# On the S-box in GAGE and InGAGE

Danilo Gligoroski

Department of Information Security and Communication Technology, NTNU, Norway
danilog@ntnu.no

**Abstract.** We discuss some properties of the S-box of GAGE and its nonlinear layer. The submission documentation of GAGE mentions that "*The nonlinear operations can be realized either as reading from small lookup tables or as register operations that perform only* `SHIFT`, `XOR`, `AND` *and* `NOT` *operations*", but it is not explained how can it be done. In this note we describe how that can be realized. As a consequence, we show that modeled as a sequence of vector operations in the vector space $GF(2)^{b+2}$, GAGE has only one nonlinear vector `AND` operation per round. It makes GAGE a potential lightweight cryptographic candidate for the recent increased interest in designing of MPC protocols, SNARKs, and post-quantum signature schemes.

## 1 Introduction

GAGE is a family of lightweight cryptographic hash functions submitted as a Round 1 candidate for the NIST Lightweight Cryptography Standardization process [2]. It is a sponge-based family [1] with states between 232 up to 576 bits, and rates from 8 up to 128 bits. InGAGE is an authenticated cipher based on GAGE. Among all 56 submissions, GAGE uses the smallest S-box with a size $4 \times 2$, and the smallest round constants that have a size of only 2 bits. Another unique design property of GAGE's S-box is that it is applied in an interleaved fashion unlike the traditional use of S-boxes where the state is separated in disjunctive subsets. The submission documentation of GAGE mentions that "*The nonlinear operations can be realized either as reading from small lookup tables or as register operations that perform only* `SHIFT`, `XOR`, `AND` *and* `NOT` *operations*", but it is not explained how can it be done. In this note we describe how that can be realized, and we discuss some possible benefits of that.

Next, we repeat some of the definitions for the nonlinear part of GAGE as given in [2].

## 2 The nonlinear layer and the choice of the S-box in GAGE

The nonlinear substitution part uses one 4-to-2 bits s-box $Q$ that is applied in an interleaved way on the state of $b$ bits. Interleaved application means that the set of state bits is split in 2-bit subsets, and they enter the s-boxes in two different roles: as two left most bits and as two rightmost bits. The s-box is applied in parallel. This interleaved application makes the substitution layer as one big s-box with $(b+2)$-to-$b$ bits. For transforming the two left most bits of the state, a two bit round constant $l = (l_{0,1}, l_{1,1})$ is used. A graphical presentation of the substitution layer is given in Figure 1.

The 4-to-2 bits s-box $Q$ can be represented in a usual manner as one table with $2^4$ elements, where the elements are from the set $\{0, 1, 2, 3\}$ (given in Table 1).

| Input: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Output: | 1 | 0 | 3 | 2 | 0 | 2 | 1 | 3 | 2 | 3 | 0 | 1 | 3 | 1 | 2 | 0 |

Table 1: The 4-to-2 bits s-box $Q$.

It can be also represented as in expression (1) as a $4 \times 4$ multiplication table for a binary operation $*$ over the set $Q = \{0, 1, 2, 3\}$ (Note: here we abuse and overload the use of the symbol $Q$ both as a

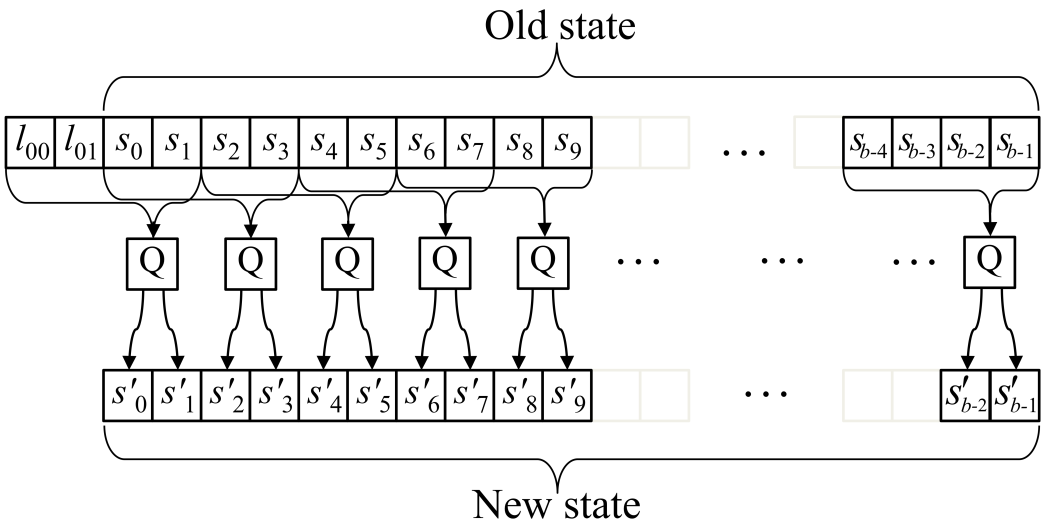
Old state
$l_{00}$ $l_{01}$ $s_0$ $s_1$ $s_2$ $s_3$ $s_4$ $s_5$ $s_6$ $s_7$ $s_8$ $s_9$ ... $s_{b-4}$ $s_{b-3}$ $s_{b-2}$ $s_{b-1}$
Q Q Q Q Q ... ... ... Q
$s'_0$ $s'_1$ $s'_2$ $s'_3$ $s'_4$ $s'_5$ $s'_6$ $s'_7$ $s'_8$ $s'_9$ ... $s'_{b-2}$ $s'_{b-1}$
New state

