On the Concrete Hardness of LWR with a Power of Two Modulus

Jules Baudrin[1⋆], Rachelle Heim Boissier[2⋆], François-Xavier Standaert[3]

[1] Université Paris-Saclay, UVSQ, CNRS, Laboratoire de mathématiques de Versailles, Versailles, France
[2] Université Libre de Bruxelles, Bruxelles, Belgium
[3] Crypto Group, ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium

Abstract. LWR has been introduced by Banerjee et al. in 2012 as a deterministic variant of LWE. Since then, it has found many applications in the design of symmetric primitives and post-quantum schemes. Despite its deterministic nature, LWR is usually analyzed as LWE, under the (implicit) assumption that no improved attack can take advantage of the additional structure it provides. In this paper, we tackle this assumption in the context of power-of-two moduli and investigate the security of LWR against algebraic attacks in depth. For this purpose, we model its samples as the outputs of a vectorial Boolean function. We first observe that there are corner cases where the state-of-the-art linearisation attack of Arora and Ge does not apply. In contrast, we propose the first LWR-specific cryptanalysis, which applies in any parameter regime as long as the modulus is a power of two. We combine analyses of standard criteria such as the algebraic degree and the number of monomials in the secret with an analysis of the algebraic normal form, that we are able to express exactly in a compact representation by leveraging group action theory. Our results exhibit specificities in the structure of LWR that we are able to exploit. They allow refining and strengthening the understanding of this important problem, and systematically improve over the attack of Arora and Ge. They also put forward new tools for (symmetric) cryptanalysis, which we believe can be of independent interest.

1 Introduction

State of the art. Hard learning problems are of central importance in post-quantum cryptography. For example, the Learning With Errors (LWE) problem [40], and its variants Ring-LWE [38] or Module-LWE [19,35], are used as hardness assumptions in many encryption and signature schemes, including the new standards CRYSTALS-Kyber [42] and CRYSTALS-Dilithium [37].

One potential drawback of LWE is that it generates non-deterministic "noisy" samples. This for example limits its suitability in the design of deterministic (symmetric) primitives. In order to circumvent this problem, Banerjee et al. proposed a deterministic version of LWE, called Learning With Rounding (LWR),

⋆ Work performed in part as post-doctoral researcher at UCLouvain.

where the noise addition is replaced by a deterministic rounding function [10,5]. LWR is known to be asymptotically as hard as LWE and has found applicability in the design of symmetric cryptographic primitives for different applications [14,9,28,2,24,30,47]. It is also the basis of (asymmetric) post-quantum cryptographic algorithms such as the SABER encryption scheme [26].

Despite LWE's solid theoretical foundations [20], the selection of security parameters for lattice-based cryptosystems is for now mostly driven by heuristic estimates, based on the best-known attacks [3]. In this context, a usual assumption (e.g. found in the SABER specifications) is that "*the hardness of (ring or module) LWR can be analyzed as an LWE problem, since there is no known attacks that make use of the additional structure offered by these variants*". Such an assumption is of course questionable given that the aforementioned reductions between LWE and LWR are not tight for the parameters typically considered in practice. The lack of a deeper understanding of the differences and similarities between LWE and LWR, and therefore the lack of LWR-specific cryptanalysis, is actually mentioned as one of the reasons why the NIST favored an LWE-based encryption scheme like Kyber over LWR-based ones like SABER [1].

In this paper, we question the assumption that attacks cannot use the structure offered by certain variants of learning problems, so that their application to LWE and LWR is identical. In particular, we propose the *first LWR-specific (algebraic) attack* and focus on the case of a power-of-two moduli 2^q, which has been considered in the design of PRFs [10,5], possibly specialized for distributed key management [28,30] or leakage applications [28]. Such PRFs are typically instantiated with a large noise level so that they can withstand adversaries obtaining large numbers of samples. Besides, LWR with power-of-two moduli has also been considered with more aggressive parameters, e.g. in constructions enforcing that the noise accumulates [24], where the number of samples is limited [26] or where additional secrets are used in the primitive [47].

Our contribution. As a first result, we show in Section 2 that the main algebraic approach to the cryptanalysis of LWE and LWR, namely the 2011 attack of Arora and Ge [7], and extensions thereof [3,44,43], does not systematically work when the modulus is a power of two. This is in line with the recent and independent work of Chen, Ji & Li [23]. In particular, we exhibit corner case parameter regimes such that this attack is displayed as the best attack by lattice estimators whilst it is not applicable. In the same section, we thus introduce an LWR-specific approach which allows one to mount an algebraic attack *regardless of the parameter regime*. By identifying $\mathbb{Z}_{2^q}$ and $\mathbb{F}_2^q$, we observe that LWR samples are produced by a deterministic *vectorial Boolean function* mapping $2nq$-bit words to p-bit words. Indeed, the LWR function maps $(\boldsymbol{a}, \boldsymbol{x}) \in \mathbb{Z}_{2^q}^n \times \mathbb{Z}_{2^q}^n$ (viewed as $\mathbb{F}_2^{qn} \times \mathbb{F}_2^{qn}$) to $\lfloor \langle \boldsymbol{a}, \boldsymbol{x} \rangle \rfloor_{2^p}$, where $\langle \boldsymbol{a}, \boldsymbol{x} \rangle = \sum_{i=0}^{n-1} a_i \times x_i$ denotes the dot product over $\mathbb{Z}_{2^q}$ and where the rounding function $\lfloor \cdot \rfloor_{2^p}$ removes the $q-p$ least significant bits. The coordinates of this function allow us to build Boolean polynomial equations in the secret. For example, the least significant coordinate function maps $(\boldsymbol{a}, \boldsymbol{x})$ to the LSB of the sample $\lfloor \langle \boldsymbol{a}, \boldsymbol{x} \rangle \rfloor_{2^p}$, *i.e.* the bit of index $q-p$ of $\langle \boldsymbol{a}, \boldsymbol{x} \rangle$. The rest of the paper then investigates the extent to which this approach,

besides working generically for any power-of-two modulus, can be improved in order to be *systematically more efficient* than the Arora-Ge approach.

We begin by showing in Section 3 that, when using equations derived from the LSB of the sample, the Boolean systems obtained have at most the same number of monomials as the systems of Arora and Ge. This is a somewhat surprising result given the number of variables and degree, which we also derive in the same section. It allows us to demonstrate that our new approach does not come with an additional cost compared to Arora and Ge's, and in fact comes with a direct improvement by taking advantage of the practical benefits of working over $\mathbb{F}_2$.

Next, we tackle in Sections 4 and 5 a necessary step for the application of most known-plaintext attack paths relying on Boolean functions[1] and in particular algebraic attacks: the computation of the *Algebraic Normal Form* (ANF), *i.e.*, the polynomial representation over $\mathbb{F}_2$. Whilst our (tight) results on the number of monomials suggest that computing - let alone storing - the ANF is out-of-reach, we leverage group action theory to show that it is in fact possible using a relevant *system of representatives*. This allows us to compute and store a compact representation of the ANF of the least significant bit of LWR samples for 2^{q-p} up to 16. Interestingly, the size of the system of representatives depends *only* on the index of the bit considered and, in particular, not on n. More generally, we show that understanding the ANFs for $n = 2^{q-p}$ allows us to fully understand the ANF for *arbitrary large* n values, thanks to scaling theorems.

Finally, we show in Section 6 that we can identify rank defects and sparsity phenomena, and leverage them in order to improve algebraic attacks. We show that our *system of representatives* together with group action theory allow computing metrics that are relevant for linearisation attacks. More precisely, for 2^{q-p} up to 2^3 and for arbitrary n, we identify an improved linearisation family by grouping monomials that appear or vanish together for any $\boldsymbol{a}$, yielding a direct and systematic improvement of linearisation attacks. Furthermore, for 2^{q-p} up to 2^4 and for arbitrary n, we are able to compute the average number of non-zero terms per equation, allowing to precisely evaluate whether sparse linear algebra techniques allow additional improvements. We summarize our findings together with conclusive remarks and open questions in Section 7.

All in all, we systematically improve the linearisation attack of Arora and Ge, with the improvement varying depending on the exact parameters considered. Hence, our results are also the best attack in any parameters' regime where Arora-Ge was so far considered the best cryptanalysis method (including regimes where it is, in fact, inapplicable). To the best of our knowledge, these improvements do not impact existing schemes which either use large noise levels or limit the adversaries in other ways (e.g., by limiting the number of samples in SABER). Yet, we believe they can contribute to the general understanding of LWR and help refining the security estimates of parameter regimes that could be considered in future use cases. For example, using the linear algebra constant

[1] Attacks on LWR must be known-plaintext as $\boldsymbol{a}$ is drawn uniformly at random.

$\omega = 3$, and for the parameters $2^{q-p} = 8$, $n = 128$ (resp. 256), we improve upon Arora and Ge's attack by 15 bits, obtaining a concrete complexity of $2^{110.8}$ (resp. $2^{134.7}$) binary operations. More examples (including results with a conservative linear algebra constant $\omega = 2$ as used in the lattice estimator of Albrecht et al. [3]) and comparisons with the Arora-Ge attack are discussed in Section 7.1, and our code is available at the link https://github.com/baudrin-j/ac2026_concrete_hardness_lwr_power_of_two_modulus.

Finally, besides offering a complete characterization of an important attack vector and exhibiting exploitable specificities in the structure of LWR, we also introduce tools for the exact modeling of algebraic attacks that may be of independent interest for cryptanalysis. We hope their application to the understanding of the Boolean properties of the dot product can be applied to other primitives. More generally, we also hope group actions which, to the best of our knowledge, have hardly been used before to study ANFs (except for theoretical studies of some Boolean function families [21]), will also find cryptanalytic applications beyond the LWR case in the future.

General notations. In the following, we denote by $\mathbb{F}_2 \coloneqq \{0,1\}$. For any $n \in \mathbb{N}$, $[\![0,n]\!]$ is defined by $[\![0,n]\!] \coloneqq \{0,1,\dots,n\}$ and for any $q \in \mathbb{N}$, the quotient ring $\mathbb{Z}/2^q\mathbb{Z}$ is denoted by $\mathbb{Z}_{2^q}$. The Hamming weight of a binary vector $v \in \mathbb{F}_2^s$ is denoted by $hw(v)$, the support of v is denoted by $\mathrm{Supp}(v)$, and the dot product of two n-long vectors is denoted by $\langle \cdot,\cdot \rangle_n$ or $\langle \cdot,\cdot \rangle$ if n is clear from context.

2 Algebraic attacks with power-of-two moduli

Whilst the algebraic attack of Arora and Ge is the best attack for ranges of parameters with low noise and large numbers of samples according to lattice estimators, we show in this section that the applicability of this technique is non-trivial when considering power-of-two moduli. We begin this section by briefly recalling the definitions of the LWE and LWR problems in their power-of-two moduli versions. We let $n, q \in \mathbb{N}^*$ be positive integers, $\boldsymbol{k} \in \mathbb{Z}_{2^q}^n$ be a secret drawn uniformly at random, and χ be a Gaussian distribution over $\mathbb{Z}_{2^q}$.

Learning With Errors [40]. Learning With Errors (LWE) formalizes the problem of solving a noisy linear system. In its search version, it consists in finding $\boldsymbol{k} \in \mathbb{Z}_{2^q}^n$ given samples from the distribution

$$\mathcal{D}^{\mathsf{LWE}} = \{\ (\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{k} \rangle + e),\ \boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_{2^q}^n,\ e \leftarrow \chi\ \}\,.$$

Decision LWE is the problem of distinguishing $\mathcal{D}^{\mathsf{LWE}}$ from the distribution $\mathcal{D}_0 = \{(\boldsymbol{a}, r) \mid \boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_{2^q}^n, r \xleftarrow{\$} \mathbb{Z}_{2^q}\}$.

Learning With Rounding [10]. Learning With Rounding (LWR) can be viewed as an LWE variant in which the noise is replaced by a deterministic rounding. In the power-of-two modulus case, letting $p \in [\![1, q-1]\!]$, the rounding function

$\lfloor \cdot \rfloor_{2^p} : \mathbb{Z}_{2^q} \longrightarrow \mathbb{Z}_{2^p}$ removes the $q-p$ least significant bits. In its search version, LWR consists in finding $\boldsymbol{k}$ given samples from the distribution

$$\mathcal{D}^{\mathsf{LWR}} = \{ \ (\boldsymbol{a}, \lfloor \langle \boldsymbol{a}, \boldsymbol{k} \rangle \rfloor_{2^p}), \ \boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_{2^q}^n \} \, .$$

Similarly to Decision LWE, Decision LWR aims at distinguishing $\mathcal{D}^{\mathsf{LWR}}$ from $\mathcal{D}_1 = \{(\boldsymbol{a}, r) \mid \boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_{2^q}^n, r \xleftarrow{\$} \mathbb{Z}_{2^p}\}$.[2]

LWE-to-LWR reductions. Naturally, our results do not contradict security reductions like [6,13]. These reductions essentially show that there exist LWE samples such that an LWR sample can be deterministically computed from them. For example, given an LWE sample $(\boldsymbol{x}, y+e)$ where $y = \langle \boldsymbol{a}, \boldsymbol{x} \rangle$, if the error e is known to be in an interval $[0, 2^t - 1]$, any sample such that $y + e \equiv 0 \mod 2^t$ satisfies $\lfloor y + e \rfloor_{2^t} = \lfloor y \rfloor_{2^t}$. Hence, our work directly applies to the LWE samples that can be transformed to LWR samples, by paying in data the inverse of the probability to receive such a sample (2^t in the example above).

2.1 The Arora-Ge linearisation technique

We provide a brief description of the algebraic attack on LWE by Arora and Ge [7]. Assume that the noise is drawn in a finite set E of cardinality $|E|$. In this case, for any LWE sample $(\boldsymbol{a}, s)$ where $s := \langle \boldsymbol{a}, \boldsymbol{k} \rangle + \tilde{e}$, the secret $\boldsymbol{k}$ is a root of the following equation:

$$\prod_{e \in E} (s - \langle \boldsymbol{a}, \boldsymbol{x} \rangle - e) = 0 \, ,$$

where $\langle \boldsymbol{a}, \boldsymbol{x} \rangle$ is the linear equation $\sum_{i \in [\![0, n-1]\!]} a_i x_i$ in the unknowns $x_0, \ldots, x_{n-1}$. If the attacker receives M samples $\{(\boldsymbol{a}^{(i)}, s^{(i)})\}_{i \in [\![1, M]\!]}$, she can recover the secret by solving the polynomial system:

$$\left\{ \prod_{e \in E} (s^{(i)} - \langle \boldsymbol{a}^{(i)}, \boldsymbol{x} \rangle - e) = 0 \right\}_{i \in [\![1, M]\!]} .$$

As observed in the original paper of Arora and Ge, this system can be linearised when M is of the order of the number of monomials. This multivariate polynomial system has n unknowns $x_0, \ldots, x_{n-1}$, and each equation has degree at most $|E|$. The system thus has at most $\binom{n+|E|}{|E|}$ monomials. Straightforward linearisation yields a time complexity of $\binom{n+|E|}{|E|}^{\omega}$, with ω the linear algebra constant (e.g. $\omega = 3$ using Gaussian elimination). Subsequent works have studied the complexity of solving this system using Gröbner bases, but their results apply only when the modulus is either odd [3] or prime [43] - in particular, not a power-of-two.[3]

[2] The only difference with $\mathcal{D}_0$ is that r is drawn in $\mathbb{Z}_{2^p}$ rather than $\mathbb{Z}_{2^q}$.

[3] In a nutshell, these results show that the data complexity, which is of the same order as the number of monomials in the original attack, can be improved at the cost of a higher time complexity.

In the LWE case, the noise is Gaussian. The aforementioned works [7,3,44,43] compute the trade-off between a number of samples large enough so that the system can be solved, and low enough so that the noise belongs to a set E of smaller cardinality with high probability. We do not go into the details here as this does not impact our analysis.

The case of LWR. In the case of LWR, the Arora-Ge attack can be adapted by observing that any LWR sample $(\boldsymbol{a}, s \coloneqq \lfloor \langle \boldsymbol{a}, \boldsymbol{k} \rangle \rfloor_{2^p})$ provides an equation in the secret $\prod_{e \in \mathbb{Z}_{2^{q-p}}} (s||e - \langle \boldsymbol{a}, \boldsymbol{x} \rangle)$ where $s||e$ is the bit concatenation of the p-bit sample s and the $q-p$-bit "error value" e.

2.2 On the applicability of the Arora-Ge technique

As previously mentioned, the lattice estimator of Albrecht et al. [4] cites the attack of Arora and Ge as the best attack for some parameters with low noise, including with a power-of-two modulus (e.g. similar to the parameters of SABER $n = 512$, $q = 2^{13}$, $p = 2^{10}$ but without the existing bound on the number of samples). Yet, we argue in this section that in this case, the Arora-Ge attack is not systematically applicable.

