

Cryptographic applications of combinatorial ranking for integer compositions

Gustavo Zambonin¹[0000–0002–6962–9347], Larissa Gremelmaier Rosa¹[0009–0000–1238–2489], Ricardo Custódio¹[0000–0001–9611–5694], and Daniel Panario²[0000–0003–3551–4063]

¹ Universidade Federal de Santa Catarina, Florianópolis, SC, 88040-900, Brazil
{gustavo.zambonin,larissa.gremelmaier.rosa}@posgrad.ufsc.br,
ricardo.custodio@ufsc.br

² Carleton University, Ottawa, ON, K1S 5B6, Canada
daniel@math.carleton.ca

Abstract. Bounded integer compositions are fundamental combinatorial objects employed in cryptography, yet they lack a unified theoretical treatment and efficient implementations for combinatorial (un)ranking. We propose generic algorithms that allow us to describe proposals found in the literature as specific parameter sets. Consequently, we are able to provide a single proof of correctness that applies simultaneously to all derived algorithms; characterize a number of ad hoc optimizations with respect to our generic framework; and discuss the storage complexity of pre-computed combinatorial count tables. Furthermore, drawing on Gaussian limit laws from analytic combinatorics, we propose performance and storage optimizations for the particular case of uniform random inputs to (un)ranking algorithms, a common setting in cryptographic applications. To validate our results, we give portable low-level implementations ready for use by algorithm designers, filling a practical gap in the literature.

Keywords: Uniform sampling · Gaussian law · Enumerative coding

1 Introduction

Combinatorial ranking provides a standard method for the uniform sampling and optimal compression of ordered structures, which are fundamental problems in cryptographic algorithms. In particular, ranking of (bounded) integer compositions is employed in (i) hash-based signatures, (ii) format-preserving encryption, and (iii) code-based cryptosystems. We expand on these use cases below, and argue how a unified approach joins the perspectives of the literature into a systematic framework of ranking compositions in applied cryptography.

In hash-based signatures like SLH-DSA (formerly SPHINCS⁺) [25], mapping messages to bounded compositions is standard practice to improve performance and signature sizes [14]. While fundamental (un)ranking algorithms are frequently rediscovered [26,11,29], recent research has focused on optimizations like rejection sampling [12] and parameter tuning [35,14,27] to reduce overhead.

Format-preserving encryption (FPE) ensures plaintexts and ciphertexts maintain the same structural constraints, often via the *rank-then-encipher* paradigm introduced in [4]. In *sum-preserving* (SPE) schemes, messages are bounded compositions, which is relevant for thumbnail-preserving encryption [31], and statistical analysis of encrypted data [21].

Finally, a number of code-based cryptosystems require mapping binary strings to fixed-weight error vectors [16,19]. Although deterministic unranking has largely been superseded by randomized, constant-time alternatives [3] to mitigate side-channel risks, algorithmic improvements could reinstate combinatorial ranking as a competitive, deterministic solution.

Contributions. We present a comprehensive framework for the combinatorial ranking of bounded integer compositions. By parameterizing the problem into a strategy composed of three functions, focused on this particular structure, we are able to (i) recover classical algorithms and correctness proofs as specific instances of our framework; (ii) analyze common optimizations such as bisection search and partial sum re-usage; (iii) propose a probabilistic strategy that exploits the statistical structure of random compositions, suitable for cryptographic algorithms. We also measure the performance and storage requirements of several (un)ranking algorithms and pre-computed combinatorial count tables.

2 Preliminaries

Notation. We write $x \stackrel{r}{\leftarrow} X$ to mean x is sampled uniformly at random from X , $[x^j]f(x)$ to be the coefficient of x^j in the generating function f , $[d] = \{0, 1, \dots, d\}$, and $\lg = \log_2$. For any statement P , $[[P]]$ is the Iverson bracket, where $[[P]] = 1$ if P is true and 0 otherwise. Let $()$ be the empty sequence. Given sequences $Y = (y_1, y_2, \dots, y_\alpha)$ and $W = (w_1, w_2, \dots, w_\beta)$ such that $|Y| = \alpha$ and $|W| = \beta$, let $y \oplus Y = (y, y_1, \dots, y_\alpha)$, $Y \oplus y = (y_1, \dots, y_\alpha, y)$, $Y \oplus W = (y_1, \dots, y_\alpha, w_1, \dots, w_\beta)$ and $Y^R = (y_\alpha, \dots, y_1)$.

Combinatorial ranking & compositions. Let S be a finite set, $N = |S|$ and $s, t \in S$ such that $s \neq t$. A *ranking function* is a bijection $\mathbf{R} : S \rightarrow [N-1]$ that defines a total order $\prec_{\text{ord}}$ on the elements of S , i.e., $s \prec_{\text{ord}} t$ if and only if $\mathbf{R}(s) < \mathbf{R}(t)$ [17, Section 2.1]. Given a ranking function, the *unranking function* is the inverse bijection $\mathbf{U} = \mathbf{R}^{-1}$, i.e., $\mathbf{U} : [N-1] \rightarrow S$ such that $\mathbf{R}(s) = i$ if and only if $\mathbf{U}(i) = s$ for all $i \in [N-1]$.

Consider $n, k \in \mathbb{Z}_{>0}$. A *weak k -composition* of n is a sequence $(a_1, \dots, a_k)$ such that $\sum_{i=1}^k a_i = n$ and each part $a_i \in \mathbb{Z}_{\geq 0}$. An *S -bounded composition* with $S \subseteq \mathbb{Z}_{\geq 0}$ is a composition of n such that $a_i \in S$. If $S = [d]$ for $d \in \mathbb{Z}_{>0}$, we denote the compositions d -bounded [5]. We denote the set of d -bounded k -compositions of n by $\mathcal{C}_d(n, k)$, and its cardinality by $\mathfrak{C}_d(n, k)$, which is expressed by the generating function below.

Lemma 1 ([8, I.15]). $\mathfrak{C}_d(n, k) = [x^n] \left(\frac{1-x^{d+1}}{1-x} \right)^k$.

The polynomial interpretation yields a double combinatorial sum, deemed a “simple binomial convolution” [8, p. 43] and originally attributed to [22].

Lemma 2 ([1, Remark 60]). $\mathfrak{C}_d(n, k) = \sum_{i=0}^k (-1)^i \binom{k}{i} \binom{n-(d+1)i+k-1}{k-1}$.

We consider the upper limit of the summation to be $j = \min(k, \lfloor \frac{n}{d+1} \rfloor)$, which prevents unnecessary computations, and hereafter denote $\mathfrak{C}_d(n, k)_\alpha$ to be the α -th term of the right-hand side of Lemma 2, for $\alpha \in [j]$. The cardinality may be computed by two recursive formulas that generalize Pascal’s and Vandermonde’s identities, respectively.