Fig. 1: A graphical presentation of the nonlinear part in GAGE.

symbol for the s-box and for a set.)

$$\begin{array}{c|cccc} * & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 0 & 3 & 2 \\ 1 & 0 & 2 & 1 & 3 \\ 2 & 2 & 3 & 0 & 1 \\ 3 & 3 & 1 & 2 & 0 \end{array} \tag{1}$$

Finally, the s-box $Q$ can be represented in ANF form as a vector-valued Boolean function that receives 4 input variables $x_1, x_2, x_3, x_4$ and outputs two Boolean functions:

$$Q(x_1, x_2, x_3, x_3) = (f_1(x_1, x_2, x_3, x_3), f_2(x_1, x_2, x_3, x_3))$$

given with the expression (2). Operations of addition and multiplication are in the finite filed $GF(2)$.

$$Q(x_1, x_2, x_3, x_4) = (x_1 + x_3 + x_2x_3 + x_2x_4,\ 1 + x_1 + x_2 + x_2x_3 + x_4 + x_2x_4) \tag{2}$$

We describe the nonlinear substitution part using the algebraic structure $(Q, *)$. For this, a state of $b$ bits that is subject of transformation is represented as an array $A = \{a_0, \ldots, a_{b/2-1}\}$ of two bit elements. The notation $x * y = z$ means $Q(x, y) = z$.

**Algorithm 1** Nonlinear substitution of $A = \{a_0, \ldots, a_{b/2-1}\}$ with $(Q, *)$ and a leader $l$.

```
1: procedure D-TRANSFORMATION($l$, $A = \{a_0, \ldots, a_{b/2-1}\}$ )
2:     $ldr \leftarrow l$
3:     for $i = 0$ to $b/2 - 1$ do
4:         $nextldr \leftarrow a_i$
5:         $a_i \leftarrow ldr * nextldr$
6:         $ldr \leftarrow nextldr$
7:     endfor
8:     Return $A$
```

We give here a brief overview of the mathematical properties for the chosen s-box in GAGE and the mode of operation for using that s-box.

**Definition 1.** *A quasigroup $(Q,*)$ is a groupoid satisfying the law*

$$(\forall u,v \in Q)(\exists!\ x,y \in Q) \quad u*x=v \ \&\ y*u=v. \tag{3}$$

What is characteristic for quasigroups is that equations of the type: $a*x=b$ or $x*a=b$ have unique solutions. The binary operation $*$ induces another binary operation on $Q$, called *left conjugate or left parastrophe*, defined as $x=a\backslash_* b$ iff $a*x=b$. It is obvious that $(Q,\backslash_*)$ is a quasigroup and that the algebra $(Q,*,\backslash_*)$ satisfies the identities

$$x\backslash_*(x*y)=y, \quad x*(x\backslash_* y)=y. \tag{4}$$

Consider an alphabet (i.e., a finite set) $Q$. By $Q^+$ we denote the set of all nonempty words (i.e., finite strings) formed by the elements of $Q$. Depending on the context, we use two notations for elements of $Q^+$: $a_1a_2\dots a_n$ and $(a_1,a_2,\dots,a_n)$, where $a_i\in Q$.