An important observation. Essentially, problems arise whenever the set E is a somewhat large interval. For simplicity, we assume that the cardinality of E is a power of two $|E| \coloneqq 2^c$, but it is straightforward to generalise our observation to an arbitrary cardinality. A trivial yet significant observation is the following. For any $y \in \mathbb{Z}_{2^q}$, $\prod_{e \in E}(y - e)$ is the product of 2^c consecutive elements. Among those elements, 2^{c-1} are even, 2^{c-2} are dividable by 4, and so on. All in all, for any $y \in \mathbb{Z}_{2^q}$, $\prod_{e \in E}(y - e)$ is dividable by

$$\prod_{i=1}^{c} 2^{2^{c-i}} = 2^{2^c - 1}\,.$$

In particular, the function $y \mapsto \prod_{e \in E}(y - e) \mod 2^{2^c - 1}$ is the zero function. Thus, whenever $2^{2^c - 1} \geq 2^q$ the Arora-Ge equations provides *no information on the secret*. In the case of LWR, using the exact same reasoning, the Arora-Ge equations are the trivial equation $0 = 0$ whenever $2^{q-p} - 1 \geq q$. For example, our observation implies that the technique of Arora and Ge is not applicable to the distributed PRF LaKey [30] which has parameters $q - p = 4$ and $q = 12$ so that $2^{q-p} - 1 = 15 > q = 12$.

Lattice estimator errors. In the lattice estimator by Albrecht et al. [4], Arora-Ge is given as the best attack against LWE with parameters

$$(2^q, 2^p) \in \{(2^7, 2^4), (2^6, 2^3), (2^5, 2^2)\}\,.$$

and $n = 512$ (and in fact, most reasonably large n values). This can be verified by entering the following code with $(\texttt{q1}, \texttt{p1}) \in \{(7,4),(6,3),(5,2)\}$ in the estimator:

```
params = LWE.Parameters(n=300, q=2**q1,
    Xs=ND.Uniform(-2**q1-1,0), Xe=ND.Uniform(-2**(q1-p1)-1,0))
LWE.estimate(params)
```

Yet, in these cases, $q - p = 3$ and thus $2^{q-p} - 1 = 7 \geq q$ – the attack does not apply.

Generalizations of the observation. The previous observation stems from the fact that $\mathbb{Z}_{2^q}$ is not an integral domain (unless $q = 1$) and can be generalized to other rings $\mathbb{Z}_s$. In particular, when s admits small prime divisors, such corner cases will occur frequently. As an example if $s = 6$ and if E is made of more than 3 consecutive values, then the Arora-Ge equation will provide no information since a product of three consecutive integers is necessarily dividable by 6.

These observations are in line with the recent and independent work of Chen, Ji & Li [23].

2.3 On the complexity of the Arora-Ge technique

The previous observation implies that in several cases, the Arora-Ge attack simply cannot be used in the case of power-of-two moduli. However, for real-life parameters, the error set E is typically sufficiently small compared to the size of q. For example, in SABER, $q - p = 3$ and $q = 13$ and thus $2^{q-p} - 1 \ll q$. Nevertheless, our observation impacts the concrete complexity of solving the linearised system.

Solving a (homogeneous) linear system over a (finite) ring is not as straightforward as solving it over a field - classical techniques such as Gaussian elimination must be tailored to make up for the fact that not all elements are invertible. In the case of $\mathbb{Z}_{2^q}$, two approaches can be considered. The first one is to solve the system directly in the ring, which costs upwards of $q \cdot n^\omega$ binary operations for a square matrix of size n. In this case, q is of course a generous lower bound on the constant since any arithmetic operation in $\mathbb{Z}_{2^q}$ beyond addition costs strictly more than q binary operations [41,15]. The second one, *Hensel lifting*, allows in general to obtain a constant equal to q. It simply consists in solving the system modulo 2, then lifting the obtained solution(s) to find solution(s) modulo 4, and so on by only solving a Boolean linear system at each step. Details can be found in [29, Section 15.4]. However, our former observation shows that it cannot be directly applied to the equations obtained with Arora-Ge. Indeed, reducing modulo 2^t for any $t \leq 2^c - 1$ yields all solutions in the ring: in this case, one is again solving the trivial equation $0 = 0$. Thus, the smallest ring in which the equations can be considered is $\mathbb{Z}_{2^{2^c}}$. Solving the system thus costs upwards of $n^\omega \cdot 2^c$ binary operations for a square matrix of size n, with $c = q - p$ in the case of LWR.

2.4 A new Boolean approach

In this paper, we propose a natural yet new approach to the cryptanalysis of LWR. Instead of looking at the LWR function $(\boldsymbol{a}, \boldsymbol{x}) \mapsto \lfloor \langle \boldsymbol{a}, \boldsymbol{x} \rangle \rfloor_{2^p}$ as a function from $\mathbb{Z}_{2^q}^n \times \mathbb{Z}_{2^q}^n$ to $\mathbb{Z}_{2^p}$, we can consider this function as mapping two vectors of nq bits to a p-bit vector. By identifying $\mathbb{Z}_{2^q}$ and $\mathbb{F}_2^q$ (*i.e.* identifying integers of $[\![0, 2^q - 1]\!]$ with their binary decompositions), the LWR function can be seen as the following *vectorial Boolean function*:

$$\begin{array}{rcl} (\mathbb{F}_2^q)^n \times (\mathbb{F}_2^q)^n & \longrightarrow & \mathbb{F}_2^p \\ (\boldsymbol{a}, \boldsymbol{x}) & \longmapsto & \lfloor \langle \boldsymbol{a}, \boldsymbol{x} \rangle \rfloor_{2^p} . \end{array}$$

In particular, a function that maps two nq-bit vectors $(\boldsymbol{a}, \boldsymbol{x})$ to any (product of) bit(s) of $\lfloor \langle \boldsymbol{a}, \boldsymbol{x} \rangle \rfloor_{2^p}$ is a Boolean function. We will show in the sequel that this approach allows us to generate Boolean polynomial equations which, contrarily to the Arora-Ge's equation, provide information on the secret *regardless of the parameter regime*. Furthermore, we manage to exploit them in a way that is systematically cheaper.

In the following, the coordinates of $\boldsymbol{x}$ are indicated by a single subscript: $\boldsymbol{x} = (x_0, \ldots, x_{n-1}) \in \mathbb{Z}_{2^q}^n$. Each $x_i \in \mathbb{Z}_{2^q}$ is a q-bit vector. We refer to the bit of index j of x_i by using a second subscript, $x_{i,j} \in \mathbb{F}_2$, and reserve bold characters, e.g. $\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{u}, \boldsymbol{v}$ for vectors of $\mathbb{Z}_{2^q}^n$.

Algebraic Normal Form. It is well-known that any Boolean function $f \colon \mathbb{F}_2^s \to \mathbb{F}_2$ is a multivariate polynomial function in s variables $X_0, \ldots, X_{s-1}$. This polynomial representation is unique modulo the field equations $X_0^2 - X_0, \ldots, X_{s-1}^2 - X_{s-1}$ and known as the *Algebraic Normal Form* (ANF) of f. In the following, any multivariate polynomial will thus *by default* be considered modulo $X_i^2 - X_i$ for all i. Since the LWR function is a vectorial Boolean function, each coordinate or product of coordinates admits an ANF. In particular, it is possible to mount algebraic attacks on LWR by solving a Boolean polynomial system, as long as one can compute and store it.

Generically, for any Boolean function $F \colon \mathbb{Z}_{2^q}^n \times \mathbb{Z}_{2^q}^n \longrightarrow \mathbb{F}_2$, the ANF of F is expressed as $F \colon (\boldsymbol{a}, \boldsymbol{x}) \mapsto \sum_{\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}_{2^q}^n} \alpha_{\boldsymbol{u}, \boldsymbol{v}}(F) \boldsymbol{a}^{\boldsymbol{u}} \boldsymbol{x}^{\boldsymbol{v}}$ where $\alpha_{\boldsymbol{u}, \boldsymbol{v}}(F) \in \mathbb{F}_2$ and

$$\boldsymbol{a}^{\boldsymbol{u}} \boldsymbol{x}^{\boldsymbol{v}} = \prod_{0 \le i < n} a_i^{u_i} x_i^{v_i} = \prod_{0 \le i < n} \prod_{0 \le j < q} a_{i,j}^{u_{i,j}} x_{i,j}^{v_{i,j}} .$$

Since we mostly consider Boolean functions, we emphasize that the exponential notation $x_i^{u_i}$ for integers x_i and u_i *does not* denote the product of x_i with itself u_i times, but the Boolean monomial $\prod_{u_i \neq 0} x_i$. In the sequel, we will use $\times$ to denote a (modular) integer product.

Example 1. Consider variables $\boldsymbol{a} = (a_0, a_1)$ and $\boldsymbol{x} = (x_0, x_1)$ in $\mathbb{Z}_{2^q}^2$ (each a_i, x_i is a q-bit variable). Our notations yield

$$\boldsymbol{a}^{(1,2)} \boldsymbol{x}^{(0,2)} = a_0^1 a_1^2 x_1^2 = a_{0,0} a_{1,1} x_{1,1} . \qquad \triangleright$$

In contexts where we need to refer to the transformation $y \mapsto y^m$ as a *function*, we introduce the slightly redundant notation π_m. This is particularly useful for composition. By definition, when m is of Hamming weight 1, *i.e.* $m = 2^j$, π_{2^j} is nothing more than the projection on the coordinate of index j. For example, $\pi_{2^j} \circ \langle \boldsymbol{a}, \boldsymbol{x} \rangle$ returns the bit of index j of the dot product between $\boldsymbol{a}$ and $\boldsymbol{x}$. Understanding this function's ANF can be viewed as understanding algebraically the recursive propagation of the carries through the modular additions and multiplications involved in a dot product. When $hw(m) > 1$, π_m computes a product of bits (whose ANF will mostly consist of an intricate product of carries).

Important notation. For shorter notation, $F^{m,n}$ stands in the following for the Boolean function $\mathbb{Z}_{2^q}^n \times \mathbb{Z}_{2^q}^n \to \mathbb{F}_2$ $(\boldsymbol{a}, \boldsymbol{x}) \mapsto \pi_m \circ \langle \boldsymbol{a}, \boldsymbol{x} \rangle_n$.

Example 2. Since the rounding removes the $q-p$ least significant bits, the function $F^{2^{q-p},n} = \pi_{2^{q-p}} \circ \langle \cdot, \cdot \rangle$ maps $(\boldsymbol{a}, \boldsymbol{x})$ to the LSB of the *sample* $\lfloor \langle \boldsymbol{a}, \boldsymbol{x} \rangle \rfloor_{2^p}$. $\triangleright$

Known-plaintext setting. Since the LWR function is evaluated on random inputs, LWR should be studied in the known-plaintext setting. When F is a function of public variables $\boldsymbol{a}$ and private variables $\boldsymbol{x}$, it is convenient to view it as a polynomial in $\boldsymbol{x}$ with coefficients in $\mathbb{F}_2[\boldsymbol{a}]$, *i.e.* as:

$$F = \sum_{\boldsymbol{v} \in \mathbb{Z}_{2^q}^n} \alpha_{\boldsymbol{v}}(F) \boldsymbol{x}^{\boldsymbol{v}}, \text{ where } \quad \alpha_{\boldsymbol{v}}(F) \coloneqq \sum_{\boldsymbol{u} \in \mathbb{Z}_{2^q}^n} \alpha_{\boldsymbol{u},\boldsymbol{v}}(F) \boldsymbol{a}^{\boldsymbol{u}} \in \mathbb{F}_2[\boldsymbol{a}]\,. \tag{1}$$

This decomposition can in fact be transformed by an adversary into an equation. Indeed, if one has access to both the ANF of F and the value of a sample $\boldsymbol{a} \in (\mathbb{Z}_{2^q})^n$, then each evaluation $\alpha_{\boldsymbol{v}}(F)(\boldsymbol{a}) \in \mathbb{F}_2$ can be computed and one thus obtains a Boolean equation in the unknowns $\boldsymbol{x}$.

We therefore distinguish between the set of exponents in $\boldsymbol{a}$ and $\boldsymbol{x}$ on the one hand, and the exponents in $\boldsymbol{x}$ on the other, that we respectively define by:

$$\mathrm{Exp}(F) = \left\{ (\boldsymbol{u}, \boldsymbol{v}) \in (\mathbb{N}^n)^2, \alpha_{\boldsymbol{u},\boldsymbol{v}}(F) \neq 0 \right\}, \ \mathrm{Exp}_{\boldsymbol{x}}(F) = \left\{ \boldsymbol{v} \in \mathbb{N}^n, \alpha_{\boldsymbol{v}}(F) \neq 0 \right\}.$$

In the previous definition, we let $\boldsymbol{v}$ (resp. $\boldsymbol{u}$) be a vector of positive integers, and not of integers modulo 2^q. We will often make use of this notation which is unambiguous : indeed, if $u_i \geq 2^q$ for some i then for any $\boldsymbol{x} \in \mathbb{Z}_{2^q}$, $\boldsymbol{x}^{\boldsymbol{v}} = 0$ since there exists a bit of (high) index j such that $u_{i,j} = 1$ and $x_{i,j} = 0$. We also define the *algebraic degree of* F in $\boldsymbol{x}$ and the coefficients in $\boldsymbol{x}$ of F by:

$$\deg_{\boldsymbol{x}}(F) \coloneqq \max_{\boldsymbol{v} | \alpha_{\boldsymbol{v}}(F) \neq 0} hw(\boldsymbol{v}) \quad \text{and} \quad \mathrm{Coefficients}_{\boldsymbol{x}}(F) = \{ \alpha_{\boldsymbol{v}}(F), \boldsymbol{v} \in \mathrm{Exp}_{\boldsymbol{x}}(F) \}\,.$$

Finally, for any fixed value $\boldsymbol{a} \in \mathbb{Z}_{2^q}^n$, we define the specialized function $F_{\boldsymbol{a}}$ by:

$$F_{\boldsymbol{a}} \colon \mathbb{Z}_{2^q}^n \to \mathbb{F}_2 \quad \boldsymbol{x} \mapsto F(\boldsymbol{x}, \boldsymbol{a}).$$

3 A new attack path against LWR

3.1 Challenges and objectives of our approach

Our goal is to explore the Boolean modeling of LWR. Overall, switching from non-linear systems over $\mathbb{Z}_{2^q}$ to systems over $\mathbb{F}_2$ might seem like a bad idea. Indeed, independently of the technique that is used, the complexity of solving a non-linear system is mainly driven by the degree and the number of monomials. Without further investigation, this therefore implies switching from equations in $\mathbb{Z}_{2^q}$ with at most $\binom{n+2^{q-p}}{2^{q-p}}$ monomials in $\mathbb{Z}_{2^q}$ (see Section 2.1) to Boolean equations which could have up to 2^{nq} monomials in $\boldsymbol{x}$, or up to $\sum_{i=0}^{d}\binom{nq}{i}$ monomials if the degree of the equations was known to be bounded by d.

The main result of this section is that our Boolean point of view *does not* come with an increased number of monomials nor a higher complexity. On the contrary, we prove that the number of monomials in the ANF of the bit of index 2^{q-p} of $(\boldsymbol{a},\boldsymbol{x}) \mapsto \langle \boldsymbol{a},\boldsymbol{x}\rangle$, that is, the least significant bit available to an adversary, admits at most $\binom{n+2^{q-p}}{2^{q-p}}$ Boolean monomials, *i.e.* the exact same number of monomials as $\mathbb{Z}_{2^q}$-equations of degree at most 2^{q-p} in n variables. The least significant bits of the secret can thus be recovered in about $\binom{n+2^{q-p}}{2^{q-p}}^{\omega}$ *binary* operations, that is, for a cost at least 2^{q-p} times lower than the cost of solving Arora-Ge equations as discussed in Section 2.3. We further show in Section 3.3 that the rest of the secret can also be recovered for a negligible additional cost.

However, this improvement is obtained *only if* the ANF of each coefficient in $\boldsymbol{x}$, *i.e.* $\alpha_{\boldsymbol{v}}(F^{m,n})$, is known. These ANFs are indeed necessary to derive an equation in $\boldsymbol{x}$ from each sample $(\boldsymbol{a}, s_{\boldsymbol{a}})$. The coefficients $\alpha_{\boldsymbol{v}}(F^{m,n})$ are *a priori* random-looking polynomials of $\mathbb{F}_2[\boldsymbol{a}]$ and it is rare to have information beyond e.g. a bound on the algebraic degree of symmetric primitives [17,32]. As detailed in Section 3.4, both the time and memory complexities to compute the ANF are typically exceptionally high. The goal of Section 5 will be to prove that this usually prohibitive step can be efficiently achieved in the case of the LWR function, by taking advantage of symmetries described in Section 4.

For now, we prove in Section 3.2 the announced bound on the number of monomials in the Boolean coordinates of the LWR function. We then discuss in Section 3.3 the final step to mount a full key-recovery for a marginal additional cost (under the assumption that the ANF can be computed).

3.2 Number of terms of the Boolean coordinates of the dot product

Recall that $F^{m,n}$ stands for the Boolean function $\mathbb{Z}_{2^q}^n \times \mathbb{Z}_{2^q}^n \to \mathbb{F}_2$ $(\boldsymbol{a},\boldsymbol{x}) \mapsto \pi_m \circ \langle \boldsymbol{a},\boldsymbol{x}\rangle_n$.

ANF of modular addition. To study the ANF of (product of) bits of the dot product, we rely on the ANF of modular addition, which has been previously investigated by Braeken and Semaev [18]. We denote by SUM^n the modular addition of n elements, namely: $\mathsf{SUM}^n : \mathbb{Z}_{2^q}^n \to \mathbb{Z}_{2^q}$ $\boldsymbol{x} \mapsto \sum_{i=0}^{n-1} x_i$. Equivalently, $\mathsf{SUM}^n(\boldsymbol{x}) = \langle \mathbf{1}, \boldsymbol{x}\rangle_n$, where $\mathbf{1}$ is the all-1 vector of length n. For any

$n, m \in \mathbb{N}$, the set of n-long vectors that sum to m is denoted by $S_m^n := \left\{ \boldsymbol{c} \in \mathbb{N}^n \mid \sum_{i=0}^{n-1} c_i = m \right\}$.

