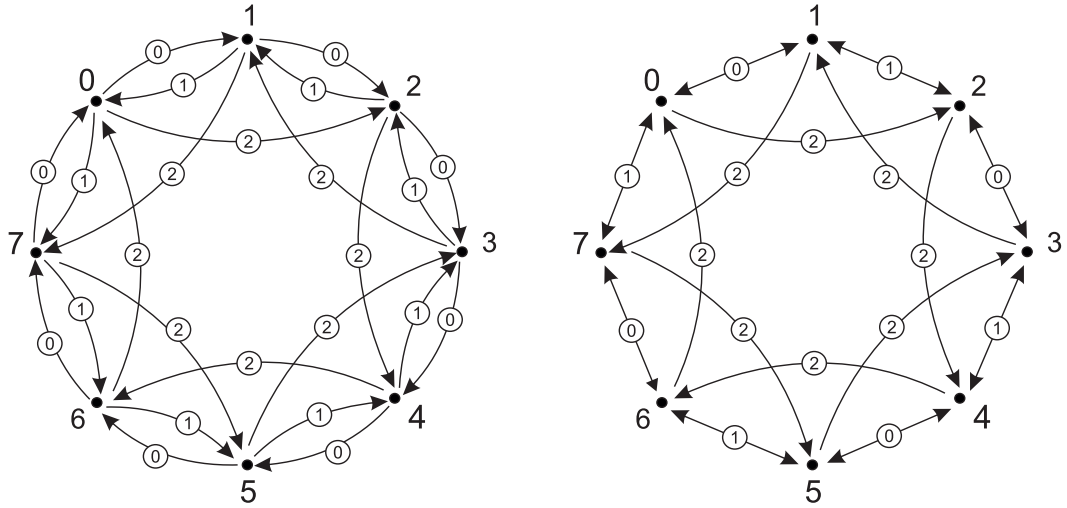

Figure 3: The canonical almost-symmetric digraphs

Walks in G. Let $S[v, c]$ be the c-th word of the v-th vertical plane, $v \in \{0, 1, \ldots, 7\}$, $c \in \{0, 1, 2\}$. In other words, $S[v, c] = S_{3v+c}$.

With each word $S[v, c]$ we associate a set $A[v, c] \subseteq \{0, 1, \ldots, 63\}$. This set consists of indices of vertical bit triples which depend on the 0th bit of $S[v, c]$ after $L3$ is applied. If the v-th column is transformed by $L3[m_1, n_1, m_2, n_2]$ of type I, then

$$A[v, 0] = \{0, m_1, n_1, n_1 + n_2\},$$
$$A[v, 1] = \{0, n_1, n_2, n_1 + n_2\},$$
$$A[v, 2] = \{0, n_1, m_2, n_1 + n_2\}.$$

Let $(v_0, v_1, \ldots, v_d)$ be a walk in G and c_i be a label of its arc (v_{i-1}, v_i), $i = 1, 2, \ldots, d$. The walk characterizes how errors propagate through the rounds of the sponge function. Specifically, consider how the error in the k-th bit of $S[v_0, c_0]$ is processed by the cascade $L3$-then-$S3$-then-P. After $L3$ the error activates S-boxes of the v_0-th vertical plane whose indices are from the set $\{k\} + A[v_0, c_0]$. After $S3$ input differences of the activated S-boxes can affect each bit of the corresponding output differences. After P some of the affected bits become bits of $S[v_1, c_1]$ and, in turn, in the next round can affect the bit triples of the v_1-th vertical plane having the indices from the set $\{k\} + A[v_0, c_0] + A[v_1, c_1]$. After all d steps along the chosen walk the affected bit triples are those which lay in the v_d-th plane and whose indices belong to the set $\{k\} + A[v_0 v_1 \ldots v_d, c_0]$, where

$$A[v_0 v_1 \ldots v_d, c_0] = A[v_0, c_0] + A[v_1, c_1] + \ldots + A[v_d, c_d].$$

By the sum of sets A and B in the previous expressions we understand the following:

$$A + B = \{(a + b) \bmod 64 \colon a \in A, b \in B\}.$$

We can omit the parentheses since such a summation is associative. In particular, the cardinality of the set $\{k\} + A[v_0 v_1 \ldots v_d, c_0]$ does not depend on k. We can use $k = 0$ without changing the cardinality.

Now consider the walks of length 2 in more detail. Define

$$B[v_0 v_2, c_0] = \cup_{v_1} A[v_0 v_1 v_2, c_0],$$

where the union is over the vertices v_1 such that (v_0, v_1, v_2) is a walk in G. We have exactly one choice for v_1 if $v_0 \neq v_2$ and two choices if $v_0 = v_2$. The set $B[v_0 v_2, c_0]$ describes the influence of 0th bit of $S[v_0, c_0]$ on the bit triples of the v_2-th vertical plane after 3 rounds.