Theorem 1 ([1, Theorem 11]). *If $k = 1$ and $n > d$, $\mathfrak{C}_d(n, k) = 0$. If $k = 1$ and $n \leq d$, $\mathfrak{C}_d(n, k) = 1$. Otherwise, $\mathfrak{C}_d(n, k) = \sum_{i=0}^d \mathfrak{C}_d(n-i, k-1)$.*

Theorem 2 ([6, Table 1]). *Let $k_1, k_2 \in \mathbb{Z}_{\geq 0}$ such that $k_1 + k_2 = k$. If $k = 1$ and $n > d$, $\mathfrak{C}_d(n, k) = 0$. If $k = 1$ and $n \leq d$, $\mathfrak{C}_d(n, k) = 1$. Otherwise, $\mathfrak{C}_d(n, k) = \sum_{n_1=0}^n \mathfrak{C}_d(n_1, k_1) \cdot \mathfrak{C}_d(n-n_1, k_2)$.*

For further information on bounded integer compositions, we refer the reader to [6] for historical perspectives and applications to other disciplines.

Combinatorial orders. We recall that for any set S of size N , there exist $N!$ total orders for S , corresponding to the number of distinct permutations of its elements. We give several descriptions of usual total orders found in the literature, following the “constructive” notation of [2, Section 14.1]. We use bold letters (e.g., $\mathbf{C}_d(n, k)$) to denote a permutation of $\mathcal{C}_d(n, k)$ related to a specific order.

The canonical total order is *lexicographic*, or “dictionary” order, which aligns naturally with the recursive structure of Theorem 1. A common variant, defined below, can be intuitively understood as “lexicographic read from right to left”.

Definition 1. *The colexicographic order, or $\prec_{\text{colex}}$, is defined as follows. If $k = 1$ and $n > d$, $\mathbf{C}_d(n, 1) = ()$. If $k = 1$ and $n \leq d$, $\mathbf{C}_d(n, 1) = (n)$. Otherwise,*

$$\mathbf{C}_d(n, k) = (\mathbf{C}_d(n, k-1) \oplus 0, \mathbf{C}_d(n-1, k-1) \oplus 1, \dots, \mathbf{C}_d(n-d, k-1) \oplus d).$$

We obtain the *reflected Gray code* [32, Theorem 1], or Eades-McKay [2, Section 6.5] order, by reversing the $\mathbf{C}_d(n-1, k-1)$ substructure at odd d . This order, which we denote by $\prec_{\text{gray}}$, is *minimal-change*: two consecutive compositions differ only by two parts [2, Section 6.4].

We can also order compositions via the structure of Theorem 2, as proposed recently in [21], but first connected to cryptographic applications in [29]. Let $\mathbf{l} = (l_i)_{i=1}^p$, $\mathbf{r} = (r_j)_{j=1}^q$ be any two ordered lists of sequences with $p, q \in \mathbb{Z}_{\geq 0}$. The “product” operation is defined as $\mathbf{l} \odot \mathbf{r} = \oplus_{i=1}^p (\oplus_{j=1}^q (l_i \oplus r_j))$, or $\mathbf{l} \odot \mathbf{r} = ()$ if $p = 0$ or $q = 0$. Explicitly, this produces the sequence $(l_1 \oplus r_1, \dots, l_1 \oplus r_q, \dots, l_p \oplus r_1, \dots, l_p \oplus r_q)$.

Definition 2. *The recursive block order, or $\prec_{\text{rb}}$, is defined as follows. If $k = 1$ and $n > d$, $\mathbf{R}_d(n, 1) = ()$. If $k = 1$ and $n \leq d$, $\mathbf{R}_d(n, 1) = (n)$. Otherwise, for $k_l = \lfloor \frac{k}{2} \rfloor$ and $k_r = k - k_l$,*

$$\mathbf{R}_d(n, k) = \bigoplus_{n_l=0}^{\min(n, k_l \cdot d)} (\mathbf{R}_d(n_l, k_l) \odot \mathbf{R}_d(n-n_l, k_r)).$$

It is essentially a lexicographic ordering of the subproblems defined by Theorem 2 for k_1, k_2 split exactly at $\frac{k}{2}$. We note that $\prec_{\text{rb}}$ is remarkably connected to the *boustrophedonic* order [20], hereafter denoted $\prec_{\text{bous}}$. They differ only in how n_l and n_r are chosen; while for $\prec_{\text{rb}}$, n_l grows monotonically from 0 to $\min(n, k_l \cdot d)$, for $\prec_{\text{bous}}$ the iteration sequence is $0, n, 1, n-1, \dots$, giving rise to a more complex recursive case:

$$\mathbf{B}_d(n, k) = \bigoplus_{j=0}^{\lfloor \frac{n}{2} \rfloor} \left((\mathbf{B}_d(j, k_l) \odot \mathbf{B}_d(n-j, k_r)) \oplus \tau \right),$$

where $\tau = \mathbf{B}_d(n-j, k_l) \odot \mathbf{B}_d(j, k_r)$ if $j < n-j$, and $()$ if $j = n-j$; see [9].

3 Generic (un)ranking algorithms

The design of (un)ranking algorithms is closely linked to the total order defined on the target combinatorial class. For instance, Theorem 1 implies that an unranking algorithm for $\prec_{\text{colex}}$ must essentially determine

$$\min \left\{ p \in [\min(n', d)] : \sum_{j=0}^p \mathfrak{C}_d(n' - j, i) > r' \right\} \quad (1)$$

for every part $i \in (k-1, \dots, 1)$, while tracking the partial sum n' and remaining rank r' . (Of course, p is the i -th part of the resulting composition.) Historically, this iterative approach has dominated the literature, due to its intuitive definition and ease of verification [15, p. 358].

The generalized framework proposed in [20,23] unified the derivation of (un)ranking algorithms for decomposable combinatorial structures, via the symbolic method of analytic combinatorics [8]. However, it prioritizes universality by relying on fixed orders ($\prec_{\text{lex}}$ and $\prec_{\text{bous}}$). As a result, research into alternative orders remains sparse, despite evidence that order choice significantly impacts the performance of enumeration and (un)ranking algorithms [20, Theorem 2].

Inspired by this generic framework, we take a complementary approach; rather than fixing the order to derive an algorithm for a generic class, we fix the combinatorial object (bounded compositions) and treat the *strategy* as a variable parameter. We let a strategy be a tuple of functions $\Sigma = (\mathbf{D}, \mathbf{P}, \mathbf{S})$, where the components are formally defined as follows.

Decomposition. The function $\mathbf{D} : \mathbb{Z}_{\geq 1} \rightarrow \mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0}$; $k \mapsto (k_l, k_r)$ determines the recursive structure of the algorithms by mapping a problem size k to a pair of smaller sizes satisfying $k_l + k_r = k$. Examples are $\mathbf{D}_l(k) = (1, k-1)$, $\mathbf{D}_r(k) = (k-1, 1)$ and $\mathbf{D}_s(k) = (\lfloor \frac{k}{2} \rfloor, \lceil \frac{k}{2} \rceil)$.

Path. The function $\mathbf{P} : \mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0} \rightarrow \bigcup_{m=0}^{n+1} (\mathbb{Z}_{\geq 0})^m$; $(n, k_l, k_r) \mapsto \mathbf{w}$ generates the traversal schedule. It maps the current context to a sequence of values to visit. The output $\mathbf{w} = (w_1, w_2, \dots)$ must be a permutation of the valid range $\{n_1 \in [n] : \mathfrak{C}_d(n_1, k_l) > 0 \wedge \mathfrak{C}_d(n - n_1, k_r) > 0\}$. Examples are $\mathbf{P}_{\text{inc}}(n, k_l, k_r) = (w_1, \dots, w_m)$, $\mathbf{P}_{\text{dec}}(n, k_l, k_r) = (w_m, w_{m-1}, \dots, w_1)$ and $\mathbf{P}_{\text{alt}}(n, k_l, k_r) = (w_1, w_m, w_2, w_{m-1}, \dots)$.

