Boolean Arithmetic over $\mathbb{F}_2$ from Group Commutators

Marc Joye

Zama
Paris, France
marc@zama.ai

Abstract. This paper studies efficient realizations of arithmetic over the binary field $\mathbb{F}_2$ in nonabelian groups using only intrinsic group operations, namely multiplication and inversion. The constructions rely on commutators to implement Boolean computation within the group structure. Two complementary approaches are presented: a realization of a universal Boolean gate (NAND) and direct realizations of the field operations XOR and AND. These approaches apply to finite nonabelian simple groups and can be implemented using a small number of group operations. Explicit realizations are provided in the alternating groups A_5 and A_6. For the smallest nonabelian simple group A_5, these constructions achieve state-of-the-art efficiency in the number of group operations.

Keywords: Arithmetic over $\mathbb{F}_2$ · Boolean computation · Commutators · Group-based computation · Fully homomorphic encodings

1 Introduction

The expressive power of nonabelian groups as computational media has been recognized for several decades. In algebraic complexity, Barrington's theorem [Bar86] shows that polynomial-size branching programs of width 5 over a fixed nonabelian simple group capture exactly the class NC^1. Earlier work of Maurer and Rhodes [MR65] (see also [KMR66]) already demonstrated that Boolean functions can be realized using finite nonabelian simple groups. These results illustrate that noncommutativity provides a rich computational structure even when only a small set of algebraic operations is available.

This observation raises a natural question: to what extent can arithmetic and Boolean computation be realized directly inside group operations? At first sight this appears unlikely, since a group provides essentially a single binary operation (together with inversion), whereas arithmetic requires both addition and multiplication. Nevertheless, suitable encodings of bits as group elements may allow both operations to be recovered from the intrinsic group structure. In particular, arbitrary Boolean circuits can then be evaluated using only the operations of group multiplication and group inversion.

To describe this phenomenon, it is convenient to adopt the viewpoint of a *fully homomorphic encoding* inside a group. Informally, elements of a ring are represented by group elements in such a way that appropriate combinations of intrinsic group operations implement the desired arithmetic operations. When the message space is $\mathbb{F}_2 = \{0, 1\}$, this corresponds to realizing the Boolean operations XOR and AND on encoded bits using only group operations.

Related work: Early examples of arithmetic over $\mathbb{F}_2$ in groups include the embedding of $(\mathbb{F}_2, +, \cdot)$ into $\mathrm{SL}(3, 2)$ by Ben-Or and Cleve [BC92], the realization inside S_7 studied by Rappe [Rap04], and the construction in the alternating group A_5 given by Khamsemanan, Ostrovsky, and Skeith [KOS16]. Subsequent work of Nuida gave realizations in the symmetric group S_5 [Nui21], with further refinements in [Nui26]. More recently, Guillot et al. [GHKM25] described an explicit realization in the group S_6, building on earlier constructions. Related perspectives appear in works investigating group-theoretic approaches to fully homomorphic encryption [GP05,KOS16,LW17,Nui21,GHKM25].

Our techniques and contributions: The present work studies how Boolean computation and arithmetic over $\mathbb{F}_2$ can be realized efficiently using only intrinsic operations of certain nonabelian groups, namely multiplication and inversion. Our constructions rely on commutators as a basic algebraic primitive. A key structural property supporting this framework is Ore's theorem, proved in [LOST10], which asserts that every element of a finite nonabelian simple group is a commutator. In particular, the commutator map is sufficiently expressive to reach any element of the group, suggesting that commutator-based constructions can serve as building blocks for Boolean computation inside such groups. Related results appear in [KOS16, Lemma 6.3], where it is shown that, for any distinct elements α, β with $\beta \neq e$ in a finite nonabelian simple group G, the element α can be expressed as a product of functions of the form $(x, y) \mapsto \left[g\, x\, g^{-1}, h\, y\, h^{-1}\right]$ evaluated at (β, β). While this result allows arbitrary products of such commutators, the constructions considered here focus on very small products, with the main case being a single commutator (i.e., $k = 1$) and, in one instance, a product of two commutators.

We present two complementary approaches: (i) a commutator-based realization of a universal Boolean gate (NAND), and (ii) direct realizations of the arithmetic operations XOR and AND over $\mathbb{F}_2$. Both approaches rely solely on multiplication and inversion in the underlying group and apply to finite nonabelian simple groups.