Criterion **R4** requires the cardinalities $|B[v_0 v_2, c_0]|$ to be as close to 64 as possible for all vertices v_0, v_2 and initial labels c_0. It means that any input bit of the hash state after 3 rounds affects as many output bits as possible.

Shift tuples. Criterion **R4** relates both to the labels in G and to the shift tuples in $L3$. Until now we have considered the single $L3$ mapping with a fixed shift tuple. But in fact 8 separate $L3$ mappings are used, each potentially having its own tuple.

We have decided to define the shift tuple for the v-th vertical plane as

$$g^v [m_1, n_1, m_2, n_2] \bmod 64.$$

Here $[m_1, n_1, m_2, n_2]$ is a base tuple and $g \in \{7, 9, 23, 25, 39, 41, 55, 57\}$ is an element of the order 8 modulo 64.

Under such a choice of shift tuples, the sets $A[v, c]$, $v = 0, 1, \ldots, 7$, are derived from each other by multiplication by g, repeating in a cycle. Due to this fact and almost-symmetry of G, the simultaneous maximization of $|B[v_0 v_2, c_0]|$ becomes easier. Indeed, for an even d and $v'_i = (v_i + d)$ the sets $B[v_0 v_2, c_0]$ and $B[v'_0 v'_2, c_0]$ differ by multiplication by g^d and, therefore, have the same cardinality.

Recall that the considered in this paper characteristics of $L3$ do not change if the shift tuple $[m_1, n_1, m_2, n_2]$ is multiplied by an odd factor g^d.

The generation of parameters. Labeling arcs in G and setting the shifts in $L3$ have been done as follows.

1. For ($[m_1, n_1, m_2, n_2] \in$ Table 6, $i \in \{1, 2\}$, $j \in \{0, 1, \ldots, 5\}$, $g \in \{7, 9, \ldots, 57\}$) do:

1) $r \leftarrow \max_{v_0,v_2,c_0}(64 - |B[v_0v_2, c_0]|)$;

2) $R \leftarrow \sum_{v_0,v_2,c_0}(64 - |B[v_0v_2, c_0]|)$;

3) if $R \leq 3072$ append the configuration $([m_1, n_1, m_2, n_2], G_i^j, g)$ to the list of the optimal ones.

2. From the list of the optimal configurations, select one for which the vector (r, R) is lexicographically minimal. If several configurations have the same minimal vector (r, R), select the first one of them.

The characteristics r and R describe the dependency matrix of the 3 rounds of `Bash-f`. It is a square matrix of order 1536, whose (i, j)-th element is the indicator of whether the error in the i-th bit of the `Bash-f` state can trigger the error in the j-th bit after 3 rounds. The characteristic r describes the maximum "incompleteness" of the 64×64 blocks of the dependency matrix, while R describes the "full incompleteness". We have selected the threshold of 3072 meaning that the matrix contains no more than $3072 \cdot 3 \cdot 64$ zero items, i.e. the matrix is incomplete by no more than $\frac{3072 \cdot 3 \cdot 64}{1536^2} = \frac{1}{4}$.

Table 7: The optimal configurations ($[8, 53, 14, 1]$, G_i^j, g)

#	G_i^j	g	r	R	#	G_i^j	g	r	R
1	G_2^1	9	27	2988	**7**	G_2^4	7	25	2968
2	G_2^1	57	27	2988	8	G_2^4	9	26	2964
3	G_2^2	7	25	2980	9	G_2^4	55	25	2968
4	G_2^2	9	25	2996	10	G_2^4	57	26	2964
5	G_2^2	55	25	2980	11	G_2^5	9	28	2952
6	G_2^2	57	25	2996	12	G_2^5	57	28	2952

All the optimal configurations have the same basic shift tuple $[m_1, n_1, m_2, n_2] = [8, 53, 14, 1]$. Other parts of configurations are listed in Table 7. Eventually, the 7th configuration was selected.

References

[1] Bee2: A cryptographic library. Avail. at `https://github.org/agievich/bee2`, last access 04.06.2016.

[2] Bertoni G., Daemen J., Peeters M., Van Assche G. Cryptographic sponge functions. Version 0.1. Avail. at `http://sponge.noekeon.org/CSF-0.1.pdf`, 14.01.2011.

[3] Bertoni G., Daemen J., Peeters M., Van Assche G. Sponge functions. Ecrypt Hash Workshop 2007, May 2007.

[4] Brouwer A. E., Hobart S. A. Parameters of directed strongly regular graphs. Avail. at `http://homepages.cwi.nl/~aeb/math/dsrg/dsrg.html`, last access 04.06.2016.