**Definition 2.** *Let $(Q,*)$ be a quasigroup and $M=a_1a_2\dots a_n\in Q^+$ For each $l\in Q$ we define two functions $e_{l,*},d_{l,*}:Q^+\longrightarrow Q^+$ as follows:*

$$e_{l,*}(M)=b_1b_2\dots b_n \iff b_1=l*a_1,\ b_2=b_1*a_2,\dots,\ b_n=b_{n-1}*a_n,$$

$$d_{l,*}(M)=c_1c_2\dots c_n \iff c_1=l*a_1,\ c_2=a_1*a_2,\dots,\ c_n=a_{n-1}*a_n,$$

*The functions $e_{l,*}$ and $d_{l,*}$ are called a quasigroup string e–transformation (or e–transformation for short) and a quasigroup string d–transformation (or d–transformation for short) of $Q^+$ based on the operation $*$ with leader $l$.*

Graphical representations of $e$–transformation and $d$–transformation are shown in Fig. 2.

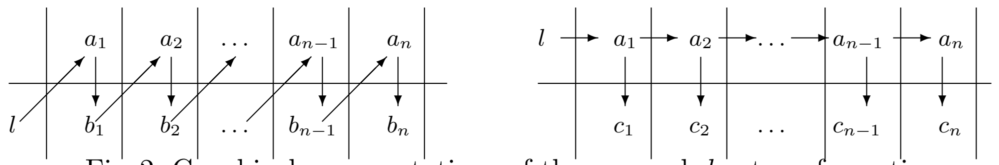

Fig. 2: Graphical representations of the $e_{l,*}$ and $d_{l,*}$ transformations

Using Definition 2 and the identities (4) it is easy to prove the following theorem.

**Theorem 1.** *If $(Q,*)$ is a finite quasigroup, then $e_{l,*}$ and $d_{l,\backslash_*}$ are mutually inverse permutations of $Q^+$, i.e.,*

$$d_{l,\backslash_*}(e_{l,*}(M))=M=e_{l,*}(d_{l,\backslash_*}(M))$$

*for each leader $l\in Q$ and for every string $M\in Q^+$.* □

In [3] we proved that some of the well known modes of operations of block ciphers (such as CBC, OFB and CTR) are actually $e$–transformation or $d$–transformation. While the $e$–transformation is in essence a sequential procedure (as it is the case with the CBC encryption mode), the $d$–transformation is essentially a parallel procedure (a well known fact also for the decryption of the CBC mode). In contrast to the design of the stream cipher Edon80 [4], the design of GAGE instead of the sequential $e$–transformation uses the parallel $d$–transformation.

There are 576 quasigroups of order 4, and if we sort all of them in a lexicographic order, in GAGE we are using the quasigroup Nr. 173. Here are the criteria for choosing this particular quasigroup:

1. The quasigroup and its left conjugate (left parastrophe) should be nonlinear Boolean functions;
2. The quasigroup should not have fixed points;
3. The quasigroup should give as little as possible cells with 100% probability in its differential distribution table and in differential distribution tables obtained from its corresponding $d$–transformation.

## 3 Some yet unpublished properties of the nonlinear substitution part of GAGE

**Proposition 1.** *The quasigroup $*$ used in GAGE is equal to its left conjugate $\backslash_*$.*

*Proof.* Let us recall that

$$x * y = z \iff x \backslash_* z = y.$$

So, from the last equivalence relation, looking at the table (1) we have that $0\backslash_* 0 = 1$, $0\backslash_* 1 = 0$, $0\backslash_* 2 = 3$, $0\backslash_* 3 = 2$ and so on. Completing the finding for all entries of the table for $(Q, \backslash_*)$ gives us

| $\backslash_*$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 1 | 0 | 3 | 2 |
| 1 | 0 | 2 | 1 | 3 |
| 2 | 2 | 3 | 0 | 1 |
| 3 | 3 | 1 | 2 | 0 |

(5)

□

Having the property that the S-box in GAGE and its "inverse" S-box are the same, and having the Theorem 1 it is clear that the following Corollary holds:

**Corollary 1.** *The inverse permutation of the nonlinear D-TRANSFORMATION of GAGE given in Algorithm 1 can be realized with the same S-box.* □

Despite the fact that both the nonlinear permutation and its inverse in GAGE use the same S-box component, their algebraic degrees are different.