We illustrate these approaches through explicit realizations in small alternating groups. For the smallest nonabelian simple group, namely the alternating group A_5, we obtain a commutator-based realization of the NAND gate requiring 8 group operations. This improves the operation count of the recent construction of Nuida [Nui26], which uses compression functions over groups and requires 10 group multiplications to realize a NAND gate. In addition, we obtain a direct commutator-based realization of the AND gate in A_5 requiring 8 group operations, a functionality not explicitly provided in [Nui26]. Its companion XOR gate over $\mathbb{F}_2$ achieves the same operation count as the construction in [Nui26], namely 10 group operations. We also give a different realization of the AND gate that uses only group multiplications, requiring 9 group multiplications in total; its companion XOR gate is then realized by a single group multiplication.

Our framework also yields particularly efficient realizations in larger simple groups. In the group A_6 (and similarly in S_6), we obtain an AND gate requiring only 5 group multiplications and an XOR gate requiring a single group multiplication. This viewpoint provides a conceptual explanation for the encoding over S_6 described in [GHKM25], which is obtained there by modifying an S_7 construction by hand.

Outline of the paper: The remainder of this paper is organized as follows. Section 2 introduces the required group-theoretic notation and the notion of a homomorphic encoding over a group. Section 3 presents a commutator-based realization of a NAND gate, including an explicit construction in A_5. Section 4 gives realizations of the XOR and AND gates over $\mathbb{F}_2$ with parameters in A_5 and A_6. Section 5 concludes the paper.

2 Background and Notation

Unless specified otherwise, all groups considered in this paper are finite, written multiplicatively, and their identity element is denoted by e.

2.1 Elements of Group Theory

We recall several standard notions from group theory; a general reference is [Rob96].

Basic notions: Let G be a group. A subgroup $N \leq G$ is said to be *normal*, written $N \trianglelefteq G$, if it is invariant under conjugation, that is, if $g\, N\, g^{-1} = N$ for all $g \in G$. Equivalently, $g\, n\, g^{-1} \in N$ for all $g \in G$ and $n \in N$. The *center* of G is the subgroup $Z(G) \coloneqq \{z \in G \mid\ z\, g = g\, z \text{ for all } g \in G\}$, consisting of all elements that commute with every element of G. For $g, h \in G$, the (group) *commutator* of g and h is

$$[g, h] \coloneqq g\, h\, g^{-1}\, h^{-1}\ .$$

Thus $[g, h] = e$ if and only if g and h commute.

Finally, a nontrivial group G is called *simple* if its only normal subgroups are $\{e\}$ and G itself.

Permutation groups: For a positive integer n, the *symmetric group* S_n is the group of all permutations of the set $\{1, \dots, n\}$, with group multiplication given by composition. Its elements are bijections $\sigma\colon \{1, \dots, n\} \to \{1, \dots, n\}$, and $\sharp S_n = n!$. Permutations in S_n are written in *cycle notation*. A cycle $(i_1\, i_2\, \dots\, i_k)$ denotes the permutation sending $i_1 \mapsto i_2, i_2 \mapsto i_3, \dots, i_k \mapsto i_1$, and fixing all other elements of $\{1, \dots, n\}$. The identity permutation is denoted by $()$. Disjoint cycles commute, and every permutation can be written uniquely (up to the order of disjoint cycles) as a product of disjoint cycles. A 2-cycle is called a *transposition*.

The *alternating group* A_n is the subgroup of S_n consisting of all even permutations, that is, permutations that can be written as a product of an even number of transpositions. It is a normal subgroup of S_n of index 2, and hence $\sharp A_n = n!/2$. For $n \geq 5$, the group A_n is nonabelian and simple.

2.2 Fully Homomorphic Encodings over Groups

We formalize the notion of realizing arithmetic inside a group.

Definition 1. *Let $(\mathcal{M}, +, \cdot)$ be a commutative ring with 1 and let G be a group. A* fully homomorphic encoding over G *is an injective map*

$$\mathrm{Ecd}\colon \mathcal{M} \longrightarrow G$$

together with efficiently computable operations

$$\mathrm{Add}, \mathrm{Mult}\colon G \times G \to G$$

such that for all $m_1, m_2 \in \mathcal{M}$,

$$\mathrm{Add}\big(\mathrm{Ecd}(m_1), \mathrm{Ecd}(m_2)\big) = \mathrm{Ecd}(m_1 + m_2)$$

and

$$\mathrm{Mult}\big(\mathrm{Ecd}(m_1), \mathrm{Ecd}(m_2)\big) = \mathrm{Ecd}(m_1 \cdot m_2)\ .$$

We illustrate this notion with the realization in $G = A_6$ described in Section 4.4. There the message space is $\mathcal{M} = \mathbb{F}_2$, identified with the set $\{0, 1\}$ equipped with addition $\oplus$ (XOR) and multiplication $\wedge$ (AND). The encoding is given by

$$\mathrm{Ecd}(0) = ()\ , \qquad \mathrm{Ecd}(1) = (1\,2)(3\,4)\ .$$

If $x = \mathrm{Ecd}(\beta)$ and $x' = \mathrm{Ecd}(\beta')$ for $\beta, \beta' \in \mathbb{F}_2$, then

$$\mathrm{Add}(x, x') = \mathrm{XOR}(x, x')\ , \qquad \mathrm{Mult}(x, x') = \mathrm{AND}(x, x')\ ,$$

where XOR is realized by group multiplication and AND by a commutator-based construction.