Theorem 1 (ANF of modular addition [18, Theorem 1]). *For any $m \in \mathbb{N}$, and any $x, y \in \mathbb{Z}_{2^q}$, it holds that: $(x+y)^m = \sum_{c=0}^{m} x^{m-c} y^c$. More generally for any $n \in \mathbb{N}^*$, it holds that:*

$$\pi_m \circ \mathsf{SUM}^n(\boldsymbol{x}) = \sum_{\boldsymbol{c} \in S_m^n} \boldsymbol{x}^{\boldsymbol{c}} \,. \tag{2}$$

We refer to the original paper for the proof [18, Theorem 1] or to an alternative one based on monomial prediction [34, Theorem 2]. It should be noted that the ANF of $\pi_m \circ \mathsf{SUM}^n$ does not depend on the size q of the integers. Indeed, if $\boldsymbol{c} \in S_m^n$, then each c_i satisfies $c_i \leq m$ and is therefore of size less than $\lfloor \log_2(m) \rfloor + 1$. This implies that only bits of index lower than or equal to $\lfloor \log_2(m) \rfloor + 1$ are actually involved. This essentially captures the fact that the i-th carry depends on the $(i-1)$ least significant bits of each term – a natural consequence of the triangular nature of modular addition. Next, we derive the exact degree and the set of exponents of the functions $\pi_m \circ \mathsf{SUM}^n$ from the previous theorem.

Proposition 1. *Let $n, m \in \mathbb{N}^*$. The following statements hold.*

(i) Exponents. $\mathrm{Exp}(\pi_m \circ \mathsf{SUM}^n) = S_m^n$ *and* $|\mathrm{Exp}(\pi_m \circ \mathsf{SUM}^n)| = \binom{n+m-1}{m}$.
(ii) Degree. $\deg(\pi_m \circ \mathsf{SUM}^n) \leq m$. *If $m \leq n$, then* $\deg(\pi_m \circ \mathsf{SUM}^n) = m$.

Proof. **(i).** This is an immediate rewriting of Theorem 1. The cardinality of S_m^n is equal to the number of monomials of degree m in n variables (in $R[x_1, \ldots, x_n]$ with R an arbitrary ring), which is well-known (see e.g. [7, Section 3]) to be equal to $\binom{n+m-1}{m}$ (as a consequence of the "stars and bars" reasoning and Pascal's triangle formula).

(ii). For any $\boldsymbol{c} \in S_m^n$, $\deg(\boldsymbol{x}^{\boldsymbol{c}}) = \sum_{i=0}^{n-1} hw(c_i) \leq \sum_{i=0}^{n-1} c_i = m$. When $m \leq n$, for any $\mathcal{I} \subset \{0, \ldots, n-1\}$ of cardinality $|\mathcal{I}| = m$, the vector $\boldsymbol{c} = (c_0, \ldots, c_{n-1})$ such that $c_i = 1$ if $i \in \mathcal{I}$ and $c_i = 0$ otherwise belongs to S_m^n and corresponds to the monomial $\boldsymbol{x}^{\boldsymbol{c}} = \prod_{i \in \mathcal{I}} x_i^{c_i} = \prod_{i \in \mathcal{I}} x_{i,0}$ which has degree m. □

ANF of modular multiplication. We now derive similar bounds for $F^{m,n}$. To do so, we also rely on the ANF of modular multiplication as described by Braeken and Semaev [18].

Theorem 2 (ANF of modular multiplication [18, Theorem 2]). *For any $m \in \mathbb{N}^*$, and any $a, x \in \mathbb{Z}_{2^q}$, it holds that:*

$$(a \times x)^m = \sum_{\substack{\boldsymbol{e} \in S_m^q \\ \forall\ i \in [\![0, q-1]\!],\ 2^i \mid e_i}} a^{g(\boldsymbol{e})} x^{h(\boldsymbol{e})} \,,$$

with for any j, $\pi_{2^j} \circ g(\boldsymbol{e}) := \bigvee_{i=0}^{q-1} \pi_{2^i}\left(\frac{e_j}{2^j}\right)$ and $h(\boldsymbol{e}) := \bigvee_{i=0}^{q-1} \frac{e_i}{2^i}$.

As pointed out in [18, Remark 2], this formula is not reduced because the function $\boldsymbol{e} \mapsto (g(\boldsymbol{e}), h(\boldsymbol{e}))$ is not injective: some terms $a^u x^v$ may appear an even number of times and cancel out.

Example 3 (Extracted from [18, Example 2]). Assume $m = 5$. The ordered partitions $\boldsymbol{e}$ of 5 such that $2^i | e_i$ are $(5,0,0)$, $(1,4,0)$, $(1,0,4)$, $(3,2,0)$. Dividing e_i by 2^i, we obtain the vectors $(5,0,0)$, $(1,2,0)$, $(1,0,1)$, $(3,1,0)$. This yields:

$$(a_0 \times x_0)^5 = a_0^1 x_0^5 + a_0^3 x_0^3 + a_0^5 x_0^1 + a_0^3 x_0^3 = a_0^1 x_0^5 + a_0^5 x_0^1 = a_{0,0} x_{0,0} x_{0,2} + a_{0,0} a_{0,2} x_{0,0}.$$

▷

Corollary 1. *For any $m \in \mathbb{N}^*$, it holds that* $\mathrm{Exp}_x\big((a,x) \mapsto (a \times x)^m\big) \subset [\![1, m]\!]$.

Proof. According to Theorem 2, it is sufficient to prove that for any $\boldsymbol{e} \in S_m^q$, $h(\boldsymbol{e}) \subset [\![1, m]\!]$, *i.e.* $1 \leq h(\boldsymbol{e}) \leq m$. By definition, $\sum_{i=0}^{q-1} e_i = m > 1$, so $\boldsymbol{e} \neq (0, \ldots, 0)$ and thus $h(\boldsymbol{e}) \neq 0$. Furthermore it holds that:

$$h(\boldsymbol{e}) = \bigvee_{i=0}^{q-1} \frac{e_i}{2^i} \leq \sum_{i=0}^{q-1} \frac{e_i}{2^i} \leq \sum_{i=0}^{q-1} e_i = m\,. \qquad \square$$

As for modular addition, this corollary captures the 'triangular nature'[4] of the ANF of modular multiplication, easily observed in the classical shift-and-add multiplication algorithm in base 2. Indeed, the bit of index j of $a \times x$ only depends linearly on x_j, non-linearly on bits of index $i < j$ (because of the carries) and the constant term 1 does not appear (as $a \times 0 = 0$). This corresponds to the $1 + 2^j - 1 = 2^j$ terms of $(a \times x)^{2^j}$.

Finally, we obtain the announced bound on the number of terms in $\boldsymbol{x}$ in the ANF of (products of) bits of the dot product.

Theorem 3 (Exponents and degree of $F^{m,n}$). *For any $m, n \in \mathbb{N}^*$, the following statements hold.*

(i) Exponents. $\mathrm{Exp}_{\boldsymbol{x}}(F^{m,n}) \subset \bigcup_{i=1}^{m} S_i^n$ *and thus* $|\mathrm{Exp}_{\boldsymbol{x}}(F^{m,n})| \leq \binom{n+m}{m} - 1$.
(ii) Degree. $\deg(F^{m,n}) \leq m$. *If $m \leq n$, then* $\deg_{\boldsymbol{x}}(F^{m,n}) = m$.
(iii) Variables. $F^{m,n}$ *depends on at most* $n \cdot (\lfloor \log_2(m) \rfloor + 1)$ *variables of* $\boldsymbol{x}$.

Proof. By Theorem 1, $F^{m,n} = \sum_{\boldsymbol{c} \in S_m^n} \prod_{i=0}^{n-1} (a_i \times x_i)^{c_i}$. For any $\boldsymbol{c} \in S_m^n$ and any i, by Corollary 1, it holds that $\mathrm{Exp}_{x_i}\big((a_i \times x_i)^{c_i}\big) \subset [\![1, c_i]\!]$ Therefore, the monomials $\boldsymbol{x}^{\boldsymbol{u}}$ in the ANF of $F^{m,n}$ necessarily satisfy $\sum_{i=0}^{n-1} u_i \leq \sum_{i=0}^{n-1} c_i = m$ since $\boldsymbol{c} \in S_m^n$ and $\boldsymbol{u} \neq (0, \ldots, 0)$. This directly implies **(i)**, of which **(iii)** is a direct consequence since the binary decomposition of m is of size $\lfloor \log_2(m) \rfloor + 1$. The algebraic degree of such a monomial is equal to $\sum_{i=1}^{n} hw(u_i) \leq \sum_{i=1}^{n} u_i \leq m$, which implies **(ii)**. It holds that $\deg_{\boldsymbol{x}}(F^{m,n}) = \max_{\boldsymbol{a} \in \mathbb{Z}_{2^q}^n} \deg(F_{\boldsymbol{a}}^{m,n})$. As shown before, if $\boldsymbol{a} = \boldsymbol{1}$ then $F_{\boldsymbol{a}}^{m,n} = \langle \boldsymbol{1}, \boldsymbol{x} \rangle = \pi_m \circ \mathsf{SUM}^n(\boldsymbol{x})$. By Proposition 1, when $m \leq n$, it then holds that $\deg(F_{\boldsymbol{a}}^{m,n}) = m$, which proves that $\deg_{\boldsymbol{x}}(F^{m,n}) = m$ in that case. $\square$

[4] This triangularity property being also shared by modular addition, it also extends to any polynomial function from $\mathbb{Z}_{2^q}$ to itself, see e.g. [39].

Remark 1. Our experimental results suggest that the inclusion of Theorem 3, $\mathrm{Exp}_{\boldsymbol{x}}(F^{m,n}) \subset \bigcup_{i=1}^{m} S_i^n$, is in fact an equality whenever m is a power of two. This has been verified for m up to 16 and for arbitrary large n, thanks to the dedicated ANF analysis described in Sections 4 and 5. The proof of this fact is an interesting open question.

Since $\boldsymbol{a}$ and $\boldsymbol{x}$ play symmetric roles in the dot product, the results of Theorem 3 apply to the degree, set of exponents and number of variables "in $\boldsymbol{a}$". As for modular addition and multiplication, Item **(iii)** above states that the ANF of $F^{m,n}$ (*i.e.*, bits of the dot product) *does not* depend on the size q of the input integers. This is again due to the triangular nature of modular operations.

3.3 A solving strategy based on Boolean equations

As announced, applying Theorem 3 to the least significant bit available to an adversary, *i.e.* the bit of index $q-p$ (or equivalently $m = 2^{q-p}$),[5] indicates that the corresponding Boolean equations will have up to $\binom{n+2^{q-p}}{2^{q-p}}$ monomials that only depends in $q-p+1$ least significant variables $x_{i,j}$ $(0 \leq j \leq q-p)$ for each i. This implies that these bits can be recovered by a straightforward linearisation at the cost of about $\binom{n+2^{q-p}}{2^{q-p}}$ in data, $\binom{n+2^{q-p}}{2^{q-p}}^2$ in memory and $\binom{n+2^{q-p}}{2^{q-p}}^\omega$ in time.

While the entire secret is not yet recovered, the remaining bits of index $i \geq q-p+1$ can be recovered for a negligible cost. Indeed, for any ℓ, $F^{2^\ell,n}(\boldsymbol{a},\boldsymbol{x})$ can be expressed as $F^{2^\ell,n}(\boldsymbol{a},\boldsymbol{x}) = \sum_{i=0}^{n-1} a_{i,0}x_{i,\ell} + c_\ell$ where the carry c_ℓ only depends on bits of index strictly lower than ℓ of $\boldsymbol{a}$ and $\boldsymbol{x}$. Since the lower bits of $\boldsymbol{x}$ are recovered and the ones of $\boldsymbol{a}$ are known, this carry value can be computed and this provides a Boolean linear equation in n unknowns, $x_{0,2^\ell}, \ldots, x_{n-1,2^\ell}$. The most significant bits of the secret can thus be recovered one after the other. The extra cost corresponds to solving $p-1$ *linear* systems in n unknowns over $\mathbb{F}_2$, which is negligible before the cost of the initial non-linear system solving, both in time and memory. The data cost is not increased since we leverage the so-far unused bits of the samples that were collected in the first step.

3.4 Towards the computation of the full ANF

The benefits of this new attack path being clearly established, it remains to gauge whether or not it can be efficiently implemented. In contrast with the Arora-Ge equations where for each sample, the coefficients in $\boldsymbol{a}$ can be precomputed (using multinomial expansions) and quickly evaluated to obtain an equation in the unknowns, this is clearly less trivial with our Boolean equations.

To the best of out knowledge, the only known generic method to compute the ANF of a Boolean function in r variables is the *binary Möbius transform*. It has complexity $r2^{r-1}$ (see e.g. [22]), independently of the degree or the number

[5] By Theorem 3, since the number of terms, degree and number of variables grow with m, this seems to be the most reasonable choice.

of monomials, and is only applicable if the look-up table can be fully computed and stored. This rules it out in almost all symmetric cryptographic contexts, including ours, since primitives are designed with a large enough codebook.

When the primitive can be algebraically described in a simple way, for instance as the composition of simple components (such as adder and multiplier logical circuits in the case of LWR), it is theoretically possible to directly compute the ANF as the composition of the ANF of each sub-functions, or to use alternative methods[6] such as *monomial prediction* or *perfect division property* [45,16,33,32,12]. However, all these methods suffer from the same bottlenecks and become quickly impractical.

We show in the sequel that computing the ANF of $F^{m,n}$ – more precisely, a relevant *system of representatives* of the ANF – is in fact possible. To solve both memory and computational complexity issues and investigate the ANF for very large values of n, we rely on a further understanding of the structure of $F^{m,n}$. In Section 4, we show that $F^{m,n}$ inherits strong symmetries from commutativity of modular addition. We exploit those results to:

- reduce the study of the ANF of $F^{m,n}$ to the study of a relevant *system of representatives* which captures the *full ANF* up to permutations, and to
- reduce the study of $F^{m,n}$ for all values $m \leq n$ to the study of $F^{m,m}$.

The effective computation of the system of representatives of the ANF is described in Section 5.

Finally, we provide an in-depth study of the ANF in Section 6. By exploiting further the symmetries mentioned above, we compute additional metrics such as an improved upper bound on the dimension of the subspace spanned by the family of functions $\langle F^{m,n}_{\boldsymbol{a}}, \boldsymbol{a} \in (\mathbb{Z}_{2^q})^n \rangle$ and the average number of non-zero terms. As described in Section 7.1, this study allows us to systematically improve the linearisation attack of Arora and Ge.

4 Formalising symmetries in the ANF of $F^{m,n}$

In this section, we formalize the significant implications of two simple observations on the dot product.

1. By commutativity of modular addition, each pair of variables (a_i, x_i) plays an identical role. This creates strong symmetries in the ANF of (products of) bits of the dot product that are detailed in Lemma 2 and Theorem 4.
2. Furthermore, the dot product of n-long vectors can be seen as a dot product of n'-long vectors where $n \leq n'$ and where the last $n' - n$ coordinates are zeros. This implies that the ANF of $F^{m,n}$ is "included" in the ANF of $F^{m,n'}$. Theorem 5 below shows a converse property: it is actually sufficient to understand $F^{m,n}$ to fully understand $F^{m,n'}$.

To achieve such goals, we rely on the formalism of *group actions*. Standard definitions and results on group actions can e.g. be found in [27, Chapter 4].

[6] While these formalisms are well-suited to use generic MILP, SAT or CP solvers, estimating their memory and computational complexity remains also an open problem.

4.1 Action of $\mathfrak{S}_n$ on Boolean functions

We denote by $\mathfrak{S}_n$ the symmetric group over the set $\{0, \dots, n-1\}$ and the composition of two permutations $\sigma, \tau \in \mathfrak{S}_n$ by $\sigma \circ \tau$ or $\sigma\tau$.

Action on vectors. Let E be a set. For any vector $\boldsymbol{u} \in E^n$ and any permutation $\sigma \in \mathfrak{S}_n$, we denote by $\sigma \cdot \boldsymbol{u}$, the permuted copy of $\boldsymbol{u}$ with respect to σ, namely:

$$\sigma \cdot \boldsymbol{u} \coloneqq (u_{\sigma^{-1}(0)}, \dots, u_{\sigma^{-1}(n-1)}).$$

This corresponds to an action of the group $\mathfrak{S}_n$ on the set E^n. Similarly, $\mathfrak{S}_n$ acts on pairs of vectors and we denote by $\sigma \cdot (\boldsymbol{u}, \boldsymbol{v})$ the pair $(\sigma \cdot \boldsymbol{u}, \sigma \cdot \boldsymbol{v})$ for any n-long vectors $\boldsymbol{u}, \boldsymbol{v}$ and any permutation $\sigma \in \mathfrak{S}_n$.