Remark 1. We restrict $\mathbf{D}$ to binary splits instead of a full m -ary decomposition $k \rightarrow (k_1, \dots, k_m)$. The generic decomposition would require $\mathbf{P}$ to iterate over all m -compositions of n , increasing the size of the local search space to $\mathcal{O}(n^{m-1})$. Since any m -ary decomposition can be represented as a tree of binary splits with constant overhead, the binary restriction bounds the iteration size of $\mathbf{P}$ to $\mathcal{O}(n)$ at each step without loss of generality.

Split. The component $\mathbf{S}$ is a family of bijections indexed by dimensions $(u, v, w) \in \mathbb{Z}_{\geq 1}^2 \times \mathbb{Z}_{\geq 0}$. The function $\mathbf{S}^{(u,v,w)} : [uv - 1] \rightarrow [u - 1] \times [v - 1]$; $r \mapsto (r_l, r_r)$ decomposes a rank r to sub-ranks r_l, r_r satisfying $r_l + r_r = r$ based on the value w . Examples are $\mathbf{S}_{\text{div}}^{(u,v,w)}(r) = (\lfloor \frac{r}{v} \rfloor, r \bmod v)$, the usual specialization $\mathbf{S}_{\text{id}}^{(1,v,w)}(r) = (0, r)$, and $\mathbf{S}_{\text{g}}^{(1,v,w)}(r) = (0, r)$ if $w \equiv 0 \pmod{2}$ and $(0, v - 1 - r)$ otherwise. Specific instantiations may discard certain parameters if they are not required by the algebraic structure of the split.

Recursive construction. The ordered sequence of compositions $\mathcal{C}_d(n, k)$ generated by a strategy Σ , denoted $\mathbf{Z}_d(n, k)$, is defined recursively. We first extend the product operation $\odot$ to account for the operation of $\mathbf{S}$. Let $\mathbf{L}$ and $\mathbf{R}$ be ordered lists of compositions with sizes u and v , respectively. The *permuted product* $\mathbf{L} \odot_{\mathbf{S}} \mathbf{R}$ is the list of size uv where the r -th element is $L_{r_l} \oplus R_{r_r}$, with $(r_l, r_r) = \mathbf{S}^{(u,v,w)}(r)$ for $r \in [uv - 1]$. For $(k_l, k_r) = \mathbf{D}(k)$, the generic sequence is

$$\mathbf{Z}_d(n, k) = \bigoplus_{w_l \in \mathbf{P}(n, k_l, k_r)} \left(\mathbf{Z}_d(w_l, k_l) \odot_{\mathbf{S}} \mathbf{Z}_d(n - w_l, k_r) \right).$$

Standard instantiations. The strategy Σ can be instantiated to recover standard algorithms. We denote Σ_{ord} as the strategy “corresponding to $\prec_{\text{ord}}$ ”. The same convention is applied to (un)ranking functions. We give a number of strategies as follows: (i) $\Sigma_{\text{lex}} = (\mathbf{D}_l, \mathbf{P}_{\text{inc}}, \mathbf{S}_{\text{id}})$; (ii) $\Sigma_{\text{colex}} = (\mathbf{D}_r, \mathbf{P}_{\text{inc}}, \mathbf{S}_{\text{id}})$; (iii) $\Sigma_{\text{gray}} = (\mathbf{D}_l, \mathbf{P}_{\text{inc}}, \mathbf{S}_{\text{g}})$; (iv) $\Sigma_{\text{rb}} = (\mathbf{D}_s, \mathbf{P}_{\text{inc}}, \mathbf{S}_{\text{div}})$; and (v) $\Sigma_{\text{bous}} = (\mathbf{D}_s, \mathbf{P}_{\text{alt}}, \mathbf{S}_{\text{div}})$.

Formalization. All components of the generic (un)ranking algorithms are mechanized in Lean 4 [24]. We prove their correctness via strong induction, and instantiate the Σ_{colex} , Σ_{rb} and Σ_{gray} strategies to show that the framework is sound. We also prove the auxiliary propositions, lemmas and theorems discussed in the following sections, which might be of independent interest. The complete artifact is available at [34].

We present the generic unranking and ranking algorithms for bounded integer compositions in Algorithm 1, respectively. We assume that the parameters n, k, d are valid, such that $\mathfrak{C}_d(n, k) > 0$. Then, Theorem 3 states that the algorithms are mutual inverses, i.e., they establish a strict bijection between the set of combinatorial objects and their indices. In other words, composing the two operations yields the identity function on both domains, guaranteeing that the enumeration is exhaustive and collision-free.

Theorem 3. *Let $S = \mathbf{Z}_d(n, k)$. Then, for any $z \in S$ and $r \in [|S| - 1]$,*

$$\mathbf{R}_{\Sigma}(n, k, d, \mathbf{U}_{\Sigma}(n, k, d, r)) = r \quad \text{and} \quad \mathbf{U}_{\Sigma}(n, k, d, \mathbf{R}_{\Sigma}(n, k, d, z)) = z.$$

$\mathbf{R}_\Sigma(\text{in: } n, k, d, z; \text{ out: } r).$

1. If $k = 1$, set $r \leftarrow 0$ and exit.
Otherwise, go to Step 2.
2. Set $(k_l, k_r) \leftarrow \mathbf{D}(k)$.
3. Set $w_l \leftarrow \sum_{i=1}^{k_l} z_i$.
4. Set $L \leftarrow (z_1, \dots, z_{k_l})$.
5. Set $R \leftarrow (z_{k_l+1}, \dots, z_k)$.
6. Set $\mathbf{w} \leftarrow \mathbf{P}(n, k_l, k_r)$.
7. Set $r \leftarrow 0$.
8. For each $v \in \mathbf{w}$:
 - 8A. If $v = w_l$, go to Step 9.
 - 8B. Set $r \leftarrow r + \mathfrak{C}_d(v, k_l) \cdot \mathfrak{C}_d(n - v, k_r)$.
9. Set $r_l \leftarrow \mathbf{R}_\Sigma(w_l, k_l, d, L)$.
10. Set $r_r \leftarrow \mathbf{R}_\Sigma(n - w_l, k_r, d, R)$.
11. Set $n_l \leftarrow \mathfrak{C}_d(w_l, k_l)$.
12. Set $n_r \leftarrow \mathfrak{C}_d(n - w_l, k_r)$.
13. Set $r \leftarrow r + (\mathbf{S}^{(n_l, n_r, w_l)})^{-1}(r_l, r_r)$.