3 Boolean Computation via a NAND Gate

We denote the Boolean NAND operator by $\overline{\wedge}$, defined by

$$u \mathbin{\overline{\wedge}} v = \neg(u \wedge v) = 1 - u\, v\ , \qquad u, v \in \{0, 1\}\ .$$

Since the NAND gate is functionally complete, it suffices to realize NAND in order to obtain universal Boolean computation. In this section, we present a construction in which NAND is implemented by a *single commutator* in a finite nonabelian simple group, and we give an explicit instantiation in A_5.

3.1 Commutator-Based NAND Gate

Let G be a finite nonabelian simple group. Select two distinct nontrivial elements $z_0, z_1 \in G$ that encode the bits

$$0 \longleftrightarrow z_0 \ , \qquad\qquad 1 \longleftrightarrow z_1 \ .$$

The requirement that both encoding elements be nontrivial is essential: if $z_0 = e$, then any commutator involving z_0 is trivial, so NAND cannot be realized as a single commutator.

Fix elements $g, h \in G$ and define

$$\mathrm{NAND}(x, y) \coloneqq \left[g\, x\, g^{-1}, h\, y\, h^{-1}\right] , \qquad x, y \in \{z_0, z_1\} \ . \tag{1}$$

This map depends only on intrinsic group operations.

Correctness conditions: Let

$$a_i = g\, z_i\, g^{-1} \ , \qquad b_j = h\, z_j\, h^{-1} \ .$$

The map Eq. (1) realizes the Boolean NAND gate under the encoding $0 \leftrightarrow z_0$ and $1 \leftrightarrow z_1$, provided that

$$[a_i, b_j] = z_{i \barwedge j} \ , \qquad i, j \in \{0, 1\} \ .$$

Thus the construction reduces to finding four elements a_0, a_1, b_0, b_1 in prescribed conjugacy classes whose commutator table matches the Boolean NAND table. Ore's theorem guarantees that every element of G is a commutator; hence z_0 and z_1 both occur in the image of the commutator map. The present construction strengthens this requirement by imposing simultaneous constraints on four specific commutators arising from fixed conjugacy classes.

Universality: Once such elements z_0, z_1, g, h are found, the commutator map

$$\mathrm{NAND}(x, y) = \left[g\, x\, g^{-1}, h\, y\, h^{-1}\right]$$

realizes the Boolean NAND operation under the given encoding. Since NAND is functionally complete, arbitrary Boolean functions can then be implemented by composing this operation.

3.2 An Explicit NAND Gate in A_5

We now give a concrete realization that implements a Boolean NAND gate entirely within the group law of $G = A_5$.

Define

$$z_0 = (1\,2\,3\,4\,5) \ , \qquad z_1 = (1\,3\,5\,4\,2) \ , \qquad g = (2\,4)(3\,5) \ , \qquad h = (1\,5\,3) \ .$$

The elements z_0 and z_1 are both 5-cycles and therefore lie in the same conjugacy class of A_5. Moreover, g is an involution ($g^{-1} = g$), while h has order 3 (so $h^{-1} = (1\,3\,5)$).

Let $a_i = g\, z_i\, g^{-1}$ and $b_j = h\, z_j\, h^{-1}$. A direct computation in A_5 yields the commutator table

$[a_i, b_j]$	$j = 0$	$j = 1$
$i = 0$	z_1	z_1
$i = 1$	z_1	z_0

which coincides with the Boolean NAND table under the encoding $0 \leftrightarrow z_0$ and $1 \leftrightarrow z_1$.