Action on monomials and functions. The group $\mathfrak{S}_n$ also acts on monomials by permuting input variables. This is clearly equivalent to applying the inverse permutation to the vectors of exponents and we denote:

$$\sigma \cdot (\boldsymbol{x}^{\boldsymbol{v}}) \coloneqq \boldsymbol{x}^{\sigma^{-1} \cdot \boldsymbol{v}} = (\sigma \cdot \boldsymbol{x})^{\boldsymbol{v}} \text{ and } \sigma \cdot (\boldsymbol{a}^{\boldsymbol{u}} \boldsymbol{x}^{\boldsymbol{v}}) \coloneqq \boldsymbol{a}^{\sigma^{-1} \cdot \boldsymbol{u}} \boldsymbol{x}^{\sigma^{-1} \cdot \boldsymbol{v}} = (\sigma \cdot \boldsymbol{a})^{\boldsymbol{u}} (\sigma \cdot \boldsymbol{x})^{\boldsymbol{v}}\,.$$

More generally, for any Boolean function f over $\mathbb{Z}_{2^q}^n$ we define $\sigma \cdot f$ as:

$$\sigma \cdot f \coloneqq f(\sigma \cdot \boldsymbol{x}) = \sum_{\boldsymbol{u} \in \mathbb{N}^n} \alpha_{\boldsymbol{u}} (\sigma \cdot \boldsymbol{x})^{\boldsymbol{u}} = \sum_{\boldsymbol{u} \in \mathbb{N}^n} \alpha_{\boldsymbol{u}} \boldsymbol{x}^{\sigma^{-1} \cdot \boldsymbol{u}} = \sum_{\boldsymbol{u} \in \mathbb{N}^n} \alpha_{\sigma \cdot \boldsymbol{u}} \boldsymbol{x}^{\boldsymbol{u}}\,; \tag{3}$$

For a function F over $\mathbb{Z}_{2^q}^n \times \mathbb{Z}_{2^q}^n$, $\sigma \cdot F$ can be seen as:

$$\sigma \cdot F = \sum_{\boldsymbol{u}, \boldsymbol{v} \in \mathbb{N}^n} \alpha_{\sigma \cdot (\boldsymbol{u}, \boldsymbol{v})} \boldsymbol{a}^{\boldsymbol{u}} \boldsymbol{x}^{\boldsymbol{v}} \in \mathbb{F}_2[\boldsymbol{a}, \boldsymbol{x}], \quad \text{or as} \tag{4}$$

$$\sigma \cdot F = \sum_{\boldsymbol{v} \in \mathbb{N}^n} (\sigma \cdot \alpha_{\boldsymbol{v}})\, (\sigma \cdot \boldsymbol{x})^{\boldsymbol{v}} = \sum_{\boldsymbol{v} \in \mathbb{N}^n} (\sigma \cdot \alpha_{\sigma \cdot \boldsymbol{v}}) \boldsymbol{x}^{\boldsymbol{v}} \in \mathbb{F}_2[\boldsymbol{a}][\boldsymbol{x}]\,. \tag{5}$$

Example 4. Let $\boldsymbol{x} = (x_0, x_1)$, let σ be the 2-cycle $\sigma = (0\ 1)$ and let $f \colon \mathbb{Z}_{2^1}^2 \to \mathbb{F}_2$ be defined by $f(\boldsymbol{x}) = x_0^3 + x_1^2 = x_{0,0}x_{0,1} + x_{1,1}$. Then $(\sigma \cdot f)(\boldsymbol{x}) = x_1^3 + x_0^2 = x_{1,0}x_{1,1} + x_{0,1}$. Next, let $F : \mathbb{Z}_{2^1}^2 \times \mathbb{Z}_{2^1}^2 \to \mathbb{F}_2$ be defined by $F(\boldsymbol{a}, \boldsymbol{x}) = x_0^3 + x_1^2 + a_1$. Then $(\sigma \cdot F)(\boldsymbol{a}, \boldsymbol{x}) = x_1^3 + x_0^2 + a_0$. ▷

For any of the actions of $\mathfrak{S}_n$ defined above on a set E (E being a set of vectors, monomials or polynomials), the *orbit* and *stabiliser* of $e \in E$ are defined by:

$$\mathrm{Orb}(e) \coloneqq \{\sigma \cdot e, \sigma \in \mathfrak{S}_n\} \subset E \quad \text{and} \quad \mathrm{Stab}(e) \coloneqq \{\sigma \in \mathfrak{S}_n \text{ s.t. } \sigma \cdot e = e\} \subset \mathfrak{S}_n.$$

Note that the set of orbits $\{\mathrm{Orb}(e) \mid e \in E\}$ is a partition of E. Therefore, we denote by $\sim$ the equivalence relation defined by $e \sim e'$ if and only if $e' \in \mathrm{Orb}(e)$.

Invariance under group actions. Let $\mathbb{G}$ be a subgroup of $\mathfrak{S}_n$. A Boolean function f over $\mathbb{Z}_{2^q}^n$, is said to be $\mathbb{G}$-*invariant* if $\sigma \cdot f = f$ for any $\sigma \in \mathbb{G}$ or equivalently if $\mathbb{G} \subset \mathrm{Stab}(f)$. The same holds for functions over $\mathbb{Z}_{2^q}^n \times \mathbb{Z}_{2^q}^n$. By identifying the coefficients of f (resp. F) with the ones of $\sigma \cdot f$ (resp. $\sigma \cdot F$) outlined in Eqs. (3) to (5), we obtain the following characterizations of $\mathbb{G}$-invariance.

Lemma 1. *Let $f\colon \mathbb{Z}_{2^q}^n \to \mathbb{F}_2$ be a Boolean function and $\mathbb{G}$ be a subgroup of $\mathfrak{S}_n$. Then f is $\mathbb{G}$-invariant if and only if:*

$$\forall \sigma \in \mathbb{G}, \forall \boldsymbol{u} \in \mathbb{N}^n, \quad \alpha_{\sigma \cdot \boldsymbol{u}}(f) = \alpha_{\boldsymbol{u}}(f).$$

Lemma 2. *Let $F\colon \mathbb{Z}_{2^q}^n \times \mathbb{Z}_{2^q}^n \to \mathbb{F}_2$ be a Boolean function of $\boldsymbol{a}$ and $\boldsymbol{x}$ and $\mathbb{G}$ be a subgroup of $\mathfrak{S}_n$. The following statements are equivalent:*

(i) *F is $\mathbb{G}$-invariant.*
(ii) $\forall \sigma \in \mathbb{G}, \forall \boldsymbol{u}, \boldsymbol{v} \in \mathbb{N}^n, \quad \alpha_{\sigma \cdot (\boldsymbol{u}, \boldsymbol{v})}(F) = \alpha_{\boldsymbol{u}, \boldsymbol{v}}(F).$
(iii) $\forall \sigma \in \mathbb{G}, \forall \boldsymbol{v} \in \mathbb{N}^n, \quad \sigma \cdot \alpha_{\boldsymbol{v}}(F) = \alpha_{\sigma^{-1} \cdot \boldsymbol{v}}(F).$

Finally, we emphasize that the action we consider permutes n tuples of q variables *without permuting the q elements of each tuple*. The definition of $\mathfrak{S}_n$-invariance is thus more general than the well-known notion of symmetric polynomial [36, Def. 1.73] in nq variables: a symmetric polynomial in nq variables is $\mathfrak{S}_n$-invariant, while the converse is not true in general.

Example 5. The function $f : \mathbb{Z}_{2^1}^2 \mapsto \mathbb{F}_2$ defined by $f(x_0, x_1) \to x_{0,0}x_{0,1} + x_{1,0}x_{1,1}$ is $\mathfrak{S}_n$-invariant since for any $\boldsymbol{x}$ and any $\sigma \in \mathfrak{S}_2$, $\sigma \cdot f(\boldsymbol{x}) = f(\boldsymbol{x})$. Indeed $f(\boldsymbol{x}) = x_0^3 + x_1^3 = x_1^3 + x_0^3 = (0\ 1) \cdot f(\boldsymbol{x})$. However, it is not symmetric since $f(x_{0,1}, x_{0,0}, x_{1,1}, x_{1,0}) \neq f(x_{1,1}, x_{0,0}, x_{0,1}, x_{1,0})$. ▷

4.2 Symmetries in the ANF of the dot product

We can now capture the strong symmetries of the ANF of the (product of) bits of the dot product.

Theorem 4. *Let $n, m \in \mathbb{N}^*$. Then $F^{m,n}$ is $\mathfrak{S}_n$-invariant.*

Proof. Let $\otimes$ stands for the coordinate-wise modular multiplication in $\mathbb{Z}_{2^q}$. The announced property is a direct consequence of the commutativity of modular addition. Indeed, for any $\sigma \in \mathfrak{S}_n$, $\mathsf{SUM}^n(\boldsymbol{x}) = \mathsf{SUM}^n(\sigma \cdot \boldsymbol{x})$, which implies that:

$$\begin{aligned} \langle \boldsymbol{a}, \boldsymbol{x} \rangle_n &= \mathsf{SUM}^n(\boldsymbol{a} \otimes \boldsymbol{x}) = \mathsf{SUM}^n(\sigma \cdot (\boldsymbol{a} \otimes \boldsymbol{x})) \\ &= \mathsf{SUM}^n((\sigma \cdot \boldsymbol{a}) \otimes (\sigma \cdot \boldsymbol{x})) = \langle \sigma \cdot \boldsymbol{a}, \sigma \cdot \boldsymbol{x} \rangle_n . \end{aligned}$$ □

By Lemma 2, the $\mathfrak{S}_n$-invariance of $F^{m,n}$ implies that effective gains can be expected for both computing and storing the ANF of $F^{m,n}$.

– By Item **(ii)**, for any $(\boldsymbol{u}, \boldsymbol{v})$, all coefficients (in $\boldsymbol{a}$ and $\boldsymbol{x}$) $\alpha_{\boldsymbol{z},\boldsymbol{t}}(F^{m,n})$ with $(\boldsymbol{z}, \boldsymbol{t}) \in \mathrm{Orb}(\boldsymbol{u}, \boldsymbol{v})$ are equal. This implies that investigating (and storing) a single representative for each orbit is sufficient to understand the full ANF.

– By Item **(iii)**, the same holds for coefficients in $\boldsymbol{x}$ of $F^{m,n}$. In that case, if $\boldsymbol{v}, \boldsymbol{z} \in \mathbb{N}^n$ are in the same orbit, the coefficients $\alpha_{\boldsymbol{v}}(F^{m,n})$ and $\alpha_{\boldsymbol{z}}(F^{m,n})$ are not necessarily equal but can be computed one from the other by applying a permutation (or its inverse).

Example 6. The ANF of $F^{2,2}$ is given by :

$$F^{2,2} = \overbrace{\boldsymbol{a}^{(1,0)}\boldsymbol{x}^{(1,0)} + \boldsymbol{a}^{(0,1)}\boldsymbol{x}^{(0,1)}}^{\mathrm{Orb}((1,0),(1,0))} + \overbrace{\boldsymbol{a}^{(1,1)}\boldsymbol{x}^{(1,1)}}^{\mathrm{Orb}((1,1),(1,1))} + \underbrace{\boldsymbol{a}^{(1,0)}\boldsymbol{x}^{(2,0)} + \boldsymbol{a}^{(0,1)}\boldsymbol{x}^{(0,2)}}_{\mathrm{Orb}((1,0),(2,0))} + \underbrace{\boldsymbol{a}^{(2,0)}\boldsymbol{x}^{(1,0)} + \boldsymbol{a}^{(0,2)}\boldsymbol{x}^{(0,1)}}_{\mathrm{Orb}((2,0),(1,0))}.$$

We observe that a term $\boldsymbol{a}^{\boldsymbol{u}}\boldsymbol{x}^{\boldsymbol{v}}$ appears if and only if all terms in its orbit appear. Furthermore, the coefficient of $\boldsymbol{x}^{(1,0)}$ is $\alpha_{(1,0)} = \boldsymbol{a}^{(1,0)} + \boldsymbol{a}^{(2,0)}$ and it is not equal to the coefficient of $\boldsymbol{x}^{(0,1)}$, namely $\alpha_{(0,1)} = \boldsymbol{a}^{(0,1)} + \boldsymbol{a}^{(0,2)}$, despite the fact that $\sigma \cdot (1,0) = (0,1)$ where σ is the 2-cycle (01). However, it holds that $\sigma^{-1} \cdot \alpha_{(1,0)} = \alpha_{(0,1)}$. ▷

The next proposition provides two useful observations on the support of the exponents of $\mathrm{Exp}(F^{m,n})$ that will be decisive to capture the relationship between the ANF of $F^{m,m}$ and $F^{m,n}$ for $n \geq m$.

Proposition 2. *Let $n, m \in \mathbb{N}^*$. Let $(\boldsymbol{u}, \boldsymbol{v}) \in \mathrm{Exp}(F^{m,n})$. Then :*

(i) *$\boldsymbol{u}$ and $\boldsymbol{v}$ share the same support:* $\forall\ 0 \leq i < n,\ u_i = 0 \iff v_i = 0$.
(ii) *This support has cardinality smaller than m:* $|Supp(\boldsymbol{u})| = |Supp(\boldsymbol{v})| \leq m$.

Proof. By Theorem 1, $F^{m,n} = \pi_m \circ \langle \boldsymbol{a}, \boldsymbol{x} \rangle_n = \sum_{\boldsymbol{c} \in S_m^n} \prod_{i=0}^{n-1} (a_i \times x_i)^{c_i}$. By Theorem 2, the ANF of $(a_i \times x_i)^{c_i}$ only contains terms of the form $a_i^u x_i^v$ with $u \neq 0$ and $v \neq 0$. Indeed, for any $\boldsymbol{e} \in \mathbb{N}^q$, it holds that $g(\boldsymbol{e}) \neq 0$ if and only if $h(\boldsymbol{e}) \neq 0$ by definition of the functions g and h. Item **(ii)** can be seen as a consequence of Item **(i)** of Theorem 3. Indeed the coordinates of $\boldsymbol{v}$, which are positive integers, must sum to less than m so $\boldsymbol{v}$ cannot have strictly more than m non-zero coordinates. □

Together with Proposition 2, the next theorem allows us to show that the ANF of $F^{m,n'}$ for any $n' \geq n \geq m$ can be built from the ANF of $F^{m,n}$.

Theorem 5 (Scaling of the ANF). *Let $m, n, n' \in \mathbb{N}^*$ with $n' \geq n \geq m$. Let $F \colon \mathbb{Z}_{2^q}^n \times \mathbb{Z}_{2^q}^n \to \mathbb{F}_2$ be a $\mathfrak{S}_n$-invariant function that also satisfies Items **(i)** and **(ii)** of Proposition 2. Then there exists a unique $\mathfrak{S}_{n'}$-invariant function $F' \colon \mathbb{Z}_{2^q}^{n'} \times \mathbb{Z}_{2^q}^{n'} \to \mathbb{F}_2$ that satisfies Item **(i)**, Item **(ii)** and:*

$$\forall \boldsymbol{a}, \boldsymbol{x} \in \mathbb{Z}_{2^q}^n, \quad F(\boldsymbol{a}, \boldsymbol{x}) = F'(\boldsymbol{a} || \underbrace{0, \ldots, 0}_{n'-n}, \boldsymbol{x} || \underbrace{0, \ldots, 0}_{n'-n}). \tag{6}$$

Proof. A simple procedure to prove the existence is the following: for each orbit of monomials that appear in the ANF of F, select a representative $\boldsymbol{a}^{\boldsymbol{u}}\boldsymbol{x}^{\boldsymbol{v}}$ and add $\boldsymbol{a}^{\tilde{\boldsymbol{u}}}\boldsymbol{x}^{\tilde{\boldsymbol{v}}}$ to the ANF of F' where $\tilde{\boldsymbol{u}} = \big(\boldsymbol{u}||0,\dots,0\big)$ and $\tilde{\boldsymbol{v}} = \big(\boldsymbol{v}||0,\dots,0\big)$. Note that $\boldsymbol{a}^{\tilde{\boldsymbol{u}}}\boldsymbol{x}^{\tilde{\boldsymbol{v}}}$ is equal to $\boldsymbol{a}^{\boldsymbol{u}}\boldsymbol{x}^{\boldsymbol{v}}$ (but the exponents are of length n' instead of n, with $n'-n$ trailing zeros). Then, add all terms in the orbit of $\boldsymbol{a}^{\tilde{\boldsymbol{u}}}\boldsymbol{x}^{\tilde{\boldsymbol{v}}}$. By construction, F' is thus $\mathfrak{S}_{n'}$-invariant, but also satisfies the properties of Proposition 2 (since these properties are invariant up to permutations). It also satisfies the extension property: all terms of F appears in the ANF of F'. For the remaining terms of F', all of them depend on at least one of the $n'-n$ last variables, fixing these variables to 0 then cancel them all. Regarding unicity, if F'' shares the same properties as F', then $F'-F''$ is also $\mathfrak{S}_{n'}$-invariant and satisfies properties **(i)** and **(ii)**. Because $F'-F''$ is 0 for all values $(\boldsymbol{a}||0,\dots,0,\boldsymbol{x}||0,\dots,0)$, either all its terms involve one of the $n'-n$ last pairs (a_i,x_i), or it is the zero function. But the first case is impossible since any $\boldsymbol{a}^{\boldsymbol{u}}\boldsymbol{x}^{\boldsymbol{v}}$ satisfies $\mathrm{Supp}(\boldsymbol{u}) = \mathrm{Supp}(\boldsymbol{v})$, and $|\mathrm{Supp}(\boldsymbol{u})| = |\mathrm{Supp}(\boldsymbol{v})| \leq m \leq n$ so there always exists $\sigma \in \mathfrak{S}_{n'}$ such that $\sigma \cdot \boldsymbol{a}^{\boldsymbol{u}}\boldsymbol{x}^{\boldsymbol{v}}$ has support within the first n positions. So $F' = F''$. □

By Proposition 2, $F^{m,n}$ for any m,n satisfies properties **(i)** and **(ii)**. Furthermore, $F^{m,n'}$ and $F^{m,n}$ for any $n' \geq n$ also satisfy Eq. (6) since the n-long dot product $\sum_{i=0}^{n-1} a_i x_i$ can just be seen as a particular case of n'-long dot product $\sum_{i=0}^{n'-1} a_i x_i$ where the last $n'-n$ a_i's and x_i's are set to 0. In particular, $F^{m,n'}$ is the "n'-extension" of $F^{m,n}$, and the constructive proof of existence of Theorem 5 can be used in our context: when $m \leq n \leq n'$, the ANF of $F^{m,n'}$ can be fully deduced from the ANF of $F^{m,n}$, or, when $m = n$, from the ANF of $F^{m,m}$.