$\mathbf{U}_\Sigma(\text{in: } n, k, d, r; \text{ out: } z).$

1. If $k = 1$, set $z \leftarrow (n)$ and exit.
Otherwise, go to Step 2.
2. Set $(k_l, k_r) \leftarrow \mathbf{D}(k)$.
3. Set $\mathbf{w} \leftarrow \mathbf{P}(n, k_l, k_r)$.
4. For each $w_l \in \mathbf{w}$:
 - 4A. Set $n_l \leftarrow \mathfrak{C}_d(w_l, k_l)$.
 - 4B. Set $n_r \leftarrow \mathfrak{C}_d(n - w_l, k_r)$.
 - 4C. Set $V \leftarrow n_l \cdot n_r$.
 - 4D. If $r < V$, go to Step 5.
 - 4E. Set $r \leftarrow r - V$.
5. Set $(r_l, r_r) \leftarrow \mathbf{S}^{(n_l, n_r, w_l)}(r)$.
6. Set $L \leftarrow \mathbf{U}_\Sigma(w_l, k_l, d, r_l)$.
7. Set $R \leftarrow \mathbf{U}_\Sigma(n - w_l, k_r, d, r_r)$.
8. Set $z \leftarrow L \oplus R$.

Algorithm 1. Generic ranking and unranking algorithms parameterized by a strategy $\Sigma = (\mathbf{D}, \mathbf{P}, \mathbf{S})$. We recall that $n, k, d \in \mathbb{Z}_{>0}$, $z \in \mathfrak{C}_d(n, k)$ and $r \in [\mathfrak{C}_d(n, k) - 1]$.

We now discuss how specific choices for $\Sigma = (\mathbf{D}, \mathbf{P}, \mathbf{S})$ determine the efficiency of the resulting (un)ranking algorithms. We focus our analysis on $\mathbf{D}$ and $\mathbf{P}$, as we observe that $\mathbf{S}$ performs only elementary arithmetic, with negligible cost compared to the search in $\mathbf{P}$ or how $\mathbf{D}$ controls the recursion structure.

3.1 Decomposition and storage requirements

The repeated evaluation of $\mathfrak{C}_d(n, k)$ for arbitrary n, k, d is not computationally trivial. In combinatorial literature, it is often assumed that needed counts are already pre-computed [23, Section 3.2.1]. On the other hand, cryptographic applications require careful management of the storage requirements associated with the usage of (un)ranking functions [26, 21, 36], particularly if arbitrary-precision integers are required to represent $\mathfrak{C}_d(n, k)$.

We provide a systematic view of this *tabulation* mechanism that complements the empirical evidence in the literature, and correlate it with the decomposition function $\mathbf{D}$ of a strategy. We define $\chi_{\text{comb}}^{(d)}(n, k)$ as the $(k + 1) \times (n + 1)$ matrix with entries $(\chi_{\text{comb}}^{(d)})_{i,j} = \mathfrak{C}_d(j, i)$ for $0 \leq i \leq k$ and $0 \leq j \leq n$. If $d = 1$, the entries are the binomial coefficients $\binom{n}{k}$, and we denote the table by $\chi_{\text{bin}}(n, k)$. We drop the parameters from the notation if clear from the context.

Naturally, the choice of $\mathbf{D}$ controls which rows of this matrix are used by the (un)ranking algorithms. Consider $\mathbf{D}_l(k) = (1, k - 1)$ and $\mathbf{D}_r(k) = (k - 1, 1)$. In such “linear” orders, the recursion must solve problems of all sizes in the sequence $(1, \dots, k)$. In other words, counts of the form $\mathfrak{C}_d(n', i)$ for all $1 \leq i \leq k - 1$ and any partial sum $n' < n$ must be in χ_{comb} .

Remark 2. For the aforementioned “linear” orders, one can extend the table χ_{comb} to store cumulative partial sums of $\mathfrak{C}_d(n, k)$. However, this extension is only useful if $\mathbf{P}$ is specifically designed to utilize these aggregates. In the literature, this trade-off has been empirically explored in [26, Section 3.4]; we discuss the necessary interaction between storage and strategy in Section 3.2.

On the other hand, for $\mathbf{D}_s(k) = (\lfloor \frac{k}{2} \rfloor, \lceil \frac{k}{2} \rceil)$, only a sparse subset of counts is needed. While the reduced memory footprint of using $\prec_{\text{rb}}$ (and therefore $\mathbf{D}_s$) was empirically observed [21, Section 5], the authors did not provide a theoretical bound. We formalize this advantage by showing the exact set of “row” indices visited by (un)ranking algorithms whose decomposition strategy is $\mathbf{D}_s$.

Definition 3. Let $k \in \mathbb{Z}_{>0}$ and $\ell = \lfloor \lg k \rfloor$. The iterated halving set of k is $\mathbf{H}(k) = \bigcup_{p=1}^{\ell} \{\lfloor \frac{k}{2^p} \rfloor, \lceil \frac{k}{2^p} \rceil\}$, with $\mathbf{H}(1) = \emptyset$.

It is straightforward to evaluate $\mathbf{H}(k)$ via the binary representation of k . Nonetheless, we provide a closed expression for its exact size, allowing precise estimates of the size of χ_{comb} as a function of parameters n, k, d for orders based on $\mathbf{D}_s(k)$. We let $\nu_2(k) = \max\{j \in \mathbb{Z}_{\geq 0} : 2^j \mid k\}$ be the 2-adic valuation of k .

Proposition 1. For $k \in \mathbb{Z}_{>0}$ and $\ell = \lfloor \lg k \rfloor$, we have that

$$|\mathbf{H}(k)| = 2\ell - \nu_2(k) - \lfloor [2^\ell < k < 2^\ell + 2^{\ell-1}] \rfloor.$$

Practical considerations on the storage requirements of the tabling mechanisms are further explored in Section 4.

3.2 Path and algebraic search

We discuss two optimization techniques for the case when $\mathbf{P}$ produces a monotonic sequence, for instance $\mathbf{P}_{\text{inc}}(n, k_l, k_r) = (w_1, \dots, w_m)$ which is featured in $\prec_{\text{colex}}$. The search then reduces to finding an index in an ordered sequence.

The “bisect” technique. We observe that if $\mathbf{P}$ is monotonic, Equation (1) represents a search over a sorted sequence of partial sums, defined as

$$A(n', i, d) = \left(0, \mathfrak{C}_d(n', i), \dots, \sum_{j=0}^{\min(n', d)} \mathfrak{C}_d(n' - j, i)\right) \quad (2)$$

for $1 \leq i \leq k - 1$ and $n' \in [n]$. Consequently, the correct value w (or part p) can be found via binary search in $\log d$ instead of d steps. This approach is mentioned in [21, Section 4.4] and explored in [26, Section 3.2], who provide a closed-form identity to compute arbitrary terms of this sequence directly. We identified a substitution error in the proof [26, Section A.3], which actually yielded $\mathfrak{C}_d(n, k + 1)^{(l)}$. The formula is fixed here by incrementing k in the summation.

Lemma 3 ([26, Proposition 3]). Let $n, k, d \in \mathbb{Z}_{>0}$, $j = \min(k, \lfloor \frac{n}{d+1} \rfloor)$, $l \in [\min(n, d)]$ and set $u = n - (d + 1)i + k - 1$. Then,

$$\mathfrak{C}_d(n, k)^{(l)} = \sum_{i=0}^j (-1)^i \binom{k}{i} \left[\binom{u}{k} - \binom{u-l}{k} \right].$$

However, we identify redundant computations in the unranking algorithm proposed by the authors in [26]. Lemma 3 requires summing $\mathcal{O}(k)$ binomial terms; since the algorithm re-evaluates them independently at every step of the binary search, the computational cost to find a single part is $\mathcal{O}(k \log d)$. We argue that a direct bisection search on the sequence $A(n', i, d)$ is more efficient, as it can be constructed iteratively in $\mathcal{O}(d)$ steps.