Expanding the commutator operator, the NAND gate can be evaluated using 8 group operations in A_5 as

$$\mathrm{NAND}(x, y) = U \cdot V \cdot (V \cdot U)^{-1} \quad \text{where} \quad \begin{cases} U = (2\,4)(3\,5) \cdot x \cdot (2\,4)(3\,5) \\ V = (1\,5\,3) \cdot y \cdot (1\,3\,5) \end{cases} \ .$$

4 XOR and AND Gates over $\mathbb{F}_2$

Let G be a finite nonabelian simple group. We now turn from universal Boolean computation to the direct realization of the arithmetic gates over $\mathbb{F}_2$, namely

$$x \oplus y = x + y \pmod 2 \qquad \text{and} \qquad x \wedge y = x \cdot y \pmod 2.$$

In contrast to the NAND construction of Section 3, it is here natural to adopt the canonical encoding

$$0 \longleftrightarrow e\ , \qquad\qquad 1 \longleftrightarrow z_1\ .$$

so that each bit is represented by an element of the set $\{e, z_1\} \subseteq G$.

4.1 Commutator-Based AND Gate

Fix $g, h \in G$ and define

$$\mathrm{AND}(x, y) \coloneqq \left[g\, x\, g^{-1}, h\, y\, h^{-1}\right]\ , \qquad x, y \in \{z_0, z_1\}\ .$$

Let $a_i = g\, z_i\, g^{-1}$ and $b_j = h\, z_j\, h^{-1}$. Under the encoding $0 \leftrightarrow e$ and $1 \leftrightarrow z_1$, this realizes the Boolean AND gate provided

$$[a_i, b_j] = z_{i \wedge j}\ , \qquad i, j \in \{0, 1\}\ .$$

Because $z_0 = e$, we automatically have $a_0 = b_0 = e$, and therefore

$$[a_0, b_0] = e\ , \qquad\qquad [a_0, b_1] = e\ , \qquad\qquad [a_1, b_0] = e\ ,$$

which match the three zero-outputs of the AND table. The only nontrivial constraint is

$$[a_1, b_1] = z_1\ ,$$

that is, two conjugates of z_1 must have commutator equal to z_1.

Remark 1. Suppose z_1 has order 2. If a_1, b_1 are conjugates of z_1, they are involutions as well, and hence

$$[a_1, b_1] = a_1\, b_1\, a_1^{-1}\, b_1^{-1} = a_1\, b_1\, a_1\, b_1 = (a_1\, b_1)^2\ .$$

Thus the condition $[a_1, b_1] = z_1$ becomes $(a_1\, b_1)^2 = z_1$, so $T \coloneqq a_1\, b_1$ must have order 4. In particular, G must contain two involutions whose product has order 4, and hence a subgroup isomorphic to the dihedral group D_8.

When the condition of Remark 1 is not satisfied, it may still be possible to realize the AND gate by using a product of two commutators. For instance, one may consider constructions of the form

$$\mathrm{AND}(x, y) \coloneqq \left[g_1\, x\, g_1^{-1}, h_1\, y\, h_1^{-1}\right] \cdot \left[g_2\, x\, g_2^{-1}, h_2\, y\, h_2^{-1}\right]\ ,$$

which can satisfy the required Boolean table even when a single commutator does not suffice. Products of such commutator maps also appear in structural expressiveness results such as [KOS16, Lemma 6.3].

4.2 XOR Gate over $\mathbb{F}_2$

We present two XOR realizations, depending on the order of z_1.

XOR via group multiplication: With the same encoding and z_1 of order 2, group multiplication directly implements addition in $\mathbb{F}_2$:

$$z_0 \cdot z_0 = z_0 \ , \qquad z_0 \cdot z_1 = z_1 \cdot z_0 = z_1 \ , \qquad z_1 \cdot z_1 = z_0 \ .$$

Hence

$$\mathrm{XOR}(x, y) \coloneqq x \cdot y$$

realizes the XOR gate.

Commutator-based XOR gate: If G contains a subgroup isomorphic to D_8, then it contains an element $z_1 = [a_1, b_1]$ of order 2, and the above one-multiplication implementation should be preferred.

If G has no involutions, then it contains an element of order 3. Indeed, by the classification of finite simple groups, the only nonabelian finite simple groups whose order is not divisible by 3 are the Suzuki groups $\mathrm{Sz}(2^{2n+1})$, and these contain dihedral subgroups of order 8, hence involutions (so the contrapositive implies that a simple group without involutions must have order divisible by 3).

When $z_0 = e$, a direct commutator-based realization of the form $\mathrm{XOR}(x, y) = \left[g\, x\, g^{-1}, h\, y\, h^{-1}\right]$ is impossible, since XOR requires $[a_0, b_1] = [a_1, b_0] = z_1$, but $a_0 = e$ implies $[a_0, b_1] = e \neq z_1$. If z_1 has order 3, however, XOR can be obtained from the commutator-based AND gate via

$$\mathrm{XOR}(x, y) \coloneqq x \cdot y \cdot \mathrm{AND}(x, y) \ ,$$

since $z_1^3 = e$ and a direct verification shows that the Boolean XOR table is satisfied.