Example 7. The ANF of $F^{2,2}$ is given by :

$$\begin{aligned} F^{2,2} =& \boldsymbol{a}^{(1,0)}\boldsymbol{x}^{(1,0)} + \boldsymbol{a}^{(0,1)}\boldsymbol{x}^{(0,1)} + \boldsymbol{a}^{(1,1)}\boldsymbol{x}^{(1,1)} + \\ & \underline{\boldsymbol{a}^{(1,0)}\boldsymbol{x}^{(2,0)}} + \boldsymbol{a}^{(0,1)}\boldsymbol{x}^{(0,2)} + \boldsymbol{a}^{(2,0)}\boldsymbol{x}^{(1,0)} + \boldsymbol{a}^{(0,2)}\boldsymbol{x}^{(0,1)}. \end{aligned}$$

By Theorem 5, the fact that the monomial $\boldsymbol{a}^{(1,0)}\boldsymbol{x}^{(2,0)}$ appears in the ANF of $F^{2,2}$ implies that it also appears in the ANF of $F^{2,4}$ under the form $\boldsymbol{a}^{(1,0,0,0)}\boldsymbol{x}^{(2,0,0,0)} = \boldsymbol{a}^{(1,0)}\boldsymbol{x}^{(2,0)}$. It therefore implies that $\boldsymbol{a}^{(0,1)}\boldsymbol{x}^{(0,2)} = \boldsymbol{a}^{(0,1,0,0)}\boldsymbol{x}^{(0,2,0,0)}$ also appears in $F^{2,4}$, but also $\boldsymbol{a}^{(0,0,1,0)}\boldsymbol{x}^{(0,0,2,0)}$ and $\boldsymbol{a}^{(0,0,0,1)}\boldsymbol{x}^{(0,0,0,2)}$ which did not appear in $F^{2,2}$ but belongs to the orbit of $\boldsymbol{a}^{(1,0)}\boldsymbol{x}^{(2,0)}$ when considering vectors of size 4. ▷

Combined together, Theorems 4 and 5 state that it is sufficient to study a system of representatives of the terms of $F^{m,m}$ to fully understand all terms of $F^{m,n}$ for any $m \leq n$. In a similar manner, one can prove that studying the terms of support of size d of $F^{m,d}$ is sufficient to understand all terms of support of size d in $F^{m,n}$ with $d \leq n$. We show in Section 5 how we leverage these results to compute, store and analyze a system of representatives of the terms of $F^{m,m}$ in Python, on a 10-year-old standard laptop, in a few seconds for $m \leq 20$. The relevance of storing and computing this succinct representation of the ANF is illustrated in Section 6, where we provide results such as the average number of non-zero terms or an upper-bound on the rank of the family of functions $(F^{m,n}_{\boldsymbol{a}})_{\boldsymbol{a}\in\mathbb{N}^n}$ for $m \leq 20$ and, most notably, for an arbitrary n.

5 Effective computation of a succinct representation of the ANF of $F^{m,n}$

In this section, we show how to effectively compute and store the full ANF up to permutations using the results of Section 4. By $\mathfrak{S}_n$-invariance of $F^{m,n}$, we have shown that the coefficient $\alpha_{\boldsymbol{u},\boldsymbol{v}}$ of $\boldsymbol{a}^{\boldsymbol{u}}\boldsymbol{x}^{\boldsymbol{v}}$ is equal to the coefficient of any monomial in $\mathrm{Orb}(\boldsymbol{a}^{\boldsymbol{u}}\boldsymbol{x}^{\boldsymbol{v}})$. In Section 5.1, we show how to compute the value of the coefficient of a representative monomial of each orbit. Finally, in Section 5.2, we show how to compute a system of representatives of the ANF of $F^{m,n}$ in a suitable form for the cryptanalyst.

In the following, let $\mathcal{C}^n$ be a *system of representatives* of $\mathbb{N}^n/\sim$, that is, a set built by picking an arbitrary element in each equivalence class of $\mathbb{N}^n/\sim$, *i.e.* in each orbit. For any $\boldsymbol{v}$, we denote by $\boldsymbol{v}^\star$ the representative of $\mathrm{Orb}(\boldsymbol{v})$, that is, the unique element in $\mathrm{Orb}(\boldsymbol{v}) \cap \mathcal{C}^n$. Similarly, let $\mathcal{D}^n$ be a system of representatives for $\mathbb{N}^n \times \mathbb{N}^n/\sim$ and $(\boldsymbol{u},\boldsymbol{v})^\star$ be the unique element of $\mathrm{Orb}(\boldsymbol{u},\boldsymbol{v}) \cap \mathcal{D}^n$ for any $\boldsymbol{u},\boldsymbol{v}$.

5.1 Computing a system of representatives of the $\alpha_{\boldsymbol{u},\boldsymbol{v}}$

Decomposing $F^{m,n}$ into sums and products of subfunctions. Recall that $S_m^n = \left\{\boldsymbol{c} \in \mathbb{N}^n \mid \sum_{i=0}^{n-1} c_i = m\right\}$. By Theorem 1, it holds that:

$$F^{m,n}(\boldsymbol{a},\boldsymbol{x}) = \sum_{\boldsymbol{c}\in S_m^n} \prod_{i=0}^{n-1} (a_i \times x_i)^{c_i}\,.$$

Since permuting a vector preserves the sum of its coordinates, it is clear that S_m^n can be decomposed as a disjoint union of orbits and that $\mathcal{C}_m^n \coloneqq \mathcal{C}^n \cap S_m^n$ contains a single representative of each of these orbits. In particular, one can write

$$F^{m,n}(\boldsymbol{a},\boldsymbol{x}) = \sum_{\boldsymbol{c}\in\mathcal{C}_m^n} \sum_{\boldsymbol{c}'\in\mathrm{Orb}(\boldsymbol{c})} \prod_{i=0}^{n-1} (a_i \times x_i)^{c'_i}\,.$$

We introduce some auxiliary functions $H_{\boldsymbol{c}}, G_{\boldsymbol{c}}$ for $\boldsymbol{c} \in \mathbb{N}^n$, that we define by:

$$H_{\boldsymbol{c}}(\boldsymbol{a},\boldsymbol{x}) \coloneqq \prod_{i=0}^{n-1} (a_i \times x_i)^{c_i} \quad \text{and} \quad G_{\boldsymbol{c}}(\boldsymbol{a},\boldsymbol{x}) \coloneqq \sum_{\boldsymbol{c}'\in\mathrm{Orb}(\boldsymbol{c})} \prod_{i=0}^{n-1} (a_i \times x_i)^{c'_i}\,,$$

so that one can express $F^{m,n}$ as:

$$F^{m,n}(\boldsymbol{a},\boldsymbol{x}) = \sum_{\boldsymbol{c}\in\mathcal{C}_m^n} G_{\boldsymbol{c}}(\boldsymbol{a},\boldsymbol{x}) = \sum_{\boldsymbol{c}\in\mathcal{C}_m^n} \sum_{\boldsymbol{c}'\in\mathrm{Orb}(\boldsymbol{c})} H_{\boldsymbol{c}'}(\boldsymbol{a},\boldsymbol{x})\,. \tag{7}$$

We first state important properties of these auxiliary functions.

Proposition 3. *Let $\boldsymbol{c} \in \mathbb{N}^n$. Then:*

(i) *For any $\boldsymbol{z} \in \mathrm{Orb}(\boldsymbol{c})$, $G_{\boldsymbol{z}} = G_{\boldsymbol{c}}$. Furthermore, $G_{\boldsymbol{c}}$ is $\mathfrak{S}_n$-invariant.*
(ii) *For any $\sigma \in \mathfrak{S}_n$, $\sigma \cdot H_{\boldsymbol{c}} = H_{\sigma^{-1}\cdot\boldsymbol{c}}$. In particular, for any $\boldsymbol{z} \in \mathrm{Orb}(\boldsymbol{c})$, $H_{\boldsymbol{z}} \in \mathrm{Orb}(H_{\boldsymbol{c}})$ and $H_{\boldsymbol{c}}$ is $\mathrm{Stab}(\boldsymbol{c})$-invariant.*

Proof. Let $\boldsymbol{z} \in \mathrm{Orb}(\boldsymbol{c})$. The equality $G_{\boldsymbol{z}} = G_{\boldsymbol{c}}$ is an immediate consequence of the fact that $\mathrm{Orb}(\boldsymbol{c}) = \mathrm{Orb}(\boldsymbol{z})$. Before proving the $\mathfrak{S}_n$-invariance of $G_{\boldsymbol{c}}$, we show **(ii)**. It holds that:

$$\sigma \cdot H_{\boldsymbol{c}}(\boldsymbol{a}, \boldsymbol{x}) = \prod_{i=0}^{n-1} (a_{\sigma^{-1}(i)} \times x_{\sigma^{-1}(i)})^{c_i} = \prod_{j=0}^{n-1} (a_j \times x_j)^{c_{\sigma(j)}} = H_{\sigma^{-1}\cdot\boldsymbol{c}}(\boldsymbol{a}, \boldsymbol{x}) \,,$$

where the second equality is obtained using the change of variable $i = \sigma(j)$. The $\mathrm{Stab}(\boldsymbol{c})$-invariance of $H_{\boldsymbol{c}}$ is clear from the previous equality. Regarding the $\mathfrak{S}_n$-invariance of $G_{\boldsymbol{c}}$, by the previous equalities, it holds that:

$$\begin{aligned} \sigma \cdot G_{\boldsymbol{c}}(\boldsymbol{a}, \boldsymbol{x}) &= \sum_{\boldsymbol{c}' \in \mathrm{Orb}(\boldsymbol{c})} \sigma \cdot H_{\boldsymbol{c}'}(\boldsymbol{a}, \boldsymbol{x}) \\ &= \sum_{\boldsymbol{c}' \in \mathrm{Orb}(\boldsymbol{c})} H_{\sigma^{-1}\cdot\boldsymbol{c}'}(\boldsymbol{a}, \boldsymbol{x}) = \sum_{\boldsymbol{c}' \in \mathrm{Orb}(\boldsymbol{c})} H_{\boldsymbol{c}'}(\boldsymbol{a}, \boldsymbol{x}) = G_{\boldsymbol{c}}(\boldsymbol{a}, \boldsymbol{x}) \,. \end{aligned}$$

□

Theorem 6 below shows that the value of $\alpha_{\boldsymbol{u},\boldsymbol{v}}(G_{\boldsymbol{c}})$ can be computed by considering the ANF of a *single* $H_{\boldsymbol{c}'}$, where $\boldsymbol{c}' \in \mathrm{Orb}(\boldsymbol{c})$, instead of all elements in the orbit of $\boldsymbol{c}$, as the definition of $G_{\boldsymbol{c}}$ suggests.

Theorem 6. *Let $\boldsymbol{c}, \boldsymbol{u}, \boldsymbol{v} \in \mathbb{N}^n \times \mathbb{N}^n$. Let $\mathcal{E}_{\boldsymbol{c}}(\boldsymbol{u}, \boldsymbol{v}) \coloneqq \mathrm{Exp}(H_{\boldsymbol{c}}) \cap \mathrm{Orb}(\boldsymbol{u}, \boldsymbol{v})$. It holds that:*

$$\alpha_{\boldsymbol{u},\boldsymbol{v}}(G_{\boldsymbol{c}}) \quad = \quad \frac{|\mathcal{E}_{\boldsymbol{c}}(\boldsymbol{u}, \boldsymbol{v})|\, n!}{|\mathrm{Stab}(\boldsymbol{c})|\, |\mathrm{Orb}(\boldsymbol{u}, \boldsymbol{v})|} \mod 2 \,. \tag{8}$$

Proof. For any $(\boldsymbol{u}, \boldsymbol{v})$ and any $\boldsymbol{c}$ we have:

$$\alpha_{\boldsymbol{u},\boldsymbol{v}}(G_{\boldsymbol{c}}) = \sum_{\sigma \in \mathfrak{S}_n / \mathrm{Stab}(\boldsymbol{c})} \alpha_{\boldsymbol{u},\boldsymbol{v}}(H_{\sigma\cdot\boldsymbol{c}}) = \sum_{\sigma \in \mathfrak{S}_n / \mathrm{Stab}(\boldsymbol{c})} \alpha_{\sigma^{-1}\cdot(\boldsymbol{u},\boldsymbol{v})}(H_{\boldsymbol{c}}),$$

since for any $\sigma \in \mathfrak{S}_n$, it can be shown that $\alpha_{\boldsymbol{u},\boldsymbol{v}}(H_{\sigma\cdot\boldsymbol{c}}) = \alpha_{\sigma^{-1}\cdot(\boldsymbol{u},\boldsymbol{v})}(H_{\boldsymbol{c}})$ using Proposition 3. Therefore it holds that:

$$\begin{aligned} |\mathrm{Stab}(\boldsymbol{c})|\, \alpha_{\boldsymbol{u},\boldsymbol{v}}(G_{\boldsymbol{c}}) &= \sum_{\tau \in \mathrm{Stab}(\boldsymbol{c})} \alpha_{\boldsymbol{u},\boldsymbol{v}}(G_{\tau\cdot\boldsymbol{c}}) \\ &= \sum_{\tau \in \mathrm{Stab}(\boldsymbol{c})} \sum_{\sigma \in \mathfrak{S}_n / \mathrm{Stab}(\tau\cdot\boldsymbol{c})} \alpha_{\sigma^{-1}\cdot(\boldsymbol{u},\boldsymbol{v})}(H_{\tau\cdot\boldsymbol{c}}) \\ &= \sum_{\tau \in \mathrm{Stab}(\boldsymbol{c})} \sum_{\sigma \in \mathfrak{S}_n / \mathrm{Stab}(\boldsymbol{c})} \alpha_{\tau^{-1}\sigma^{-1}\cdot(\boldsymbol{u},\boldsymbol{v})}(H_{\boldsymbol{c}}) \\ &= \sum_{\varphi \in \mathfrak{S}_n} \alpha_{\varphi\cdot(\boldsymbol{u},\boldsymbol{v})}(H_{\boldsymbol{c}}) \,, \end{aligned}$$

Algorithm 1 `Compute_Exp_star_Gc` ▷ Computing $\mathrm{Exp}^{\star}_{\boldsymbol{a},\boldsymbol{x}}(G_{\boldsymbol{c}})$.

Initialize an empty dictionary `MAP` and an array `Exp_star_Gc`.
`Exp_Hc` ← `Compute_Exp_Hc` ▷ $\mathrm{Exp}_{\boldsymbol{a},\boldsymbol{x}}(H_{\boldsymbol{c}})$
for all $(\boldsymbol{u},\boldsymbol{v}) \in$ `Exp_Hc` **do**
    Compute $(\boldsymbol{u},\boldsymbol{v})^{\star}$
    `MAP`$[(\boldsymbol{u},\boldsymbol{v})^{\star}]$ += 1
        ▷ `MAP`$[(\boldsymbol{u},\boldsymbol{v})^{\star}] = |\mathcal{E}_{\boldsymbol{c}}((\boldsymbol{u},\boldsymbol{v})^{\star})|$
for all $(\boldsymbol{z},\boldsymbol{t})$ such that `MAP`$[(\boldsymbol{z},\boldsymbol{t})] \neq 0$ **do**
    if `MAP`$[(\boldsymbol{z},\boldsymbol{t})] \cdot n!/(|\mathrm{Stab}(\boldsymbol{c})| \cdot |\mathrm{Orb}(\boldsymbol{z},\boldsymbol{t})|) \equiv 1 \bmod 2$ **then**
        Append $(\boldsymbol{z},\boldsymbol{t})$ to `Exp_star_Gc`
return `Exp_star_Gc`. ▷ $\mathrm{Exp}^{\star}_{\boldsymbol{a},\boldsymbol{x}}(G_{\boldsymbol{c}})$

where we use again Proposition 3 for the third equality, but also the fact that $\mathrm{Stab}(\tau \cdot \boldsymbol{c}) = \mathrm{Stab}(\boldsymbol{c})$ since $\tau \in \mathrm{Stab}(\boldsymbol{c})$. Finally, we obtain

$$\sum_{\varphi \in \mathfrak{S}_n} \alpha_{\varphi\cdot(\boldsymbol{u},\boldsymbol{v})}(H_{\boldsymbol{c}}) = \frac{n!}{|\mathrm{Orb}(\boldsymbol{u},\boldsymbol{v})|} \sum_{(\boldsymbol{z},\boldsymbol{t}) \in \mathrm{Orb}(\boldsymbol{u},\boldsymbol{v})} \alpha_{(\boldsymbol{z},\boldsymbol{t})}(H_{\boldsymbol{c}}) = \frac{n!}{|\mathrm{Orb}(\boldsymbol{u},\boldsymbol{v})|} |\mathcal{E}_{\boldsymbol{c}}(\boldsymbol{u},\boldsymbol{v})| \ .$$

□

Efficient computing a system of representatives of the $\boldsymbol{\alpha_{u,v}}$'s. The previous results enable us to simplify the computation of the ANF of $F^{m,n}$. By Equation 7, the ANF of $F^{m,n}$ is the sum of all $G_{\boldsymbol{c}}, \boldsymbol{c} \in \mathcal{C}^n_m$. By Theorem 6, any coefficient of $G_{\boldsymbol{c}}$ can be efficiently determined by considering the ANF of a single $H_{\boldsymbol{c}'}, \boldsymbol{c}' \in \mathrm{Orb}(\boldsymbol{c})$. Finally, by $\mathfrak{S}_n$-invariance of $G_{\boldsymbol{c}}$ (Proposition 3), it is sufficient to determine a single coefficient $\alpha_{\boldsymbol{u},\boldsymbol{v}}(G_{\boldsymbol{c}})$ per orbit to determine whether the entire orbit appears or vanishes. Based on these results, we devise two algorithms.