[5] Cryptography standards of Belarus. Avail. at `http://apmi.bsu.by/resources/std` (in Russian), last access 14.03.2016.

[6] Harrison M. A. On the Classification of Boolean Functions by the General Linear and Affine Group. Journal of the Society for Industrial and Applied Mathematics, 12: 284–299, 1964.

[7] Jorgensen L. K. Directed strongly regular graphs with $\mu = \lambda$. Discrete Math, 231: 289–293, 2001.

[8] Lidl R., Niederreiter H. Finite fields. Cambridge University Press, 1997.

[9] Lorens C. S. Invertible Boolean Functions. IEEE Transactions on Electronic Computers, EC-13(5): 529–541, 1964.

6 Appendix

6.1 Proofs

Proposition 1. *A polynomial $f(x) \in R = \mathbb{F}_2[x]/(x^{64}+1)$ is invertible if and only if it contains an odd number of monomials, i.e. $f(1) = 1$. Under invertibility,*

$$f^{-1}(x) = (f(x))^{63}.$$

Proof. To be invertible it is necessary and sufficient that $f(x)$ is coprime to the modulus $x^{64} + 1 = (x+1)^{64}$. It means that $f(x)$ is not a multiple of $x+1$ or $f(1) \neq 0$.

Let $f(x) = \sum_{i=0}^{63} a_i x^i$ be invertible, that is, $\sum_i a_i = 1$. Then

$$f(x)^{64} = \sum_{i=0}^{63} a_i \left(x^{64}\right)^i \equiv \sum_{i=0}^{63} a_i = 1 \pmod{x^{64}+1}$$

and $f^{-1} = f^{63}$. □

Proposition 2. *Each strongly regular digraph with the parameters $(8, 3, 2, 1, 1)$ is isomorphic to the digraph of Fig. 2.*

Proof. We can return from a vertex v back to itself along exactly 2 walks of length 2: (v, v', v) and (v, v'', v). These walks describe bidirectional arcs $[v, v']$ (between v and v') and $[v, v'']$ (between v and v''). Bidirectional arcs induce bidirectional cycles: a cycle $[v_0, v_1, v_2, \ldots, v_{n-1}, v_0]$ consists of the arcs $[v_0, v_1], [v_1, v_2], \ldots, [v_{n-1}, v_0]$.

Besides (v, v') and (v, v''), there exists the third arc (v, v''') whose head is v. This arc does not belong to any bidirectional cycle. Call such an arc ordinary. Ordinary arcs induce ordinary cycles.

The bidirectional cycles of length 1 and 2 are impossible. Therefore, there exist the following variants of a cycle structure:

1. A full bidirectional cycle of length 8.
2. Two bidirectional cycles of length 4.
3. Bidirectional cycles of length 5 and 3.

The 2nd variant is impossible. Indeed, if $[v_0, v_1, v_2, v_3, v_0]$ is a cycle of length 4 then v_2 can be reached from v_0 by two walks of length 2: (v_0, v_1, v_2) and (v_0, v_3, v_2). It contradicts the strong regularity.

Consider the 3rd variant. Let $[v_0, v_1, v_2, v_0]$ be a cycle of length 3. From each vertex v_i there goes exactly one arc outside the cycle. Let it end at a vertex u_i. The vertices u_0, u_1, u_2 lay on the bidirectional cycle of length 5. Consequently, there exist two vertices u_i and u_j which are connected by a bidirectional arc. Without loss of generality, let these vertices be u_0 and u_1. Then v_0 is connected with $u_1 \neq v_0$ by two walks of length 2: (v_0, u_0, u_1) and (v_0, v_1, u_1). It again contradicts the strong regularity.

Only the first variant of the bidirectional cycle structure remains possible. By renumbering vertices, we can bring the sole bidirectional cycle to the form $[0, 1, 2, \ldots, 7, 0]$.

Let us consider the ordinary arcs. They can only have a form $(i, i+2)$ or $(i, i-2)$ (here and below addition and subtraction are performed modulo 8). Indeed,

a) if an arc $(i, i+1)$ is present, then we can reach from i only 2 vertices in 1 step (by walks of length 1) and, therefore, no more than 6 vertices in 2 steps;

b) if an arc $(i, i+3)$ is present, then we can not reach $i+3$ from i in 2 steps: we can reach it either from $i+2$ or $i+4$, but the arcs $(i, i+2)$ and $(i, i+4)$ are absent;

c) if an arc $(i, i+4)$ is present, then we can not reach $i+4$ from i in 2 steps: we can reach it either from $i+3$ or $i+5$, but the arcs $(i, i+3)$ and $(i, i+5)$ are absent;

d) the arcs $(i, i+5)$ are processed in the same way as the arcs $(i, i+3)$, and $(i, i+7)$ in the same way as $(i, i+1)$.