4.3 Fully Homomorphic Encoding of $\mathbb{F}_2$

Assume that $z_0 = e$ and $z_1 \in G$ are chosen so that the above construction realizes the AND gate, and that XOR is implemented either by group multiplication (when z_1 has order 2) or via $\mathrm{XOR}(x, y) = x \cdot y \cdot \mathrm{AND}(x, y)$ when z_1 has order 3.

Then the map

$$\mathrm{Ecd}(0) = e \ , \qquad \mathrm{Ecd}(1) = z_1$$

together with Add = XOR and Mult = AND forms a fully homomorphic encoding of $\mathbb{F}_2$ over G.

4.4 Explicit Realizations

We now give explicit realizations in small simple groups.

A first explicit realization in A_5: In A_5, involutions are double transpositions. The product of two double transpositions has order 1, 2, 3, or 5, never 4. Hence $(a_1 b_1)^2$ cannot be an involution, and therefore no element $z_1 = [a_1, b_1]$ of order 2 arises in this manner (see Remark 1).

The XOR realization via group multiplication therefore does not apply. We give below concrete parameters for a commutator-based implementation in group $G = A_5$.

Define

$$z_0 = () \ , \qquad z_1 = (1\,2\,3) \ , \qquad g = (1\,5\,4) \ , \qquad h = (1\,2)(3\,4) \ .$$

Here z_1 is a 3-cycle and thus has order 3. A direct computation yields $a_1 = g\, z_1\, g^{-1} = (2\,3\,5)$ and $b_1 = h\, z_1\, h^{-1} = (1\,4\,2)$, and in turn

$$[a_1, b_1] = a_1\, b_1\, a_1^{-1}\, b_1^{-1} = (1\,2\,3) = z_1 \ .$$

Because $z_0 = ()$ (that is, $z_0 = e$), the remaining three entries of the AND table are automatically satisfied. Hence

$$\mathrm{AND}(x, y) = \left[g\, x\, g^{-1},\, h\, y\, h^{-1}\right]$$

realizes the Boolean AND gate in A_5 under the encoding $0 \leftrightarrow ()$, $1 \leftrightarrow (1\,2\,3)$. Since z_1 has order 3, XOR is obtained via

$$\mathrm{XOR}(x, y) = x \cdot y \cdot \mathrm{AND}(x, y) \ .$$

Regarding the operation count, as in the NAND realization, the AND gate requires 8 group operations (7 group multiplications and 1 group inversion), and the XOR gate requires 2 additional group multiplications.

Another realization of the AND gate requiring 9 group multiplications in the multiplication-only model (i.e., disallowing explicit inversion) is described below, together with a matching XOR gate implemented by a single group multiplication.

Remark 2. The same obstruction holds in the symmetric group S_5. Its involutions are either transpositions or double transpositions, and in both cases the product of two conjugate involutions never has order 4. Consequently, S_5 likewise admits no realization with $z_1 = [a_1, b_1]$ of order 2, and one must again rely on the commutator-based constructions for AND and XOR.

A second explicit realization in A_5: We now present an explicit realization of the above type in the group $G = A_5$, using a product of two commutators. Define

$$z_0 = () \ , \qquad z_1 = (1\,2)(3\,4) \ , \qquad g_1 = (1\,5)(2\,3) \ , \qquad h_1 = (1\,2)(4\,5) \ ,$$
$$g_2 = (2\,5)(3\,4) \ , \qquad h_2 = g_2\, g_1^{-1}\, h_1 = (1\,4)(3\,5) \ .$$

The choice of h_2 is made so that $g_1^{-1} h_1 = g_2^{-1} h_2 = (1\,3\,2\,5\,4)$, which allows a common factor to be extracted in the evaluation of the two commutators. Writing

$$T = x \cdot (1\,3\,2\,5\,4) \cdot y \ ,$$

the AND gate can then be evaluated as

$$\mathrm{AND}(x, y) = \big((1\,5)(2\,3) \cdot T \cdot (1\,2)(4\,5)\big)^2 \cdot \big((2\,5)(3\,4) \cdot T \cdot (1\,4)(3\,5)\big)^2 \ .$$

A direct verification shows that this map realizes the Boolean AND gate under the encoding $0 \leftrightarrow ()$ and $1 \leftrightarrow (1\,2)(3\,4)$. Since z_1 has order 2, the companion XOR gate is simply given by

$$\mathrm{XOR}(x, y) = x \cdot y \ .$$

This realization uses only group multiplications. More precisely, the AND gate requires 9 group multiplications, while XOR is implemented by a single group multiplication.