1. First, Algorithm 1 computes a succinct but comprehensive description of the ANF of $G_{\boldsymbol{c}}$ from the ANF of $H_{\boldsymbol{c}'}, \boldsymbol{c}' \in \mathrm{Orb}(\boldsymbol{c})$. More precisely, it returns the set $\mathrm{Exp}^{\star}_{\boldsymbol{a},\boldsymbol{x}}(G_{\boldsymbol{c}}) \coloneqq \{(\boldsymbol{u},\boldsymbol{v})^{\star} \in \mathcal{D}^n \mid \alpha_{(\boldsymbol{u},\boldsymbol{v})^{\star}}(G_{\boldsymbol{c}}) = 1\}$. which contains a single exponent for each appearing orbit. The initial computing of the ANF of $H_{\boldsymbol{c}}$ can be done using a direct and efficient procedure described in the long version of this work [11, Appendix A.1].
2. Secondly, Algorithm 2 internally uses Algorithm 1 to determine the value of the set $\mathrm{Exp}^{\star}_{\boldsymbol{a},\boldsymbol{x}}(F^{m,n}) \coloneqq \{(\boldsymbol{u},\boldsymbol{v})^{\star} \in \mathcal{D}^n \mid \alpha_{(\boldsymbol{u},\boldsymbol{v})^{\star}}(F^{m,n}) = 1\}$, which again contains a single representative per orbit. As implied by the $\mathfrak{S}_n$-invariance of $F^{m,n}$ (Theorem 4, Lemma 2), this fully determines all the coefficients of the ANF of $F^{m,n}$.

All the subroutines of our algorithms are described in [11, Appendix A.1].

Algorithm 2 `Compute_Exp_star_Fmn` ▷ Computing $\mathrm{Exp}^{\star}_{\boldsymbol{a},\boldsymbol{x}}(F^{m,n})$.

Initialize an empty dictionary `Coeff_Fmn` and an empty array `Exp_star_Fmn`
for all $\boldsymbol{c} \in \mathcal{C}^n_m$ **do**
    `Exp_star_Gc` $\leftarrow$ `Compute_Exp_star_Gc`$(\boldsymbol{c})$ ▷ Algorithm 1.
    for all $(\boldsymbol{u}, \boldsymbol{v})$ in `Exp_star_Gc` **do**
        `Coeff_Fmn`$[(\boldsymbol{u}, \boldsymbol{v})]$ += 1
for all $(\boldsymbol{u}, \boldsymbol{v})$ in `Coeff_Fmn` **do**
    if `Coeff_Fmn`$[(\boldsymbol{u}, \boldsymbol{v})] \equiv 1 \bmod 2$ **then**
        Append $(\boldsymbol{u}, \boldsymbol{v})$ to `Exp_star_Fmn`
return `Exp_star_Fmn` ▷ $\mathrm{Exp}^{\star}_{\boldsymbol{a},\boldsymbol{x}}(F^{m,n})$.

5.2 A more convenient representation for the cryptanalyst

The algorithms of the previous section allow us to obtain a succinct description of the ANF of $F^{m,n}$ as a polynomial in $\boldsymbol{a}$ and in $\boldsymbol{x}$ – a system of representatives of the exponents $(\boldsymbol{u}, \boldsymbol{v})$ such that $\boldsymbol{a}^{\boldsymbol{u}}\boldsymbol{x}^{\boldsymbol{v}}$ appears in the ANF (*i.e.*, $\alpha_{\boldsymbol{u},\boldsymbol{v}} = 1$). However, this representation is not well-suited to the cryptanalyst because the variables $\boldsymbol{a}$ and $\boldsymbol{x}$ play very different roles: the variables $\boldsymbol{a}$ are *public* but *random*, whilst the variables $\boldsymbol{x}$ are secret. As detailed at the end of Section 2.4, a more convenient representation is to view $F^{m,n}$ as a polynomial in $\boldsymbol{x}$, with each monomial $\boldsymbol{x}^{\boldsymbol{v}}$ having a coefficient $\alpha_{\boldsymbol{v}} \in \mathbb{F}_2[\boldsymbol{a}]$. Indeed, this allows us to efficiently compute the polynomial $F^{m,n}_{\boldsymbol{a}}$ in $\boldsymbol{x}$ for any random $\boldsymbol{a}$.

Recall that $\mathrm{Exp}_{\boldsymbol{x}}(F^{m,n}) = \{\boldsymbol{v} \in \mathbb{N}^n, \alpha_{\boldsymbol{v}}(F^{m,n}) \neq 0\}$. A priori, the goal of this section is thus to compute the $|\mathrm{Exp}_{\boldsymbol{x}}(F^{m,n})|$ different $\alpha_{\boldsymbol{v}}$ for all $\boldsymbol{v} \in \mathrm{Exp}_{\boldsymbol{x}}(F^{m,n})$. However, as before, we can exploit the $\mathfrak{S}_n$-invariance of $F^{m,n}$: for any $\boldsymbol{v}$ and any $\boldsymbol{v}' \in \mathrm{Orb}(\boldsymbol{v})$ such that $\sigma \cdot \boldsymbol{v}' = \boldsymbol{v}$, the coefficient $\alpha_{\boldsymbol{v}'}$ can be computed from $\alpha_{\boldsymbol{v}}$ as $\sigma \cdot \alpha_{\boldsymbol{v}} = \alpha_{\sigma^{-1}\cdot\boldsymbol{v}} = \alpha_{\boldsymbol{v}'}$ by Item **(iii)** of Lemma 2. In this section, we thus aim at computing the following set:

$$\mathrm{Pairs}_{\boldsymbol{x}}(F^{m,n}) \coloneqq \{(\boldsymbol{v}^{\star}, \alpha_{\boldsymbol{v}^{\star}}) \ : \ \boldsymbol{v}^{\star} \in \mathrm{Exp}^{\star}_{\boldsymbol{x}}(F^{m,n})\} \ .$$

where $\mathrm{Exp}^{\star}_{\boldsymbol{x}}(F^{m,n}) \coloneqq \mathcal{C}^n \cap \mathrm{Exp}_{\boldsymbol{x}}(F^{m,n}) = \{\boldsymbol{v}^{\star} \in \mathcal{C}^n, \alpha_{\boldsymbol{v}^{\star}}(F^{m,n}) \neq 0\}$.

Remark 2. We emphasize that we purposefully consider a set of pairs. Indeed, we do not only want exponents as the cryptanalyst must know the ANF of each coefficient $\alpha_{\boldsymbol{v}}$ to build the linearised system. Furthermore, we do not only want a system of representatives of the coefficients, as in this case, we lose some important information to build the system:[7] there may exist $\boldsymbol{v}^{\star}_1 \neq \boldsymbol{v}^{\star}_2$ such that $\alpha_{\boldsymbol{v}^{\star}_1} = \alpha_{\boldsymbol{v}^{\star}_2}$ or, more generally, $\boldsymbol{v}^{\star}_1 \neq \boldsymbol{v}^{\star}_2$ such that $\alpha_{\boldsymbol{v}^{\star}_1} \in \mathrm{Orb}(\alpha_{\boldsymbol{v}^{\star}_2})$.

The following proposition enables us to derive $\mathrm{Pairs}_{\boldsymbol{x}}(F^{m,n})$ from the set $\mathrm{Exp}^{\star}_{\boldsymbol{x}}(F^{m,n})$ output by Algorithm 2.

[7] In fact, even in Section 6.2 where we exploit the fact that some monomials in $\boldsymbol{x}$ have the same coefficient, we need to know which ones in order to compute the new linearisation family.

Algorithm 3 `Compute_Pairs_Fmn` ▷ Computing $\mathrm{Pairs}_{\boldsymbol{x}}(F^{m,n})$.

1: Initialize an empty dictionary `Pairs_Fmn`.
2: `Exp_star_Fmn` $\leftarrow$ `Compute_Exp_star_Fmn`. ▷ Algorithm 2.
3: **for all** $(\boldsymbol{u}, \boldsymbol{v}) \in$ `Exp_star_Fmn` **do**
4: Compute $\tau \in \mathfrak{S}_n$ s.t. $\tau \cdot \boldsymbol{v} = \boldsymbol{v}^\star$.
5: `Pairs_Fmn`$[\boldsymbol{v}^\star] \leftarrow$ `Pairs_Fmn`$[\boldsymbol{v}^\star] \cup \{\boldsymbol{a}^{\sigma\tau\cdot\boldsymbol{u}}, \sigma \in \mathrm{Stab}(\boldsymbol{v}^\star)\}$.
6: **return** `Pairs_Fmn`. ▷ $\mathrm{Pairs}_{\boldsymbol{x}}(F^{m,n})$ (as a dictionary).

Proposition 4. *Let $\boldsymbol{v} \in \mathbb{N}^n$. For any $\boldsymbol{t} \in \mathrm{Orb}(\boldsymbol{v})$, let $\sigma_{\boldsymbol{t}}$ be a permutation such that $\sigma_{\boldsymbol{t}} \cdot \boldsymbol{t} = \boldsymbol{v}$. It holds that:*

$$\alpha_{\boldsymbol{v}} = \sum_{(\boldsymbol{z},\boldsymbol{t}) \in \mathrm{Exp}^\star_{\boldsymbol{a},\boldsymbol{x}}(F^{m,n}) \ \mid \ \boldsymbol{t} \in \mathrm{Orb}(\boldsymbol{v})} \ \sum_{\tau \in \mathrm{Stab}(\boldsymbol{v})} \boldsymbol{a}^{\tau\sigma_{\boldsymbol{t}}\cdot\boldsymbol{z}} \, .$$

Proof. It holds that:

$$\alpha_{\boldsymbol{v}} = \sum_{(\boldsymbol{z},\boldsymbol{t}) \in \mathrm{Exp}^\star_{\boldsymbol{a},\boldsymbol{x}}(F^{m,n})} \ \sum_{(\boldsymbol{u},\boldsymbol{v}) \in \mathrm{Orb}(\boldsymbol{z},\boldsymbol{t})} \boldsymbol{a}^{\boldsymbol{u}} \, .$$

Let $(\boldsymbol{z}, \boldsymbol{t}) \in \mathrm{Exp}^\star_{\boldsymbol{a},\boldsymbol{x}}(F^{m,n})$ such that $\boldsymbol{t} \in \mathrm{Orb}(\boldsymbol{v})$ and let $\mathcal{P}$ be the projection $\mathcal{P} : (\boldsymbol{u}, \boldsymbol{v}) \mapsto \boldsymbol{u}$. Since $\mathrm{Orb}(\boldsymbol{z}) = \mathcal{P}(\mathrm{Orb}(\boldsymbol{z}, \boldsymbol{t}))$, there exists $\boldsymbol{u}$ such that $(\boldsymbol{z}, \boldsymbol{t}) \in \mathrm{Orb}(\boldsymbol{u}, \boldsymbol{v})$, which is equivalent to $(\boldsymbol{u}, \boldsymbol{v}) \in \mathrm{Orb}(\boldsymbol{z}, \boldsymbol{t})$. Moreover, for all such $\boldsymbol{u}$, $\mathrm{Orb}(\boldsymbol{u}, \boldsymbol{v}) \cap \mathrm{Exp}^\star_{\boldsymbol{a},\boldsymbol{x}}(F^{m,n}) \neq \emptyset$ which implies that $\alpha_{\boldsymbol{u},\boldsymbol{v}} = 1$. Letting $\sigma_{\boldsymbol{t}}$ to be a permutation such that $\sigma_{\boldsymbol{t}} \cdot \boldsymbol{t} = \boldsymbol{v}$, we show that

$$\{\boldsymbol{u}, (\boldsymbol{u}, \boldsymbol{v}) \in \mathrm{Orb}(\boldsymbol{z}, \boldsymbol{t})\} = \{\tau\sigma_{\boldsymbol{t}} \cdot \boldsymbol{z}, \tau \in \mathrm{Stab}(\boldsymbol{v})\} \, .$$

Let $\tau \in \mathrm{Stab}(\boldsymbol{v})$. It holds that $(\tau\sigma_{\boldsymbol{t}})^{-1}(\tau\sigma_{\boldsymbol{t}} \cdot \boldsymbol{z}, \boldsymbol{v}) = (\boldsymbol{z}, \boldsymbol{t})$, and thus $(\boldsymbol{z}, \boldsymbol{t}) \in \mathrm{Orb}(\tau\sigma_{\boldsymbol{t}} \cdot \boldsymbol{z}, \boldsymbol{v})$. Conversely, let $\boldsymbol{u}$ such that $(\boldsymbol{z}, \boldsymbol{t}) \in \mathrm{Orb}(\boldsymbol{u}, \boldsymbol{v})$. There exists σ such that $(\sigma \cdot \boldsymbol{u}, \sigma \cdot \boldsymbol{v}) = (\boldsymbol{z}, \boldsymbol{t})$. Since $\sigma^{-1} \cdot \boldsymbol{t} = \boldsymbol{v}$, $\sigma^{-1} \in \mathrm{Stab}(\boldsymbol{v})\sigma_{\boldsymbol{t}}$, and $\sigma^{-1} = \varphi\sigma_{\boldsymbol{t}}$ for some $\varphi \in \mathrm{Stab}(\boldsymbol{v})$. Thus, $\boldsymbol{u} = \sigma^{-1} \cdot \boldsymbol{z} = \varphi\sigma_{\boldsymbol{t}} \cdot \boldsymbol{z}$. □

From Proposition 4, we derive Algorithm 3 below that outputs $\mathrm{Pairs}_{\boldsymbol{x}}(F^{m,n})$. All the implementation details and subroutines can be found in the long version of this work [11, Appendix A.2].

6 Exploiting our succinct description of the full ANF

In the previous section, we presented how to take advantage of the symmetries of the ANF of $F^{m,n}$ to fully capture the ANF without having to fully store it nor compute it. We implemented the algorithms described above in practice. They work well and fast on a standard computer for m up to 20. In this section, we show the relevancy of this succinct description of the ANF. As a main contribution, we show how to compute *exact* metrics about the Boolean non-linear

system described in Section 3 for arbitrary large n *by only computing and storing a succinct description of the ANF for* $n = m$. The computation of these metrics can be seen as a proof of concept as it illustrates both the efficiency of our method and the relevance of the system of representatives. Moreover, as detailed in the sequel, they allow to assess further the security of LWR. In particular, our results show that we can improve the linearisation attack of Arora-Ge [7] beyond the natural benefit of working over $\mathbb{F}_2$ rather than $\mathbb{Z}_{2^q}$ (see Section 3.1) for all parameters considered, with various degrees of improvement.

6.1 Average number of non-zero terms in an equation

Many linear algebra algorithms take advantage of the sparsity of the involved matrices (e.g. the Block-Wiedemann algorithm [46,25]). It is thus interesting to measure the average number of non-zero terms in $\boldsymbol{x}$ in a Boolean LWR equation when $\boldsymbol{a}$ is uniformly random [31]. This corresponds to the sparsity of a random row in the linearisation matrix (see [31] for more details).