Algorithms implementing this technique can make use of the tabling mechanism discussed in Remark 2. We define $\chi_{\text{acc}}^{(d)}(n, k)$ to be a matrix holding accumulated sizes of composition sets, with entries $(\chi_{\text{acc}}^{(d)})_{i,j} = A(j, i, d)$ for $0 \leq i \leq k$ and $0 \leq j \leq n$. As the size of each entry, itself a sequence, depends on the choice of parameters, we give a simple formula to calculate the number of entries of the entire matrix, such that it can be effectively packed.

Proposition 2. *The size of $\chi_{\text{acc}}^{(d)}(n, k)$ is $(k+1) \left\lceil \frac{d(2n-d+1)}{2} + 2(n+1) \right\rceil$.*

We discuss the trade-offs of using χ_{acc} with respect to the performance of compatible unranking algorithms, and the general behavior of the “bisect” technique, in Section 4.

The “partial sum” technique. If a binary search is not employed, the choice of P_{inc} still allows for a different optimization. Instead of computing the combinatorial count $\mathfrak{C}_d(n' - p, i)$ from scratch at each step of Equation (1), we can derive it from the previous term using an arithmetic ratio, which is a generalization of [10, Section 4.2]. A similar technique is found in [30, Section 1.13], in which the input to unranking is a number in $[0, 1)$ and no large integers are required.

Proposition 3. *Let $n, k, d \in \mathbb{Z}_{>0}$ and $j \in [\min(i, \lfloor \frac{n'-p-1}{d+1} \rfloor)]$. Then,*

$$\frac{\mathfrak{C}_d(n', i)_j}{\mathfrak{C}_d(n' - p, i)_j} = \prod_{g=1}^p \frac{(n' - g + 1) - (d+1)j + i - 1}{(n' - g + 1) - (d+1)j}.$$

We observe that Proposition 3 can be used to obtain $\mathfrak{C}_d(n' - p, i)$ from the previous term with only integer multiplications and exact divisions; it suffices to first set $p = 1$ in the product above, and then substitute n' by $n' - p$.

Corollary 1.

$$\frac{\mathfrak{C}_d(n' - p, i)_j}{\mathfrak{C}_d(n' - p - 1, i)_j} = \frac{(n' - p) - (d+1)j + i - 1}{(n' - p) - (d+1)j}.$$

Thus, the summation of Lemma 2 must be evaluated only $k - 1$ times, once at the start of every recursion path, rather than at every step of P_{inc} .

We empirically measure the performance of both techniques in Section 4, comparing them against (un)ranking algorithms to other orders.

3.3 The “spiral” order for cryptographic applications

Cryptographic applications often require sampling combinatorial objects uniformly at random, and unranking algorithms are regularly employed for this purpose. A direct consequence of the change of variable lemma [18, p. 190] is that a bijection between two finite sets maps a uniform probability distribution on one set to another such distribution on the other. Thus, providing uniform random integers to unranking functions generates compositions drawn from the same distribution.

Combinatorial counts $\mathfrak{C}_d(n, k)$ used in unranking algorithms exhibit a strong concentration of measure under the uniform random input condition. As established in [8, Section VIII], the coefficients of powers of polynomials (such as the generating function in Lemma 1) converge asymptotically to a Gaussian limit. Particularly, it is shown in [28, Theorem 1.3] that $\mathfrak{C}_d(n, k)$ follows a normal distribution with an approximation error of $\mathcal{O}(n^{-\frac{1}{5}})$.

This behavior implies that for a random rank, the partial sums S_j calculated in the unranking “path” are also asymptotically Gaussian. Empirical data [28, Table 1] suggests the approximation error is small ($< 1\%$) even for modest $n \geq 80, k \geq 30$. As parameters n, k, d suitable for cryptographic applications are usually larger [21, Table 3], or bounded to a security level $\lambda \geq 128$ such that $\mathfrak{C}_d(n, k) \approx 2^\lambda$ (e.g., [14, Table 1] and [7, Table 5]), we argue that the Gaussian behavior can be exploited to accelerate unranking algorithms. Concretely, we propose a path function $\mathbf{P}_s = (\mu, \mu + 1, \mu - 1, \mu + 2, \mu - 2, \dots) \cap [d]$ for some expected value μ . We first properly characterize this quantity.

Statistics of random bounded compositions. Let R be a discrete random variable that represents the total sum of a composition $z \stackrel{r}{\leftarrow} \mathcal{C}_d(n, k)^{(l)}$, where $\mathcal{C}_d(n, k)^{(l)} = \bigcup_{r=n-l}^n \mathcal{C}_d(r, k)$, generalizing the simpler setting of drawing from $\mathcal{C}_d(n, k)$. We recall that the p -th raw moment of R is $\mathbb{E}[R^p] = \frac{\mu_p}{\mu_0}$, with $\mu_p(n, k, d, l) = \sum_{r=n-l}^n r^p \cdot \mathfrak{C}_d(r, k)$. (We write μ_p whenever clear from the context.) For $p = 0$, the r^p term vanishes and $\mu_0 = \mathfrak{C}_d(n, k)^{(l)}$, calculated via Lemma 3. We provide a closed-form generalization of this result which might be of independent interest. First, we prove an auxiliary lemma on weighted sums of binomial coefficients.

Proposition 4. *Let $s, c, y, l \in \mathbb{Z}_{\geq 0}$ and $W(s, c, y, l) = \sum_{r=y-l}^y r^s \binom{r}{c}$. Then,*

$$W(s, c, y, l) = \begin{cases} \binom{y+1}{c+1} - \binom{y-l}{c+1} & \text{if } s = 0, \\ c \cdot W(s-1, c, y, l) + (c+1) \cdot W(s-1, c+1, y, l) & \text{otherwise.} \end{cases}$$

Now, we calculate the exact p -th raw moment.

Theorem 4. *Let $n, k, d \in \mathbb{Z}_{>0}$ as per Lemma 2, $j = \min(k, \lfloor \frac{n}{d+1} \rfloor)$, $l \in [d]$ and set $u_i = (d+1)i - k + 1$ for convenience. Then, for any $p \in \mathbb{Z}_{\geq 0}$,*

$$\mu_p(n, k, d, l) = \sum_{i=0}^j (-1)^i \binom{k}{i} \left[\sum_{s=0}^p \binom{p}{s} u_i^{p-s} W(s, k-1, n-u_i, l) \right].$$

To define the expected mean and variance of R in terms of μ_p , the following corollaries are useful. We let $W_0(k') = W(0, k', n - u_i, l)$ for readability.

Corollary 2.

$$\mu_1(n, k, d, l) = \sum_{i=0}^j (-1)^i \binom{k}{i} \left[(d+1)i \cdot W_0(k-1) + k \cdot W_0(k) \right].$$

Corollary 3.

$$\mu_2(n, k, d, l) = \sum_{i=0}^j (-1)^i \binom{k}{i} (\alpha_i + \beta_i + \gamma_i)$$

for $\alpha_i = ((d+1)i)^2 \cdot W_0(k-1)$, $\beta_i = k(2(d+1)i+1) \cdot W_0(k)$ and $\gamma_i = k(k+1) \cdot W_0(k+1)$.