An explicit realization in A_6: The situation changes in the group $G = A_6$, which contains a subgroup isomorphic to D_8 (and therefore so does A_n for all $n \geq 6$, since A_6 embeds in A_n as the subgroup fixing $\{7, \ldots, n\}$ pointwise).

Define

$$z_0 = () \ , \qquad z_1 = (1\,2)(3\,4) \ , \qquad g = (2\,3)(5\,6) \ , \qquad h = (1\,5)(2\,6) \ .$$

All elements lie in A_6, and z_1 is an involution. A direct computation yields $g\, z_1\, g^{-1} = (1\,3)(2\,4)$ and $h\, z_1\, h^{-1} = (3\,4)(5\,6)$. These are involutions whose product, $(1\,3)(2\,4) \cdot (3\,4)(5\,6) = (1\,3\,2\,4)(5\,6)$, is a cycle of type $(4, 2)$, and hence has order 4. Consequently,

$$\big\langle (1\,3)(2\,4), (3\,4)(5\,6) \big\rangle \cong D_8 \ .$$

Moreover, one has $\big[g\, z_1\, g^{-1}, h\, z_1\, h^{-1}\big] = [(1\,3)(2\,4), (3\,4)(5\,6)] = (1\,2)(3\,4) = z_1$. Thus the requirement for realizing the Boolean AND gate in A_6 under the encoding $0 \leftrightarrow ()$, $1 \leftrightarrow (1\,2)(3\,4)$ is satisfied.

Since z_1 has order 2, the XOR gate is obtained directly via group multiplication: $\mathrm{XOR}(x, y) = x \cdot y$.

Since z_0 and z_1 are both involutions, so are their conjugates. In turn, the commutator-based AND becomes $\mathrm{AND}(x, y) = g\,x\,g^{-1}\,h\,y\,h^{-1}\,g\,x\,g^{-1}\,h\,y\,h^{-1}$, for $x, y \in \{z_0, z_1\}$. Noting that $g^{-1}\,h = (1\,6\,3\,2\,5)$, the AND gate can be evaluated using 5 group multiplications as

$$\mathrm{AND}(x, y) = \big((2\,3)(5\,6) \cdot x \cdot (1\,6\,3\,2\,5) \cdot y \cdot (1\,5)(2\,6)\big)^2 \ .$$

The XOR gate $\mathrm{XOR}(x, y) = x \cdot y$ requires only a single group multiplication.

5 Conclusion

This paper revisits the problem of realizing arithmetic over $\mathbb{F}_2$ within finite nonabelian simple groups using only intrinsic group operations. Two complementary approaches are presented: one based on realizing a universal Boolean gate (NAND), and one based on direct realizations of the arithmetic operations XOR and AND.

Explicit realizations in the alternating groups A_5 and A_6 illustrate different aspects of these approaches. In A_5, compact commutator-based implementations of Boolean gates are obtained. In A_6, the arithmetic operations admit particularly simple realizations, requiring only a very small number of group multiplications. This latter example also sheds light on the encoding over S_6 that appears in recent work.

References

Bar86. David A. M. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *18th ACM Symposium on Theory of Computing (STOC '86)*, pages 1–5, 1986. doi:10.1145/12130.12131.

BC92. Michael Ben-Or and Richard Cleve. Computing algebraic formulas using a constant number of registers. *SIAM Journal on Computing*, 21(1):54–58, 1992. doi:10.1137/0221006.

GHKM25. Pierre Guillot, Auguste Hoang Duc, Michel Koskas, and Florian Méhats. Introducing GRAFHEN: GRoup-bAsed fully homomorphic encryption without noise. Cryptology ePrint Archive, Paper 2025/1907, 2025. URL: https://ia.cr/2025/1907.

GP05. Dima Grigoriev and Ilia Ponomarenko. On non-abelian homomorphic public-key cryptosystems. *Journal of Mathematical Sciences*, 126(3):1158–1166, 2005. doi:10.1007/s10958-005-0077-3.

KMR66. Kenneth Krohn, W. D. Maurer, and John Rhodes. Realizing complex Boolean functions with simple groups. *Information and Control*, 9:190–195, 1966. doi:10.1016/S0019-9958(66)90229-4.

KOS16. Nirattaya Khamsemanan, Rafail Ostrovsky, and William E. Skeith. On the black-box use of somewhat homomorphic encryption in noninteractive two-party protocol. *SIAM Journal on Discrete Mathematics*, 30(1):266–295, 2016. An earlier version appears in [OS08]. doi:10.1137/110858835.