**Proposition 2.** *Let $X = \{x_0, x_1, \ldots, x_{b-2}, x_{b-1}\}$ be a state of $b$ bits that is represented as a sequence of pairs i.e. $X \equiv A = \{a_0, \ldots, a_{b/2-1}\}$, where $\{x_{2i}, x_{2i+1}\} \equiv a_i$ for $i \in \{0, b/2-1\}$. Let the two bit leader is $l = \{l_0, l_1\}$. Let we denote the output of the nonlinear permutation i.e. the Algorithm 1 as the state $Y = \{y_0, y_1, \ldots, y_{b-2}, y_{b-1}\}$ i.e. $Y = \text{D-TRANSFORMATION}(l, X)$. Then, the algebraic degree of $y_i$ as Boolean functions of the variables $x_i$ is 2, and the following recurrence relations hold for $Y$:*

$$\begin{aligned} y_{2i} &= x_{2i-2} + x_{2i} + x_{2i-1}x_{2i} + x_{2i-1}x_{2i+1} \\ y_{2i+1} &= 1 + x_{2i-2} + x_{2i-1} + x_{2i-1}x_{2i} + x_{2i+1} + x_{2i-1}x_{2i+1} \end{aligned} \tag{6}$$

*for $i \in \{0, b/2-1\}$, and where $x_{-2} \equiv l_0$ and $x_{-1} \equiv l_1$.*

*Proof.* Let us represent the variables $a_i$, *ldr* and *nextldr* in the Step 5 of Algorithm 1 as follows: the pair $a_i \equiv \{y_{2i}, y_{2i+1}\}$, $ldr = \{x_{2i-2}, x_{2i-1}\}$ and $nextldr = \{x_{2i}, x_{2i+1}\}$. Then the relations (6) follow directly with a simple substitution. □

By a simple replacement is is easy to prove the following Proposition:

**Proposition 3.** *Let $X = \text{D-TRANSFORMATION}^{-1}(l, Y)$ be an inverse of $Y$ that corresponds to the e-transformation given in Definition 2. Then, the algebraic degree of $\{x_{2i}, x_{2i+1}\}$ for $i \in \{0, b/2-1\}$ as Boolean T-functions that depend on variables $y_0, \ldots, y_{2i}$ is $i + 1$.* □

**Definition 3.** *Let us now introduce the following notation:*

| Component | Relations/functions | Comment |
|---|---|---|
| $X = \{x_0, x_1, \ldots, x_{b-2}, x_{b-1}\}$ | | $X$ is a $b$ bit register. |
| $l = \{l_0, l_1\}$ | | $l$ is a 2 bit leader used in the function D-TRANSFORMATION(). |
| $\mathbf{X} = \{l_0, l_1, x_0, x_1, \ldots, x_{b-2}, x_{b-1}\}$ | $\mathbf{X} = l \| X$ | $\mathbf{X}$ is an extended register of $b+2$ bits ($X$ prepended with two bits of $l$). |
| $\mathbf{T1}, \mathbf{T2}, \mathbf{T3}$ | | $\mathbf{T1}, \mathbf{T2}, \mathbf{T3}$ are temporary registers of $b+2$ bits. |
| `DROP`$[\mathbf{X}, 2]$ | | Drops (removes) the first 2 bits of $\mathbf{X}$ returning a register with $b$ bits. |
| $\mathbf{Mask} = \{0, 1, 0, 1, \ldots, 0, 1\}$ | | $\mathbf{Mask}$ is a $b+2$ bits constant register with 0 on every odd position and 1 on every even position. |
| `>>` | $\mathbf{X}$>>$s$ | `SHIFT` to the right the register $\mathbf{X}$ for $s$ positions. |
| `<<` | $\mathbf{X}$<<$s$ | `SHIFT` to the left the register $\mathbf{X}$ for $s$ positions. |
| `XOR` | $\mathbf{X}$ `XOR` $\mathbf{Y}$ | Bitwise addition in $GF(2)$ of the bits of the registers $\mathbf{X}$ and $\mathbf{Y}$. |
| `AND` | $\mathbf{X}$ `AND` $\mathbf{Y}$ | Bitwise multiplication in $GF(2)$ of the bits of the registers $\mathbf{X}$ and $\mathbf{Y}$. |

Table 2: Notation for describing the GAGE's nonlinear D-TRANSFORMATION as `SHIFT`, `XOR` and `AND` operations on $b+2$ bits registers.