The first and second raw moments of R are respectively $\mathbb{E}[R] = \frac{\mu_1}{\mu_0}$ and $\mathbb{E}[R^2] = \frac{\mu_2}{\mu_0}$. Consequently, the second central moment is $\mathbb{V}(R) = \mathbb{E}[R^2] - (\mathbb{E}[R])^2$ [18, p. 244]. We are now able to provide formulas for the first raw and second central moments of the partial sums of a random bounded composition from the accumulated set.

Proposition 5. For $k > 1$, let $X_1, \dots, X_k$ be discrete random variables that represent the i -th part of a composition $z \stackrel{\leftarrow}{\leftarrow} \mathcal{C}_d(n, k)^{(l)}$ for $1 \leq i \leq k$, and $S_j = \sum_{i=1}^j X_i$ be the j -th partial sum, with $S_k = R$. Then,

$$\mathbb{E}[S_j] = j \cdot \frac{\mathbb{E}[R]}{k} \quad \text{and} \quad \mathbb{V}(S_j) = \frac{j(k-j)}{k-1} \cdot \mathbb{V}(X_1) + \frac{j(j-1)}{k(k-1)} \mathbb{V}(R)$$

with $\mathbb{V}(X_1) = \frac{1}{\mu_0(n, k, d, l)} \left(\sum_{y=0}^{\min(n, d)} y^2 \mu_0(n-y, k-1, d, l) \right) - \left(\frac{\mathbb{E}[R]}{k} \right)^2$ and the moments of R as given above.

For the simpler case of $l = 0$, we have $\mathbb{V}(R) = 0$ and $\mathbb{E}[S_j] = j \cdot \frac{n}{k}$. During the computation of $\mathbf{P}_s$, the remaining sum n' across i remaining parts shares the identical distribution as S_i due to the exchangeability of parts. Thus, at any recursive step, the expected value of the next part is $\frac{n'}{i}$, the “center of the spiral”. We are also able to define a suitable tabling mechanism. For $\max(0, n - (k-i)d) \leq n' \leq \min(n, i \cdot d)$, $1 \leq i \leq k$, and $c \in \mathbb{Z}_{>0}$, the *probabilistic table*

$$\chi_{\text{comb-}\sigma}^{(d)}(n, k) = \left\{ (n', i) \in \chi_{\text{comb}}^{(d)}(n, k) : |n' - \mathbb{E}[S_i]| \leq c \cdot \sqrt{\mathbb{V}(S_i)} \right\}$$

is a truncation of χ_{comb} which covers the high-probability “band” delimited by Proposition 5. (Without loss of generality, we define $\chi_{\text{acc-}\sigma}^{(d)}(n, k)$ analogously.) Relying on the asymptotic Gaussian convergence of these compositions, selecting $c = 4$ covers roughly 99.993% of the probability mass while storing only a fraction³ of the original matrix. An unranking algorithm employing this table will rarely fall back to direct computation (e.g., via Lemma 2) if the required combinatorial count is not found.

³ The $\sqrt{\mathbb{V}(S_j)}$ term traces an elliptic segment (characteristic of a Brownian bridge) whose integral yields a total area of $\mathcal{O}(k\sqrt{k})$.

4 Experiments

Next, we compare the performance and storage requirements of several algorithms and techniques previously discussed. To reproduce the experiments, we provide source code and accompanying documentation [33]. We choose parameter sets based on the TSL construction in [14, Table 1], which may be employed in SLH-DSA [25]. For the $\lambda = 128$ security level, equivalent to $\mathfrak{C}_d(n, k) \approx 2^\lambda$, we select $k \in \{25, 67, 136\}$ and calculate n, d according to [14, Section 4.2]. We denote by *ps* the “partial sums” technique, and by *bis* the “bisect” technique. Table 1 shows the results of our evaluations.

Experimental setup. All benchmarks are run on an AMD Ryzen™ 7 9700X @ 3.8GHz in a GNU/Linux environment (kernel 6.19.11) and gcc 15.2.1. Executables are compiled with the `-O3 -march=native -mtune=native` set of optimization flags. We employ `_BitInt(N)`, a C23 type for (un)signed integers of bit length N [13, Section 6.2.5]. We set $N = 256$ for the experiments discussed hereafter, and remark that the choice of N directly impacts storage requirements and performance of (un)ranking algorithms. We do not show $\prec_{\text{gray}}$, $\prec_{\text{bous}}$ or lexicographic variants due to their equivalent (or worse) performance compared to other orders. All tables are implemented as *ragged arrays* with appropriate metadata, packing the data structure optimally.

Table 1. Performance and behavior of selected unranking algorithms and tabling mechanisms. Table size is measured in 256-bit integers ($\times 10^3$) and performance is measured in CPU cycles ($\times 10^3$). Results are the average of 2^{10} calls with random seeds.

	Tabling mechanism			Unranking algorithm performance				
	Type	Size	Hits (%)	colex	ps	bis	rb	spiral
$\mathcal{C}_{86}(384, 25)$	—	0.00	0.00	1141.03	178.78	9000.06	1613.85	1422.24
	χ_{bin}	10.34	100.00	74.60	102.53	382.53	221.10	92.33
	$\chi_{\text{comb-2}}$	3.25	97.63	10.79	103.41	383.40	49.92	13.78
	$\chi_{\text{comb-4}}$	5.55	99.99	9.13	103.84	381.19	50.13	12.73
	χ_{comb}	10.01	100.00	9.18	103.56	381.71	52.51	11.96
	$\chi_{\text{acc-2}}$	233.82	89.04	9.09	105.80	65.75	47.45	11.82
	$\chi_{\text{acc-4}}$	417.29	97.67	9.09	103.33	33.68	47.33	11.87
	χ_{acc}	753.48	100.00	9.12	103.82	7.46	47.53	11.98
	—	0.00	0.00	490.13	282.91	3536.53	354.40	564.50
$\mathcal{C}_9(66, 67)$	χ_{bin}	6.83	100.00	37.24	41.93	171.70	77.20	42.58
	$\chi_{\text{comb-2}}$	1.29	97.36	6.12	45.59	171.56	24.34	8.18
	$\chi_{\text{comb-4}}$	2.28	100.00	5.72	42.25	172.74	25.25	7.20
	χ_{comb}	4.56	100.00	5.46	44.78	176.40	23.85	6.89
	$\chi_{\text{acc-2}}$	10.83	77.35	5.37	42.34	47.01	23.82	6.96
	$\chi_{\text{acc-4}}$	19.52	81.13	5.06	41.92	41.33	25.06	6.95
	χ_{acc}	46.36	100.00	5.11	42.34	6.45	24.13	6.90
	—	0.00	0.00	932.53	759.97	5010.33	264.35	929.96
	χ_{bin}	14.80	100.00	52.92	53.76	206.63	68.89	57.28
$\mathcal{C}_4(39, 136)$	$\chi_{\text{comb-2}}$	1.65	96.25	7.59	54.16	206.42	34.37	10.68
	$\chi_{\text{comb-4}}$	2.88	99.96	5.52	54.46	202.73	35.03	8.91
	χ_{comb}	5.48	100.00	5.21	54.32	206.02	32.66	8.68
	$\chi_{\text{acc-2}}$	6.39	58.24	5.20	55.30	97.19	32.32	8.67
	$\chi_{\text{acc-4}}$	11.28	61.33	5.15	54.33	89.26	32.49	9.24
	χ_{acc}	31.28	100.00	5.15	54.73	5.93	32.90	8.64
	—	0.00	0.00	932.53	759.97	5010.33	264.35	929.96
	χ_{bin}	14.80	100.00	52.92	53.76	206.63	68.89	57.28
	$\chi_{\text{comb-2}}$	1.65	96.25	7.59	54.16	206.42	34.37	10.68