In our context, this quantity can be computed efficiently. Indeed, by $\mathfrak{S}_n$-invariance of $F^{m,n}$, for any $\boldsymbol{v}^\star \in \mathrm{Exp}^\star_{\boldsymbol{x}}(F^{m,n})$, there exist $|\mathrm{Orb}(\boldsymbol{v}^\star)|$ permuted copies of $\alpha_{\boldsymbol{v}^\star}$ in the ANF of $F^{m,n}$, namely all $\alpha_{\boldsymbol{w}}$ for $\boldsymbol{w} \in \mathrm{Orb}(\boldsymbol{v}^\star)$. The bias of $\alpha_{\boldsymbol{v}^\star}$ (or equivalently the probability $p_{\alpha_{\boldsymbol{v}^\star}}$ that $\alpha_{\boldsymbol{v}^\star}(\boldsymbol{a}) = 0$ when $\boldsymbol{a} \in \mathbb{Z}_{2^q}^n$ is drawn uniformly at random) remains unchanged when the inputs are permuted, and is therefore constant within the orbit of $\alpha_{\boldsymbol{v}^\star}$ since $\alpha_{\sigma\cdot\boldsymbol{v}^\star} = \sigma^{-1}\cdot\alpha_{\boldsymbol{v}^\star}$. All in all, over uniformly random $\boldsymbol{a}$, the expected number of terms in $\boldsymbol{x}$ in the ANF of $F^{m,n}_{\boldsymbol{a}}(\boldsymbol{x})$ is given by:

$$\sum_{\boldsymbol{v}^\star \in \mathrm{Exp}^\star_{\boldsymbol{x}}(F^{m,n})} (1 - p_{\alpha_{\boldsymbol{v}^\star}})\,|\mathrm{Orb}(\boldsymbol{v}^\star)|\ . \tag{9}$$

When the number of variables is small and the ANF is available, $p_{\alpha_{v^\star}}$ can be effectively computed e.g. through a (Fast) Fourier Transform. While Equation 9 depends on the length n of the vector (since the size of each orbit does), the next proposition states that we can easily extrapolate the results for $n > m$ from the case $n = m$. Let $\mathbf{0}^{n-d}$ the zero vector of size $n-d$. For any $\boldsymbol{v} \in \mathbb{N}^d$ such that $|\mathrm{Supp}(\boldsymbol{v})| = d$ and for any $n \geq d$, we denote by $\mathrm{Orb}_n(\boldsymbol{v})$ the orbit of $\boldsymbol{v}$ seen "as a vector of size n": $\mathrm{Orb}_n(\boldsymbol{v}) \coloneqq \left\{\sigma\cdot\left(\boldsymbol{v}||\mathbf{0}^{n-d}\right), \sigma \in \mathfrak{S}_n\right\}$. Let $\alpha \in \mathrm{Coefficients}_{\boldsymbol{x}}(F^{m,n})$. By Proposition 2, for any $\boldsymbol{u} \in \mathrm{Exp}_{\boldsymbol{a}}(\alpha)$ and any $\boldsymbol{v}$ such that $\alpha_{\boldsymbol{v}}(F^{m,n}) = \alpha$, it holds that $\mathrm{Supp}(\boldsymbol{u}) = \mathrm{Supp}(\boldsymbol{v})$ and thus that $\mathrm{Supp}(\alpha) = \mathrm{Supp}(\boldsymbol{v})$. For any $n \geq |\mathrm{Supp}(\alpha)|$, we can thus define $\mathrm{Orb}_n(\alpha)$ in a similar manner.

Proposition 5. *Let* $\boldsymbol{v} \in \mathbb{N}^n$ *and* $d = |\mathit{Supp}(\boldsymbol{v})|$. *Let* $\alpha \in \mathit{Coefficients}_{\boldsymbol{x}}(F^{m,n})$, *and* $d' = |\mathit{Supp}(\alpha)|$. *It holds that:*

$$|\mathrm{Orb}_n(\boldsymbol{v})| = |\mathrm{Orb}_d(\boldsymbol{v})| \cdot \binom{n}{d} \quad \textit{and} \quad |\mathrm{Orb}_n(\alpha)| = |\mathrm{Orb}_{d'}(\alpha)| \cdot \binom{n}{d'}.$$

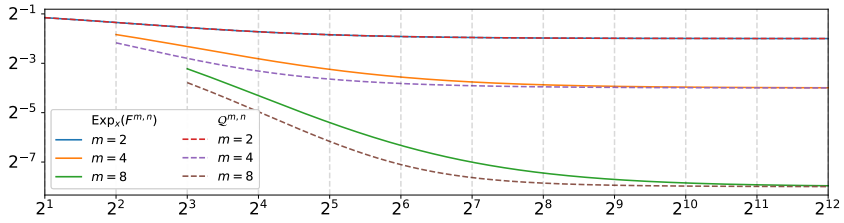

Fig. 1: Average fraction of terms in a random equation $F^{m,n}_{\boldsymbol{a}}(\boldsymbol{x})$ as a function of n, for $n \in [\![m, 4096]\!]$ w.r.t the spanning families $\mathrm{Exp}_{\boldsymbol{x}}(F^{m,n})$ and $\mathcal{Q}^{n,m}$.

Proof. Let I', I'' be two distinct subsets of $[\![0, n-1]\!]$ of cardinality d. The set $\{\boldsymbol{v}' \in \mathrm{Orb}_n(\boldsymbol{v}),\ \mathrm{Supp}(\boldsymbol{v}') = I'\}$ is of size exactly $\mathrm{Orb}_d(\boldsymbol{v})$. It holds also that:

$$\{\boldsymbol{v}' \in \mathrm{Orb}_n(\boldsymbol{v}),\ \mathrm{Supp}(\boldsymbol{v}') = I'\} \cap \{\boldsymbol{v}'' \in \mathrm{Orb}_n(\boldsymbol{v}),\ \mathrm{Supp}(\boldsymbol{v}'') = I''\} = \emptyset\,.$$

The same holds for α since $\mathrm{Supp}(\alpha) = \mathrm{Supp}(\boldsymbol{v})$ for any $\boldsymbol{v}$ such that $\alpha_{\boldsymbol{v}} = \alpha$. □

As a direct corollary, for $n > m$, the average sparsity can be efficiently and exactly computed from the case $n = m$ thanks to the following formula:

$$\sum_{\boldsymbol{v}^\star \in \mathrm{Exp}^\star_{\boldsymbol{x}}(F^{m,m})} (1 - p_{\alpha_{\boldsymbol{v}^\star}})\mathrm{Orb}_{|\mathrm{Supp}(\boldsymbol{v}^\star)|}(\boldsymbol{v}^\star)\binom{n}{|\mathrm{Supp}(\boldsymbol{v}^\star)|}. \tag{10}$$

This is illustrated by the solid-lines of Figure 1; dashed lines are explained later.

6.2 Rank defect for LWR Boolean equations

Identifying a better linearising set. As already observed with Equation 1 of Section 2.4, expressing $F^{m,n}$ under the form $F^{m,n} = \sum_{\boldsymbol{v} \in \mathbb{N}^n} \alpha_{\boldsymbol{v}}(F^{m,n})\boldsymbol{x}^{\boldsymbol{v}}$ allows the attacker to build an affine system $\mathcal{S}$ of the form

$$F^{m,n}(\boldsymbol{a}^{(0)}, \boldsymbol{x}) = y_0, \quad F^{m,n}(\boldsymbol{a}^{(1)}, \boldsymbol{x}) = y_1, \quad \ldots \quad F^{m,n}(\boldsymbol{a}^{(M-1)}, \boldsymbol{x}) = y_{M-1}.$$

where, for any i, $\boldsymbol{a}^{(i)}$ is the i-th random-input and y_i the corresponding output bit (or product of bits). Each equation is simply obtained by evaluating all coefficients $\alpha_{\boldsymbol{v}}(F^{m,n}) \in \mathbb{F}_2[\boldsymbol{a}]$ at point $\boldsymbol{a}^{(i)}$. In this context, an interesting metric to evaluate resistance to linearisation attacks is the *rank of the polynomial system*, that is, the dimension of the $\mathbb{F}_2$-linear space V spanned by the set $\left\{F^{m,n}(\boldsymbol{a}^{(i)}, \boldsymbol{x})\right\}_{i \in [\![0, M-1]\!]}$ or, more generally, by the set $\{F^{m,n}_{\boldsymbol{a}}(\boldsymbol{x})\}_{\boldsymbol{a} \in \mathbb{Z}^n_{2^q}}$.

Assume for example that some monomials $\boldsymbol{x}^{\boldsymbol{v}_0}, \ldots, \boldsymbol{x}^{\boldsymbol{v}_{s-1}}$ always appear and vanish together in the equations of $\mathcal{S}$. For a linearisation attack, it is then better to consider the polynomial $\sum_{i=0}^{s-1} \boldsymbol{x}^{\boldsymbol{v}_i}$ as a single auxiliary variable instead of introducing s independent ones. Terms that appear and vanish at the same

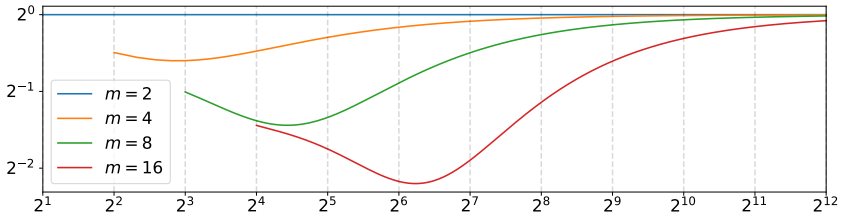

Fig. 2: $\frac{|\mathcal{Q}^{m,n}|}{|\mathrm{Exp}_{\boldsymbol{x}}(F^{m,n})|}$ as a function of n, for $n \in [\![m, 4096]\!]$.

time depending on $\boldsymbol{a}$ are in fact terms which share the same coefficient $\alpha \in \mathrm{Coefficients}_{\boldsymbol{x}}(F)$. We thus express $F^{m,n}$ as:

$$F^{m,n} = \sum_{\alpha \in \mathrm{Coefficients}_{\boldsymbol{x}}(F^{m,n})} \alpha Q_\alpha \text{ where } Q_\alpha := \sum_{\substack{\boldsymbol{v} \in \mathbb{N}^n \text{s.t.} \\ \alpha_{\boldsymbol{v}}(F^{m,n}) = \alpha}} \boldsymbol{x}^{\boldsymbol{v}} \in \mathbb{F}_2[\boldsymbol{x}]. \quad (11)$$

The set of polynomials $\mathcal{Q}^{m,n} = \{Q_\alpha, \alpha \in \mathrm{Coefficients}_{\boldsymbol{x}}(F^{m,n})\}$ is a generating set of V since, by design, any equation can be written as a linear combination of some Q_α. Its cardinality satisfies $|\mathcal{Q}^{m,n}| = |\mathrm{Coefficients}_{\boldsymbol{x}}(F^{m,n})| \leq |\mathrm{Exp}_{\boldsymbol{x}}(F^{m,n})| \leq \binom{n+m}{m} - 1$, which implies that it is a (strictly) smaller family than the family of the appearing monomials. When the first inequality is strict, a change of linearising set from $\mathrm{Exp}_{\boldsymbol{x}}(F^{m,n})$ to $\mathcal{Q}^{m,n}$ improves the complexity of a linearisation attack from $|\mathrm{Exp}_{\boldsymbol{x}}(F^{m,n})|^3$ to $|\mathcal{Q}^{m,n}|^3$, see e.g. [31] and the theoretical reason of the associated rank defect [8].

Remark 3. Note that $|\mathcal{Q}^{m,n}|$ is not the rank of the polynomial system, but a better upper bound on the rank than the number of monomials. Indeed, there may exist sets of coefficients that are linearly dependent but not all equal. Computing the exact rank and an associated basis remains an open question.

We managed to compute the set $\mathcal{Q}^{m,n}$ for values of $m \leq 20$ and n up to (an arbitrary value of) 4096. Our results are illustrated in Figure 2, where we compare $|\mathcal{Q}^{m,n}|$ to the number of monomials $|\mathrm{Exp}_{\boldsymbol{x}}(F^{m,n})|$, the latter value being easily computed thanks to the results of Sections 4 and 5. As mentioned in Remark 1, in the case of a power-of-two m, we observed that $|\mathrm{Exp}_{\boldsymbol{x}}(F^{m,n})| = \binom{n+m}{n} - 1$, matching the upper bound of Theorem 3.

Our upper bound on the rank suggests that the rank significantly drops compared to the number of monomials when n is small (when $m = 16$, $\frac{|\mathcal{Q}^{m,n}|}{|\mathrm{Exp}_{\boldsymbol{x}}(F^{m,n})|} \simeq 0.22, 0.27, 0.45$ for $n = 64, 128, 256$ respectively) but could still tend to $\binom{n+m}{m} - 1$ for large values of n. We detail the gains provided by this drop in Section 7.1. For now, we describe our method for computing $\mathcal{Q}^{m,n}$.

Computing a system of representatives of the Q_α's. In Theorem 7, we provide an orbit-by-orbit decomposition of each $Q_\alpha \in \mathbb{F}_2[\boldsymbol{x}]$ defined above and

show that $Q_{\sigma\cdot\alpha} = \sigma \cdot Q_\alpha$. From there, our strategy will again consist in leveraging these structures to efficiently build a system of representatives of $\mathcal{Q}^{m,n}/\sim$ from the output of Algorithm 3 *i.e.*, $\mathrm{Pairs}_{\boldsymbol{x}}(F^{m,n}) \coloneqq \{(\boldsymbol{v}^\star, \alpha_{\boldsymbol{v}^\star}) \mid \boldsymbol{v}^\star \in \mathrm{Exp}^\star_{\boldsymbol{x}}(F^{m,n})\}$.

Theorem 7. *Let $n \in \mathbb{N}^*$, $\alpha \in \mathit{Coefficients}_{\boldsymbol{x}}(F^{m,n})$. Let $\mathcal{B}^n_\alpha$ be defined as $\mathcal{B}^n_\alpha \coloneqq \{\boldsymbol{w} \in \mathrm{Exp}^\star_{\boldsymbol{x}}(F^{m,n}) \mid \alpha_{\boldsymbol{w}} \in \mathrm{Orb}(\alpha)\}$. Then, it holds that:*

$$\textbf{\textit{(i)}}\ Q^n_\alpha = \sum_{\boldsymbol{v}\in\mathcal{B}^n_\alpha} \sum_{\substack{\boldsymbol{w}\in\mathrm{Orb}(\boldsymbol{v})\\ \alpha_{\boldsymbol{w}}=\alpha}} \boldsymbol{x}^{\boldsymbol{w}} \quad \textit{and}\ \textbf{\textit{(ii)}}\ \textit{for any}\ \sigma \in \mathfrak{S}_n,\ Q^n_{\sigma\cdot\alpha} = \sigma \cdot Q^n_\alpha\,.$$

Proof. Item **(i)** holds because $F^{m,n}(\boldsymbol{a}, \boldsymbol{x}) = \sum_{\boldsymbol{v}\in\mathrm{Exp}^\star_{\boldsymbol{x}}(F^{m,n})} \sum_{\boldsymbol{w}\in\mathrm{Orb}(\boldsymbol{v})} \boldsymbol{x}^{\boldsymbol{w}}\alpha_{\boldsymbol{w}}$ by $\mathfrak{S}_n$-invariance of $F^{m,n}$. Item **(ii)** stems from $\mathcal{B}^n_\alpha = \mathcal{B}^n_{\sigma\cdot\alpha}$ and:

$$\sigma \cdot \sum_{\substack{\boldsymbol{w}\in\mathrm{Orb}(\boldsymbol{v})\\ \alpha_{\boldsymbol{w}}=\alpha}} \boldsymbol{x}^{\boldsymbol{w}} = \sum_{\substack{\boldsymbol{w}\in\mathrm{Orb}(\boldsymbol{v})\\ \alpha_{\boldsymbol{w}}=\alpha}} \boldsymbol{x}^{\sigma^{-1}\cdot\boldsymbol{w}} = \sum_{\substack{\boldsymbol{t}\in\mathrm{Orb}(\boldsymbol{v})\\ \alpha_{\sigma\cdot\boldsymbol{t}}=\alpha}} \boldsymbol{x}^{\boldsymbol{t}} = \sum_{\substack{\boldsymbol{t}\in\mathrm{Orb}(\boldsymbol{v})\\ \alpha_{\boldsymbol{t}}=\sigma\cdot\alpha}} \boldsymbol{x}^{\boldsymbol{t}}\,. \qquad \square$$

Algorithm 4 of [11, Appendix B] thus enables us to compute a system of representatives of $\mathcal{Q}^{m,n}/\sim$ using the following strategy. For each $\alpha_{\boldsymbol{v}^\star}$ output by Algorithm 3, we efficiently go over $\mathrm{Orb}(\alpha_{\boldsymbol{v}^\star})$ by leveraging the fact that, since $F^{m,n}$ is $\mathfrak{S}_n$-invariant, $\sigma \cdot \alpha_{\boldsymbol{v}^\star}(F) = \alpha_{\sigma^{-1}\cdot\boldsymbol{v}^\star}(F)$ for any σ and therefore $\alpha_{\boldsymbol{v}}$ is $\mathrm{Stab}(\boldsymbol{v})$-invariant.[8] Two things can happen while going through $\mathrm{Orb}(\alpha_{\boldsymbol{v}^\star})$:

- Either we detect $\alpha_{\sigma\cdot\boldsymbol{v}^\star} \in \mathrm{Orb}(\alpha_{\boldsymbol{v}^\star})$ such that $\alpha_{\sigma\cdot\boldsymbol{v}^\star} = \alpha_{\boldsymbol{w}^\star}$ for some $\boldsymbol{w}^\star \neq \boldsymbol{v}^\star$ that we already scanned before. In this case, we add $\boldsymbol{x}^{\sigma\cdot\boldsymbol{v}^\star}$ to $Q_{\boldsymbol{w}^\star}$, as well as all monomials $\boldsymbol{x}^{\tau\cdot\boldsymbol{v}^\star}$ that satisfy $\alpha_{\tau\cdot\boldsymbol{v}^\star} = \alpha_{\boldsymbol{w}^\star}$.
- If no such element exists in $\mathrm{Orb}(\alpha_{\boldsymbol{v}^\star})$, then $\alpha_{\boldsymbol{v}^\star}$ lies in a new orbit that will have $\alpha_{\boldsymbol{v}^\star}$ as representative. Thus we create an entry for $Q_{\alpha_{\boldsymbol{v}^\star}}$, and add $x^{\boldsymbol{v}^\star}$ to it, as well as all monomials $x^{\tau\cdot\boldsymbol{v}^\star}$ that satisfy $\alpha_{\tau\cdot\boldsymbol{v}^\star} = \alpha_{\boldsymbol{v}^\star}$.