LOST10. Martin W. Liebeck, E. A. O'Brien, Aner Shalev, and Pham Huu Tiep. The Ore conjecture. *Journal of the European Mathematical Society*, 12(4):939–1008, 2010. doi:10.4171/JEMS/220.

LW17. Jing Li and Licheng Wang. Noiseless fully homomorphic encryption. Cryptology ePrint Archive, Paper 2017/839, 2017. URL: https://ia.cr/2017/839.

MR65. W. D. Maurer and John L. Rhodes. A property of finite simple non-abelian groups. *Proceedings of the American Mathematical Society*, 16:552–554, 1965. doi:10.1090/S0002-9939-1965-0175971-0.

Nui21. Koji Nuida. Towards constructing fully homomorphic encryption without ciphertext noise from group theory. In T. Takagi et al., editors, *International Symposium on Mathematics, Quantum Theory, and Cryptography (MQC 2019)*, volume 33 of *Mathematics for Industry*, pages 57–78. Springer, 2021. doi:10.1007/978-981-15-5191-8_8.

Nui26. Koji Nuida. On compression functions over groups with applications to homomorphic encryption. *Journal of Algebra and Its Applications*, 25(8):2750064:1–2750064:21, 2026. doi:10.1142/S0219498827500642.

OS08. Rafail Ostrovsky and William E. Skeith III. Communication complexity in algebraic two-party protocols. In D. Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 379–396. Springer, 2008. doi:10.1007/978-3-540-85174-5_21.

Rap04. Dörte K. Rappe. *Homomorphic Cryptosystems and their Applications*. PhD thesis, Universität Dortmund, 2004. doi:10.17877/DE290R-15728.

Rob96. Derek J. S. Robinson. *A Course in the Theory of Groups*, volume 80 of *Graduate Texts in Mathematics*. Springer, 2nd edition, 1996. doi:10.1007/978-1-4419-8594-1.

A Compact Boolean Gates in A_5 and S_5

The commutator-based framework developed in the main body of the paper provides a systematic methodology for constructing Boolean gates over nonabelian groups. In this appendix we depart from this general framework and present hand-optimized realizations of standard Boolean gates over A_5 and S_5.

Two encoding families arise. Encodings by elements of distinct orders favor a compact, functionally complete realization of NAND, while the involution encoding $0 \mapsto ()$, $1 \mapsto (1\,2)(3\,4)$ targets compact XOR + AND realizations, with XOR instantiated directly by the group operation.

A.1 Gates over A_5

The $(5,3)$-encoding: We use the Boolean encoding

$$0 \longleftrightarrow z_0 \coloneqq (1\,2\,3\,4\,5)\,, \qquad 1 \longleftrightarrow z_1 \coloneqq (3\,5\,4)\,,$$

in which the two Boolean values are represented by elements of different orders (5 and 3).

NAND gate: With this encoding, a NAND gate is given by

$$\mathrm{NAND}(x,y) = \big((1\,5\,3\,2\,4)\cdot x\cdot y\big)^2\cdot(3\,5\,4)\ .$$

Cost: 4 multiplications.

Additional gates: For the same $(5,3)$-encoding in A_5, the following standard gates are obtained.

$$\begin{aligned}
\mathrm{NOT}(x) &= (2\,4)(3\,5)\cdot x\cdot(1\,5)(2\,3)\,,\\
\mathrm{OR}(x,y) &= \big((2\,3\,4)\cdot x\cdot y\big)^2\cdot y\,,\\
\mathrm{AND}(x,y) &= (2\,4\,5)\cdot\big(x\cdot y\cdot(1\,5\,3\,2\,4)\big)^2\cdot(1\,5)(2\,3)\,,\\
\mathrm{NOR}(x,y) &= (1\,2\,3\,4\,5)\cdot\big(x\cdot(1\,3)(4\,5)\cdot y\cdot(1\,5\,2)\big)^2\,,\\
\mathrm{XOR}(x,y) &= (1\,2\,3\,4\,5)\cdot\big(U^2\cdot(1\,2)(3\,4)\big)^2\,, \qquad U = x\cdot(3\,4\,5)\cdot y\,,\\
\mathrm{XNOR}(x,y) &= (3\,5\,4)\cdot\big(U^2\cdot(2\,4)(3\,5)\big)^2\,, \qquad U = x\cdot(1\,4\,2\,3\,5)\cdot y\ .
\end{aligned}$$

Thus, in the $(5,3)$-encoding, NAND and OR admit realizations of cost 4, AND and NOR of cost 5, and XOR and XNOR of cost 6, while NOT has cost 2.