Performance discussion. The performance of unranking algorithms is highly sensitive to the chosen parameter sets and the availability of pre-computed tables. In the absence of such mechanisms, $\prec_{rb}$ is the most efficient strategy for larger k , particularly against the canonical $\prec_{colex}$. On the other hand, the ad hoc optimization $\mathbf{ps}$ drastically accelerates $\prec_{colex}$ for small k regimes and monotonic strategies. While $\prec_{spiral}$ offers a theoretically optimal search path, its practical advantages are restricted by the choice of parameters. In isolated synthetic evaluations with $\mathcal{C}_8(200, 50)$ and $\mathcal{C}_{42}(525, 25)$, performance is improved by up to 8% in the absence of tabling mechanisms.

However, parameters derived from known cryptographic algorithms are usually optimized for bandwidth and security bounds rather than statistical density. In these specific configurations, the variance of the partial sums remains sufficiently large such that $\prec_{spiral}$ does not yield a decisive computational advantage over other strategies previously discussed. Nonetheless, we emphasize that $\prec_{spiral}$ can be applied to future cryptographic primitives where larger, statistically dense parameter regimes are required.

Storage discussion. The introduction of χ_{comb} essentially balances performance across the selected algorithms, except for $\mathbf{ps}$, which does not benefit from any pre-computed tables other than χ_{bin} . If ample memory is available, $\mathbf{bis}$ paired with χ_{acc} is usually the fastest solution. Table 1 also shows how many wide integers are required to allocate several types of tabling mechanisms for D_l or a similar “linear” decomposition. Neither χ_{bin} nor χ_{acc} offer effective trade-offs; notably, χ_{acc} consumes 3–42 $\times$ more storage than χ_{comb} for negligible speed gains ($< 3\times$). It also grows quickly even for moderate parameters and can only be used by a particular class of strategies.

Consequently, we identify χ_{comb} as the superior general-purpose choice. For constrained environments, probabilistic alternatives provide excellent value: setting $c = 4$ halves storage with only a 2–6% performance penalty. Finally, D_s remains highly efficient: as $|H(k)| \approx \lg k$ by Proposition 1, it makes $\prec_{rb}$ competitive against $\prec_{colex}$ or $\prec_{spiral}$, even when the recursion tree is unbalanced. For instance, the χ_{comb} table size is reduced from 5480 to only 400 integers if $\prec_{rb}$ is employed for $\mathcal{C}_4(39, 136)$.

Constant-time implementations. We observe that while the generic framework does not eliminate side-channel vulnerabilities, it isolates the sources of information leakage. The side-channel resistance of any instantiated algorithm reduces to the choice of strategy $\Sigma = (D, P, S)$. Specifically, D operates only on public parameters, and S performs basic arithmetic on the rank r , which can be secured using standard constant-time arithmetic primitives. The attack surface is thus restricted to P ; since it dictates the recursion path, it introduces data-dependent branching (loop terminations derived from z or r) and memory accesses (table lookups for $\mathcal{C}_d(n, k)$). Consequently, producing a data-oblivious (un)ranking algorithm does not require restructuring the combinatorial logic, and reduces to the design of a constant-time evaluator for P .

5 Conclusion

We argue that our approach allows us to unify distinct algorithms found in the literature under a single algebraic definition, and study orders from a structural viewpoint. Furthermore, the choice of strategies is more suitable to finding new perspectives on optimizing (un)ranking algorithms for specialized applications. We provide an example of this methodology by proposing the “spiral” order, based on the uniform random input condition imposed by cryptographic algorithms. We hope this framework sheds light on the deep connections of the combinatorial theory underpinning the discussed cryptographic applications.

Future work. We suggest the characterization of other ad hoc optimizations, such as pre-computing “pivots” to optimize the unranking algorithm, with respect to our strategy framework. Additionally, the statistics can be extended to analyze the storage requirements of the binomial table χ_{bin} under the uniform random input condition; preliminary evidence shows that similar high-probability patterns arise when Lemma 2 is used to evaluate $\mathfrak{C}_d(n, k)$. The formal verification of our generic implementations and the exploration of constant-time (un)ranking algorithms for cryptography are also highly interesting avenues of research.

Acknowledgments. We are grateful to the anonymous reviewers for their helpful comments and suggestions. G. Zambonin was partially supported by the Fundacao de Amparo a Pesquisa e Inovacao do Estado de Santa Catarina (FAPESC), Edital 62/2024, and by the National Operator of the Civil Registry of Natural Persons (ON-RCPN) of Brazil. L. G. Rosa was partially supported by the Fundacao de Amparo a Pesquisa e Inovacao do Estado de Santa Catarina (FAPESC), Edital 61/2024. D. Panario was partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC), reference number RGPIN-2024-05341.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. André, D.: Mémoire sur les combinaisons régulières et leurs applications. *Annales scientifiques de l’École Normale Supérieure* **5**, 155–198 (1876). <https://doi.org/10.24033/asens.136>, 2e série
2. Arndt, J.: *Matters Computational: Ideas, Algorithms, Source Code*. Springer Berlin, Heidelberg, 1st edn. (2011)
3. Barenghi, A., Pelosi, G.: Constant weight strings in constant time: a building block for code- based post-quantum cryptosystems. In: *CF ’20: Proceedings of the 17th ACM International Conference on Computing Frontiers*. pp. 132–141 (May 2020). <https://doi.org/10.1145/3387902.3392630>
4. Bellare, M., Ristenpart, T., Rogaway, P., Stegers, T.: Format-Preserving Encryption. In: Jacobson, M.J., Rijmen, V., Safavi-Naini, R. (eds.) *Selected Areas in Cryptography. Lecture Notes in Computer Science*, vol. 5867, pp. 295–312 (Aug 2009). https://doi.org/10.1007/978-3-642-05445-7_19