Since Q_α is defined (see Eq. (11)) as the sum of all $\boldsymbol{x}^{\boldsymbol{v}}$ such that $\alpha_{\boldsymbol{v}} = \alpha$, Algorithm 4 of [11, Appendix B] also immediately provides a system of representatives for the $\alpha_{\boldsymbol{v}}$'s.

Scaling of the pairs $(\boldsymbol{\alpha}, \boldsymbol{Q_\alpha})$. The one-to-one correspondence between the α's and Q_α's implies that it is easy to compute the size of $|\mathcal{Q}^{m,n}|$ for an arbitrary $n \geq m$ by scaling the size of the orbits of each $\alpha_{\boldsymbol{v}}$ in a system of representatives of the coefficients, namely $\mathrm{Coefficients}_{\boldsymbol{x}}(F^{m,m})/\sim$. We obtain the following formula:

$$|\mathcal{Q}^{m,n}| = \sum_{\alpha\in\mathrm{Coefficients}_{\boldsymbol{x}}(F^{m,m})/\sim} \mathrm{Orb}_{|\mathrm{Supp}(\alpha)|}(\alpha) \cdot \binom{n}{|\mathrm{Supp}(\alpha)|}\,.$$

As illustrated in Figure 2, we computed the cardinality of $\mathcal{Q}^{m,m}$ for m up to 16 and scaled it for n up to (an arbitrary value of) 4096. It is also possible to compute the average number of non-zero coefficients $\alpha \in \mathrm{Coefficients}_{\boldsymbol{x}}(F^{m,m})$ for

[8] In fact, this holds for any coefficient $\alpha_{\boldsymbol{v}}$ of a $\mathfrak{S}_n$-invariant function $F\colon \mathbb{Z}^n_{2^q} \times \mathbb{Z}^n_{2^q} \to \mathbb{F}_2$.

a uniformly random sample $\boldsymbol{a}$. It exactly corresponds to the sparsity of equations w.r.t this new generating set and, similarly to Equation 10, it is given by:

$$\sum_{\alpha \in \text{Coefficients}_{\boldsymbol{x}}(F^{m,m})/\sim} (1-p_\alpha)\text{Orb}_{|\text{Supp}(\alpha)|}(\alpha)\binom{n}{|\text{Supp}(\alpha)|}.$$

We computed the average sparsity for m up to 8, and Figure 1 displays the value obtained through scaling for n up to (an arbitrary value of) 4096.

7 Conclusion and open problems

7.1 A systematic improvement over the attack of Arora and Ge

Our techniques allow to systematically improve the attack of Arora and Ge (or to launch a valid attack when the one of Arora and Ge is not applicable, see Section 2.2). In Table 1, we compare the cost of a linearisation attack using the generic method of Arora-Ge, applicable to both LWE and LWR, with our work. As detailed in Section 2.3, the complexity of applying the Arora-Ge technique to LWR is at least $2^{q-p} \cdot \binom{n+2^{q-p}}{n}^{\omega}$. We look at the LSB of the sample – which corresponds to $m = 2^{q-p}$ – since our results show that this yields the lowest complexities. We provide a comparison with Arora-Ge using two metrics. First, we compare with the cost improved thanks to the rank drop using the formula $|\mathcal{Q}^{m,n}|^{\omega}$ (see the dashed curves of Fig. 2).[9] Next, we compare with the cost improved by additionally taking into account the average sparsity in this new representation. We estimate[10] the cost to be $2^{-3.42} \cdot p_{F^{m,n}} \cdot |\mathcal{Q}^{m,n}|^3$ using the Block-Wiedemann algorithm [46,25] where $p_{F^{m,n}}$ is the average number of non-zero terms per equation in the new linearisation family (see Fig. 1).

In Table 1, we propose comparisons for the linear algebra constants $\omega = 3$ and $\omega = 2$. This first choice seems fair in the sense that it has no hidden constants as compared to e.g. Strassen's algorithm, while the second is here for completeness as it makes sense from a design perspective (and fairness, as $\omega = 2$ is e.g. used in the lattice estimator by Albrecht et al. [4]). Since the square of the size of the linearization matrix is a strict lower bound on the complexity of Block-Wiedemann (see [31]), we do not compare with this method of resolution.

7.2 Open problems

Explaining the rank defect. In Section 6.2, we identified a rank defect as $|\mathcal{Q}_{m,n}|$ yields a strictly tighter upper bound on the rank than the number of monomials.

[9] We emphasize that we verified that our generating family $\mathcal{Q}^{m,n}$ allows to recover the secret as it contains all degree-1 monomials.

[10] The constant $2^{-3.42}$ is borrowed from [31, Section 5.1]. For a matrix with δ rows and s coefficients per row on average, the authors estimate the time complexity to be $6 \cdot s \cdot \delta^2/\ell_1$ elementary operations where ℓ_1 is the size of a word on the considered processor. As in [31], we use $\ell_1 = 64$ yielding $\log_2(6/64) \simeq 2^{-3.42}$.

m	n	[7]	Our work (rank only)	Our work (rank/sparsity)
8	64	$2^{103.4}$	$2^{97.8}$	$2^{87.2}$
	128	$2^{126.3}$	$2^{121.8}$	$2^{110.8}$
	256	$2^{149.7}$	$2^{145.9}$	$2^{134.7}$
16	64	$2^{167.7}$	$2^{157.2}$	N/A
	128	$2^{211.7}$	$2^{202.0}$	N/A
	256	$2^{257.5}$	$2^{250.1}$	N/A

m	n	[7]	Our work (rank only)
8	64	$2^{70.0}$	$2^{65.2}$
	128	$2^{85.2}$	$2^{81.2}$
	256	$2^{100.1}$	$2^{97.3}$
16	64	$2^{113.2}$	$2^{104.8}$
	128	$2^{142.4}$	$2^{134.6}$
	256	$2^{173.0}$	$2^{166.7}$

Table 1: Comparison to the linearisation attack by Arora and Ge [7] using $\omega = 3$ (left) and $\omega = 2$ (right) with $m = 2^{q-p}$.

However, determining the exact rank of the family $\{F_{\boldsymbol{a}}^{m,n}(\boldsymbol{x})\}_{\boldsymbol{a}\in\mathbb{Z}_{2q}^n}$ remains an open problem. We exploit $\mathfrak{S}_n$-invariance, which stems from the commutativity of addition ; a step towards a full understanding could be to exploit symmetries coming from the commutativity of multiplication. We refer to [11, Appendix C] for a first observation of a rank drop associated to this symmetry.

ANF of modular multiplication. It remains an open problem to find an exact formula for the ANF of modular addition, *i.e.* with no term cancellation as in Theorem 2. Our experiments suggest that the bound on the number of terms of $F^{m,n}$ given in Item **(i)** of Theorem 3 to be tight when m is a power of two.

On solving strategies. Recall that, by Item **(i)** of Theorem 3, a necessary condition for $\boldsymbol{x}^{\boldsymbol{u}}$ to appear in $F^{m,n}$ is that the coordinates of $\boldsymbol{u}$ sum to less than m. This implies that for any fixed i, u_i must satisfy $u_i \geq 2^j$ for $x_{i,j}$ to appear and thus the higher j is, the more "costly" $x_{i,j}$ is. These asymmetries in the role played by $x_{i,0}, \ldots, x_{i,q-1}$ may be beneficial to an adversary, for instance for guess and solve approaches (or hybrid strategies). As an example, guessing the LSBs $x_{i,0}$ seems more interesting than guessing more significant bits, e.g. because highest-degree monomials *exclusively* depend on them (since maximizing the number of variables, implies minimizing the cost of each of them). From a dual point of view, values of $\boldsymbol{a}$ such that most a_i are even (*i.e.* $a_{i,0} = 0$) lead to strictly less high-degree monomials. Time-data trade-offs could thus be considered, and be even more efficient for variants such as Ring or Mod-LWR where such a property is spread over several equations. Finally, a natural question is whether it would be possible to use other solving strategies. Gröbner bases, for example, could allow to attack with less samples.

This non-exhaustive list of examples suggests that the Boolean modeling of LWR leads to new attack paths that deserve further investigation.

Acknowledgments. François-Xavier Standaert is a research director of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in part by the ERC project 101096871 (acronym BRIDGE). Views and opinions

expressed are those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. This work was also supported by project Cryptanalyse from PEPR Cybersécurité (22-PECY-0010).

References

1. Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D., Liu, Y.K.: Status report on the third round of the nist post-quantum cryptography standardization process. Tech. Rep. NIST Internal Report (IR) 8413, Rev. 1, Includes updates as of September 9, 2022, National Institute of Standards and Technology, Gaithersburg, MD (2022). https://doi.org/10.6028/NIST.IR.8413-upd1
2. Alamati, N., Montgomery, H., Patranabis, S.: Symmetric primitives with structured secrets. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 650–679 (Aug 2019). https://doi.org/10.1007/978-3-030-26948-7_23
3. Albrecht, M.R., Cid, C., Faugère, J.C., Fitzpatrick, R., Perret, L.: Algebraic algorithms for LWE problems. ACM Commun. Comput. Algebra **49**(2), 62 (Aug 2015). https://doi.org/10.1145/2815111.2815158
4. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Math. Cryptol. **9**(3), 169–203 (2015). https://doi.org/10.1515/jmc-2015-0016
5. Alwen, J., Krenn, S., Pietrzak, K., Wichs, D.: Learning with rounding, revisited - new reduction, properties and applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 57–74 (Aug 2013). https://doi.org/10.1007/978-3-642-40041-4_4
6. Alwen, J., Krenn, S., Pietrzak, K., Wichs, D.: Learning with rounding, revisited - new reduction, properties and applications. In: CRYPTO (1). pp. 57–74. Lecture Notes in Computer Science, Springer (2013)
7. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 403–415. Springer, Berlin, Heidelberg (Jul 2011). https://doi.org/10.1007/978-3-642-22006-7_34
8. Bak, A.: A note on the rank defect phenomena in the linearization attack on elisabeth-4. Cryptology ePrint Archive, Paper 2025/1133 (2025), https://eprint.iacr.org/2025/1133
9. Banerjee, A., Brenner, H., Leurent, G., Peikert, C., Rosen, A.: SPRING: Fast pseudorandom functions from rounded ring products. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 38–57. Springer, Berlin, Heidelberg (Mar 2015). https://doi.org/10.1007/978-3-662-46706-0_3
10. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737 (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_42
11. Baudrin, J., Boissier, R.H., Standaert, F.X.: On the concrete hardness of LWR with a power of two modulus. Cryptology ePrint Archive, Paper 2026/250 (2026), https://eprint.iacr.org/2026/250
12. Beyne, T., Verbauwhede, M.: Integral cryptanalysis using algebraic transition matrices. IACR Trans. Symm. Cryptol. **2023**(4), 244–269 (2023). https://doi.org/10.46586/tosc.v2023.i4.244-269

13. Bogdanov, A., Guo, S., Masny, D., Richelson, S., Rosen, A.: On the hardness of learning with rounding over small modulus. In: Kushilevitz, E., Malkin, T. (eds.) Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I. pp. 209–224. Lecture Notes in Computer Science, Springer (2016). https://doi.org/10.1007/978-3-662-49096-9_9, https://doi.org/10.1007/978-3-662-49096-9_9
14. Boneh, D., Lewi, K., Montgomery, H.W., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 410–428 (Aug 2013). https://doi.org/10.1007/978-3-642-40041-4_23
15. Bostan, A., Chyzak, F., Giusti, M., Lebreton, R., Lecerf, G., Salvy, B., Schost, E.: Algorithmes Efficaces en Calcul Formel. Frédéric Chyzak (self-edited), Palaiseau (Sep 2017), https://hal.archives-ouvertes.fr/AECF/, 686 pages.
16. Boura, C., Canteaut, A.: Another view of the division property. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 654–682 (Aug 2016). https://doi.org/10.1007/978-3-662-53018-4_24
17. Boura, C., Canteaut, A., De Cannière, C.: Higher-order differential properties of Keccak and Luffa. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 252–269. Springer, Berlin, Heidelberg (Feb 2011). https://doi.org/10.1007/978-3-642-21702-9_15
18. Braeken, A., Semaev, I.: The ANF of the composition of addition and multiplication mod 2^n with a Boolean function. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 112–125. Springer, Berlin, Heidelberg (Feb 2005). https://doi.org/10.1007/11502760_8
19. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012. pp. 309–325. ACM (Jan 2012). https://doi.org/10.1145/2090236.2090262
20. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 575–584. ACM Press (Jun 2013). https://doi.org/10.1145/2488608.2488680
21. Calderón-Gómez, J.E., Medina, L.A., Molina-Salazar, C.A.: Short k-rotation symmetric boolean functions. Discret. Appl. Math. **343**, 49–64 (2024). https://doi.org/10.1016/J.DAM.2023.10.003
22. Canteaut, A.: Lecture Notes on Cryptographic Boolean Functions (2016)
23. Chen, Y., Ji, L., Li, W.: Learning with Alternating Moduli, Arora-Ge over Composite Moduli, and Weak PRFs. In: Heninger, N., Rosulek, M. (eds.) Advances in Cryptology - CRYPTO 2026 (Aug 2026), Part III. LNCS, vol. 16802, pp. 201–231. Springer (2026). https://doi.org/10.1007/978-3-032-35377-1_7
24. Chuengsatiansup, C., Stehlé, D.: Towards practical GGM-based PRF from (module-)learning-with-rounding. In: Paterson, K.G., Stebila, D. (eds.) SAC 2019. LNCS, vol. 11959, pp. 693–713 (Aug 2019). https://doi.org/10.1007/978-3-030-38471-5_28
25. Coppersmith, D.: Solving homogeneous linear equations over gf(2) via block wiedemann algorithm. Mathematics of Computation **62**(205), 333–350 (1994), http://www.jstor.org/stable/2153413
26. D'Anvers, J.P., Karmakar, A., Roy, S.S., Vercauteren, F.: Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 18. LNCS, vol. 10831, pp. 282–305 (May 2018). https://doi.org/10.1007/978-3-319-89339-6_16

27. Dummit, D.S., Foote, R.M.: Abstract Algebra, 3rd Edition. John Wiley & Sons Inc (2003)
28. Dziembowski, S., Faust, S., Herold, G., Journault, A., Masny, D., Standaert, F.X.: Towards sound fresh re-keying with hard (physical) learning problems. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 272–301 (Aug 2016). https://doi.org/10.1007/978-3-662-53008-5_10
29. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra (3. ed.). Cambridge University Press (2013)
30. Geihs, M., Montgomery, H.: Lakey: Efficient lattice-based distributed prfs enable scalable distributed key management. In: USENIX Security Symposium. USENIX Association (2024)
31. Gilbert, H., Boissier, R.H., Jean, J., Reinhard, J.R.: Cryptanalysis of elisabeth-4. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part III. LNCS, vol. 14440, pp. 256–284 (Dec 2023). https://doi.org/10.1007/978-981-99-8727-6_9
32. Hebborn, P., Lambin, B., Leander, G., Todo, Y.: Lower bounds on the degree of block ciphers. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 537–566 (Dec 2020). https://doi.org/10.1007/978-3-030-64837-4_18
33. Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 446–476 (Dec 2020). https://doi.org/10.1007/978-3-030-64837-4_15
34. Hu, K., Yap, T.: Perfect monomial prediction for modular addition. IACR Transactions on Symmetric Cryptology **2024**(3), 177–199 (Sep 2024). https://doi.org/10.46586/tosc.v2024.i3.177-199
35. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. DCC **75**(3), 565–599 (2015). https://doi.org/10.1007/s10623-014-9938-4
36. Lidl, R., Niederreiter, H.: Finite Fields. Encyclopedia of Mathematics and its Applications, Cambridge University Press, 2 edn. (1996). https://doi.org/10.1017/CBO9780511525926
37. Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlé, D., Bai, S.: CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation. NIST Post-Quantum Cryptography Standard (2022)
38. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23 (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_1
39. Mouha, N., Velichkov, V., De Cannière, C., Preneel, B.: The differential analysis of S-functions. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 36–56 (Aug 2011). https://doi.org/10.1007/978-3-642-19574-7_3
40. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005). https://doi.org/10.1145/1060590.1060603
41. Schönhage, A., Strassen, V.: Schnelle multiplikation großer zahlen. Computing **7**, 281–292 (1971). https://doi.org/10.1007/BF02242355
42. Schwabe, P., Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Seiler, G., Stehlé, D., Ding, J.: CRYSTALS-Kyber Algorithm Specifications and Supporting Documentation. NIST Post-Quantum Cryptography Standard (2022)

43. Steiner, M.J.: The complexity of algebraic algorithms for LWE. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part III. LNCS, vol. 14653, pp. 375–403 (May 2024). https://doi.org/10.1007/978-3-031-58734-4_13
44. Sun, C., Tibouchi, M., Abe, M.: Revisiting the hardness of binary error LWE. In: ACISP. Lecture Notes in Computer Science, vol. 12248, pp. 425–444. Springer (2020)
45. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 287–314 (Apr 2015). https://doi.org/10.1007/978-3-662-46800-5_12
46. Wiedemann, D.: Solving sparse linear equations over finite fields. IEEE transactions on information theory **32**(1), 54–62 (1986)
47. Zhang, Y., Lu, X., Liu, Y., Yin, Y., Wang, K.: LEAP: high-performance lattice-based pseudorandom number generator. IACR Trans. Symmetric Cryptol. **2025**(3), 151–182 (2025)