Remark 3. Swapping the two codewords exchanges every gate with its De Morgan dual at identical cost, yielding a mirror library in which AND and NOR cost 4 while NAND and OR cost 5.

The involution encoding: We now consider the involution encoding

$$0 \longleftrightarrow z_0 \coloneqq ()\,, \qquad 1 \longleftrightarrow z_1 \coloneqq (1\,2)(3\,4)\ .$$

The XOR and AND gates can be realized as

$$\mathrm{XOR}(x,y) = x\cdot y$$

and

$$\mathrm{AND}(x,y) = \big((1\,2\,5)\cdot x\cdot(1\,2\,5\,4\,3)\cdot y\cdot(1\,2)(4\,5)\big)^3\ .$$

Cost: 1 and 6 multiplications, resp.

An alternative cost-six realization of the AND gate arises from a common template that also yields the OR and NOR gates at the same cost. For $c \in A_5$, let

$$W_c \coloneqq \big((1\,3\,2\,4\,5)\cdot y\big)^2 \cdot c \ .$$

Then, with W_c computed once and reused,

$$\begin{aligned}
\mathrm{AND}(x,y) &= W_c \cdot (x \cdot W_c)^2 \ , \qquad c = (1\,4\,3\,5\,2) \ , \\
\mathrm{OR}(x,y) &= W_c \cdot (x \cdot W_c)^2 \ , \qquad c = (2\,3\,5) \ , \\
\mathrm{NOR}(x,y) &= (x \cdot W_c)^3 \ , \qquad c = (2\,4\,5) \ ,
\end{aligned}$$

each at a cost of 6 multiplications.

The NOT and XNOR gates admit particularly simple realizations:

$$\begin{aligned}
\mathrm{NOT}(x) &= (1\,2)(3\,4)\cdot x \ , \\
\mathrm{XNOR}(x,y) &= (1\,2)(3\,4)\cdot x \cdot y \ ,
\end{aligned}$$

with costs of 1 and 2 multiplications. Finally, complementing the output of either AND realization gives

$$\mathrm{NAND}(x,y) = (1\,2)(3\,4)\cdot \mathrm{AND}(x,y) \ ,$$

at a cost of 7 multiplications.

This realization provides a compact XOR/AND pair over A_5: XOR is obtained with a single multiplication, while AND requires only 6 multiplications. This improves upon the product-of-two-commutators construction, which realizes AND at a cost of 9.

A.2 Gates over S_5

The order-$(4,6)$ encoding: Over the symmetric group S_5, we use the Boolean encoding

$$0 \longleftrightarrow z_0 \coloneqq (2\,3\,4\,5) \ , \qquad 1 \longleftrightarrow z_1 \coloneqq (1\,2\,3)(4\,5) \ ,$$

for which NAND is realized as

$$\mathrm{NAND}(x,y) = (2\,4\,3)\cdot W^2 \cdot (1\,3)(2\,4\,5) \ , \qquad W = x \cdot y \ .$$

Cost: 4 multiplications.

Additional gates:

$$\begin{aligned}
\mathrm{NOT}(x) &= (1\,2)\cdot x \cdot (3\,5) \ , \\
\mathrm{AND}(x,y) &= (3\,4)\cdot W^2 \cdot (2\,4\,5) \ , \qquad W = x \cdot y \ , \\
\mathrm{OR}(x,y) &= (4\,5)\cdot V^2 \cdot (1\,2\,3) \ , \qquad V = x \cdot y \cdot (1\,4\,2\,3\,5) \ , \\
\mathrm{NOR}(x,y) &= (2\,3)\cdot V^2 \cdot (3\,4\,5) \ , \qquad V = (1\,4\,2\,3\,5)\cdot x \cdot y \ , \\
\mathrm{XOR}(x,y) &= Z \cdot (2\,5\,4\,3) \cdot Z \ , \qquad Z = S \cdot x \cdot S \ , \quad S = y^2 \ , \\
\mathrm{XNOR}(x,y) &= (R^2)^2 \cdot x \ , \qquad R = x \cdot y \cdot (1\,4) \ .
\end{aligned}$$

NOT costs 2 multiplications, AND costs 4, and OR, NOR, XOR, and XNOR each cost 5. Hence, compared with the $(5,3)$-encoding, AND, XOR, and XNOR are strictly cheaper, OR is more expensive by one multiplication, and the remaining gates are unchanged. The $(5,3)$-encoding therefore remains the better choice when OR-based synthesis dominates or when the ambient group must be simple.