**Algorithm 2** Input $X = \{x_0, x_1, \ldots, x_{b-2}, x_{b-1}\}$ and a leader $l = \{l_0, l_1\}$.

```
1: $\mathbf{X} \leftarrow l||X$
2: $\mathbf{T1} \leftarrow (\mathbf{X}$ XOR $(\mathbf{X}$>>1) ) AND $\mathbf{Mask}$
3: $\mathbf{T2} \leftarrow (\mathbf{X}$>>2) AND $\mathbf{Mask}$
4: $\mathbf{T2} \leftarrow \mathbf{T1}$ AND $\mathbf{T2}$
5: $\mathbf{T3} \leftarrow \mathbf{T2}$ XOR $(\mathbf{T2}$<<1)
6: $\mathbf{T2} \leftarrow \Big( \Big( (\mathbf{X}$ XOR $(\mathbf{X}$>>2)) << 1$\Big)$ AND $\mathbf{Mask}\Big)$ >> 1
7: $\mathbf{T1} \leftarrow \Big( (\mathbf{X}$>>3) XOR $((\mathbf{X}$ XOR $(\mathbf{X}$>>2)) AND $\mathbf{Mask})\Big)$ AND $\mathbf{Mask}$
8: $\mathbf{Y} \leftarrow \mathbf{T1}$ XOR $\mathbf{T2}$ XOR $\mathbf{T3}$ XOR $\mathbf{Mask}$
9: $\mathbf{Y} \leftarrow$ DROP$[\mathbf{Y}, 2]$
10: Return $\mathbf{Y}$
```

**Proposition 4.** *The Algorithm 2 is equivalent to D-TRANSFORMATION Algorithm 1.*

*Proof.* With a direct simple replacement of the register variables in Algorithm 2 with their values defined in Definition 3, we get the same Boolean functions for the bits $y_{2i}$ and $y_{2i+1}$ given in the expression (6). $\square$

Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ are three Boolean vectors each with $b+2$ random variables and let $\mathbf{Mask}$ be a constant $(b+2)$-bit vector. Then, the operations $\mathbf{Y} \leftarrow \mathbf{X}$>>$s$, $\mathbf{Y} \leftarrow \mathbf{X}$<<$s$, $\mathbf{Z} \leftarrow \mathbf{X}$ `XOR` $\mathbf{Y}$ and $\mathbf{Y} \leftarrow \mathbf{X}$ `AND` $\mathbf{Mask}$ are linear operations in the vector space $GF(2)^{b+2}$, while the operation $\mathbf{Z} \leftarrow \mathbf{X}$ `AND` $\mathbf{Y}$ is a nonlinear operation.

**Note:** The documentation mentions also the Boolean operation `NOT`, while in Algorithm 2 it is not present. Actually, it is implicitly present with the operation `XOR` $\mathbf{Mask}$ in Step 8 that negates every even-position bit.

**Corollary 2.** *In every round of GAGE there is only one nonlinear vector* `AND` *operation.*

*Proof.* The only nonlinear vector operation in Algorithm 2 is the operation in Step 4. □

## 4 Conclusions

We presented several properties of the GAGE nonlinear layer that were not published in the initial documentation of GAGE. We showed that the GAGE's S-box when represented as a $4 \times 4$ quasigroup, it is a self-conjugate algebraic structure. That means that the same structure can be used to compute the sponge permutation and its inverse.

We showed also that the algebraic degree of the D-TRANSFORMATION operation in the GAGE's permutation is 2, and it differs significantly from the algebraic degree of its inverse which is exactly $b/2$.

We also showed interesting feature of GAGE: Modeled as a sequence of vector operations in the vector space $GF(2)^{b+2}$, it has very low number of nonlinear vector operations - one nonlinear vector `AND` operation per round. In the light of the recent increased interest in cryptographic primitives with low number of `AND` gates, it makes GAGE a potential lightweight cryptographic candidate for Somewhat Homomorphic Encryption, Fully Homomorphic Encryption, MPC protocols, SNARKs, and post-quantum signature schemes.

## References

1. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak sponge function family main document. *Submission to NIST (Round 2)*, 3(30), 2009.
2. Danilo Gligoroski, Hristina Mihajloska and Daniel Otte. "GAGE and InGAGE". NIST LWC Submission page, `https://csrc.nist.gov/Projects/Lightweight-Cryptography/Round-1-Candidates`, March 2019.
3. Danilo Gligoroski, Suzana Andova, and Svein Johan Knapskog. On the importance of the key separation principle for different modes of operation. In *International Conference on Information Security Practice and Experience*, pages 404–418. Springer, 2008.
4. Danilo Gligoroski, Smile Markovski, and Svein Johan Knapskog. The stream cipher Edon80. In *New Stream Cipher Designs*, pages 152–169. Springer, 2008.