The restrictions on the ordinary arcs imply that the ordinary cycles can only have the following forms: $(i, i+2, i+4, i+6, i)$ (the clockwise cycle) or $(i, i-2, i-4, i-6, i)$ (the counterclockwise cycle). The digraph must contain exactly 2 such cycles: one clockwise and one counterclockwise. By renumbering the vertices of the digraph we can bring it to the form of the figure. □

6.2 Dependency matrices

Fig. 4 represents MD-matrices after the first 4 rounds of `Bash-f`. The non-zero elements of the matrices are shown as black pixels. The index i numerates rows of the matrices from top to down. The index j numerates columns from left to right.

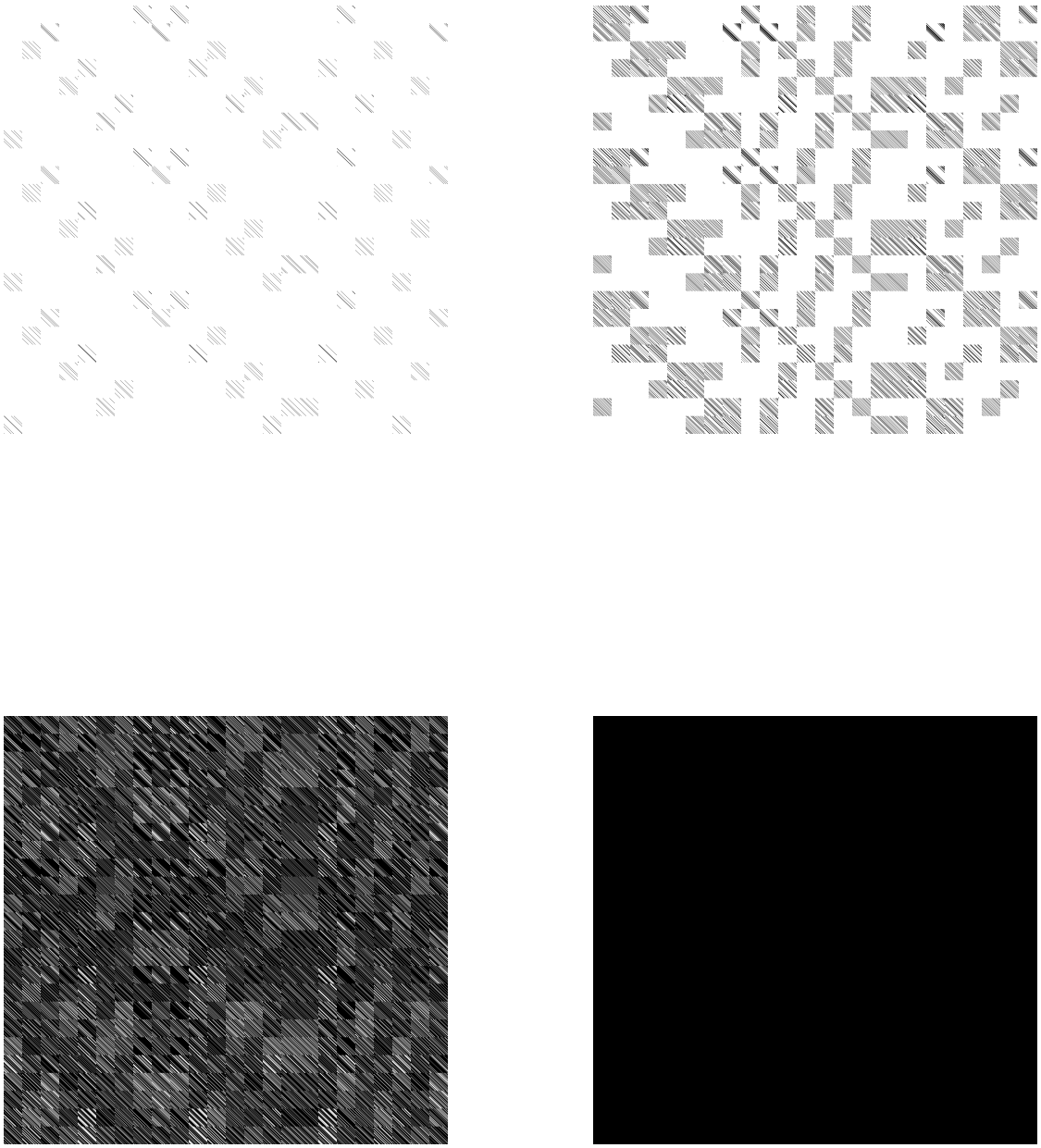

Figure 4: The dependency matrices after 1 — 4 rounds