5. Eger, S.: Stirling's Approximation for Central Extended Binomial Coefficients. *Journal of Integer Sequences* **16**(13.1.3) (2013)
6. Fahssi, N.E.: Polynomial Triangles Revisited. arXiv:1202.0228v7 (Jul 2012). <https://doi.org/10.48550/arXiv.1202.0228>
7. Filho, D.L.G., Adj, G., Bettaieb, S., Budroni, A., Chávez-Saab, J., Rodríguez-Henríquez, F.: Sota Voce: Low-Noise Sampling of Sparse Fixed-Weight Vectors. *IACR Transactions on Cryptographic Hardware and Embedded Systems* **2026**(1), 353–375 (Jan 2026). <https://doi.org/10.46586/tches.v2026.i1.353-375>
8. Flajolet, P., Sedgewick, R.: *Analytic Combinatorics*. Cambridge University Press, 1st edn. (2009)
9. Flajolet, P., Zimmermann, P., Cutsem, B.V.: A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science* **132**(1), 1–35 (Sep 1994). [https://doi.org/10.1016/0304-3975\(94\)90226-7](https://doi.org/10.1016/0304-3975(94)90226-7)
10. Genitrini, A., Pépin, M.: Lexicographic Unranking of Combinations Revisited. *Algorithms* **14**(3) (Mar 2021). <https://doi.org/10.3390/a14030097>, 97
11. Honda, M., Kaji, Y.: Optimum Fingerprinting Function for Winternitz One-Time Signature. In: 2024 IEEE International Symposium on Information Theory (ISIT). pp. 2898–2902 (Jul 2024). <https://doi.org/10.1109/ISIT57864.2024.10619351>
12. Hülsing, A., Kudinov, M., Ronen, E., Yogev, E.: SPHINCS+C: Compressing SPHINCS+ With (Almost) No Cost. In: 2023 IEEE Symposium on Security and Privacy (S&P). pp. 1435–1453 (May 2023). <https://doi.org/10.1109/SP46215.2023.10179381>
13. ISO/IEC JTC 1/SC 22: Information technology — Programming languages — C. Standard 9899:2024, International Organization for Standardization (Oct 2024)
14. Khovratovich, D., Kudinov, M., Wagner, B.: At the Top of the Hypercube – Better Size-Time Tradeoffs for Hash- Based Signatures. In: Kalai, Y.T., Kamara, S.F. (eds.) *Advances in Cryptology – CRYPTO 2025*. Lecture Notes in Computer Science, vol. 16005, pp. 93–123 (Jul 2025). https://doi.org/10.1007/978-3-032-01887-8_4
15. Knuth, D.E.: *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1*. Addison-Wesley Professional, 1st edn. (2011)
16. Krausz, M., Land, G., Richter-Brockmann, J., Güneysu, T.: A Holistic Approach Towards Side-Channel Secure Fixed-Weight Polynomial Sampling. In: Boldyreva, A., Kolesnikov, V. (eds.) *Public-Key Cryptography – PKC 2023*. Lecture Notes in Computer Science, vol. 13941, pp. 94–124 (May 2023). https://doi.org/10.1007/978-3-031-31371-4_4
17. Kreher, D.L., Stinson, D.R.: *Combinatorial Algorithms: Generation, Enumeration, and Search*. CRC Press, 1st edn. (1998)
18. Loève, M.: *Probability Theory I*, Graduate Texts in Mathematics, vol. 45. Springer (1977). <https://doi.org/10.1007/978-1-4684-9464-8>
19. López-García, M., Farouk-Marei, D.G., Cantó-Navarro, E.: Converting Fixed-Length Binary Strings Into Constant Weight Words: Application on Post-Quantum Cryptography. *IEEE Transactions on Dependable and Secure Computing* **22**(3), 3063–3074 (Dec 2024). <https://doi.org/10.1109/TDSC.2024.3524626>
20. Martínez, C., Molinero, X.: A Generic Approach for the Unranking of Labeled Combinatorial Classes. *Random Structures & Algorithms* **19**(3-4), 472–497 (Oct 2001). <https://doi.org/10.1002/rsa.10025>
21. Miracle, S., Yilek, S.: New Algorithms and Analyses for Sum-Preserving Encryption. In: Agrawal, S., Lin, D. (eds.) *Advances in Cryptology – ASIACRYPT 2022*. Lecture Notes in Computer Science, vol. 13793, pp. 3–31 (Dec 2022). https://doi.org/10.1007/978-3-031-22969-5_1

22. de Moivre, A.: *Miscellanea analytica de seriebus et quadraturis*. J. Tonson & J. Watts, 1st edn. (1730)
23. Molinero, X.: Ordered generation of classes of combinatorial structures. PhD thesis, Universitat Politècnica de Catalunya (Jul 2005)
24. de Moura, L., Ullrich, S.: The Lean 4 Theorem Prover and Programming Language. In: Platzer, A., Sutcliffe, G. (eds.) *Automated Deduction - CADE 28* (CADE 2021). *Lecture Notes in Computer Science*, vol. 12699, pp. 625–635 (Jul 2021). https://doi.org/10.1007/978-3-030-79876-5_37
25. NIST: Stateless Hash-Based Digital Signature Standard. Federal Information Processing Standards Publication 205, National Institute of Standards and Technology (Aug 2024). <https://doi.org/10.6028/nist.fips.205>
26. Perin, L.P., Zambonin, G., Custódio, R., Moura, L., Panario, D.: Improved constant-sum encodings for hash-based signatures. *Journal of Cryptographic Engineering* **11**(4), 329–351 (Jun 2021). <https://doi.org/10.1007/s13389-021-00264-9>
27. Qin, Z., Sun, S.: Extending the SPHINCS+ Framework: Varying the Tree Heights and Chain Lengths. *Cryptology ePrint Archive*, Paper 2025/2236 (Dec 2025), <https://eprint.iacr.org/2025/2236>
28. Ratsaby, J.: Estimate of the number of restricted integer-partitions. *Applicable Analysis and Discrete Mathematics* **2**(2), 222–233 (2008). <https://doi.org/10.2298/AADM0802222R>
29. Reyzin, L., Reyzin, N.: Better than BiBa: Short One-Time Signatures with Fast Signing and Verifying. In: Batten, L., Seberry, J. (eds.) *Information Security and Privacy*. *Lecture Notes in Computer Science*, vol. 2384, pp. 144–153 (Jul 2002). https://doi.org/10.1007/3-540-45450-0_11
30. Stojmenović, I.: Generating All and Random Instances of a Combinatorial Object. In: *Handbook of Applied Algorithms: Solving Scientific, Engineering and Practical Problems*, pp. 1–38. John Wiley & Sons, 1st edn. (2007). <https://doi.org/10.1002/9780470175668.ch1>
31. Tajik, K., Gunasekaran, A., Dutta, R., Ellis, B., Bobba, R.B., Rosulek, M., Wright, C.V., Feng, W.: Balancing Image Privacy and Usability with Thumbnail-Preserving Encryption. In: *Network and Distributed Systems Security (NDSS) Symposium 2019* (Feb 2019). <https://doi.org/10.14722/ndss.2019.23432>
32. Vajnovszki, V., Vernay, R.: Restricted compositions and permutations: From old to new Gray codes. *Information Processing Letters* **111**(13), 650–655 (Jul 2011). <https://doi.org/10.1016/j.ipl.2011.03.022>
33. Zambonin, G.: *bic*. Version 0.2.0 (Apr 2026). <https://doi.org/10.5281/zenodo.19476406>
34. Zambonin, G.: *bic-proofs*. Version 0.2.0 (Apr 2026). <https://doi.org/10.5281/zenodo.19476432>
35. Zhang, K., Cui, H., Yu, Y.: Revisiting the Constant-Sum Winternitz One-Time Signature with Applications to SPHINCS⁺ and XMSS. In: Handschuh, H., Lysyanskaya, A. (eds.) *Advances in Cryptology – CRYPTO 2023*. *Lecture Notes in Computer Science*, vol. 14085, pp. 455–483 (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_15
36. Zhang, Y., Zhou, W., Zhao, R., Zhang, X., Cao, X.: F-TPE: Flexible Thumbnail-Preserving Encryption Based on Multi-Pixel Sum-Preserving Encryption. *IEEE Transactions on Multimedia* **25**, 5877–5891 (Aug 2022). <https://doi.org/10.1109/tmm.2022.3200310>