POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

RandShaper: A Library for the Secure Shaping of Pseudo-Random Objects for Post-Quantum Cryptography

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Marco Pozzi

Advisor: Prof. Alessandro Barenghi

Co-advisor: Marco Gianvecchio

Academic year: 2025-2026

1. Introduction

Cryptography is the process that makes data protection and secure communication possible. The security of modern cryptographic schemes is threatened by the rise of quantum computers, which process information using qubits. Particularly, two quantum algorithms pose a significant threat to contemporary cryptographic systems: Shor's algorithm, which efficiently factors large numbers and computes discrete logarithms, and Grover's algorithm, which accelerates brute-force search processes, undermining the security of symmetric-key encryption by effectively halving the key strength.

For this reason, the National Institute of Standards and Technology (NIST) is currently working on meeting modern cryptographic requirements, including standards and technologies that should be implemented to address the quantum threat through Post-Quantum Cryptography (PQC).

Through a multi-year international competition involving industry, academia, and governments, NIST released three PQC Federal Information Processing Standards (FIPS) in 2024: the Key-Encapsulation Mechanism Standard FIPS 203 (ML-KEM) [5], and the two Digital Signature Standards FIPS 204 (ML-DSA) [6] and FIPS 205 (SLH-DSA) [7].

In September 2022, NIST called for additional digital signature proposals to be considered in the PQC standardization process to diversify its post-quantum signature portfolio. Through a multi-round competition, 11 schemes have reached the third round. In the near future, NIST will narrow down the number of candidates until the final algorithms to be standardized are selected.

Across all these PQC schemes, the Key Generation, Signature, and Verification processes, as well as other auxiliary functions, strictly require the generation of pseudo-random objects, such as binary strings, modular numbers, fixed-weight vectors, and permutation arrays.

2. The RandShaper Framework

This thesis presents the RandShaper library, a framework designed to support post-quantum cryptographic schemes in the secure generation and shaping of pseudo-random objects. The generation of such objects typically involves a trade-off between execution efficiency, optimal entropy consumption, and security. In particular, this thesis focuses on the constant-

time property, a fundamental defense mechanism against side-channel timing attacks. The golden rule of constant-time programming dictates that secret information must never influence resource usage or execution time, effectively preventing attackers from extracting sensitive data through timing measurements.
To address these challenges, the RandShaper library collects and efficiently implements existing methods, while also proposing new algorithms and variants to optimally generate the target pseudo-random objects.

2.1. Software Architecture

The RandShaper library is designed with a three-tiered architecture to guarantee flexibility and ease of integration. Figure 1 illustrates the layers of the framework, which are structured as follows:

- **Cryptographic Backend Layer**: It contains the raw implementations of the cryptographic primitives responsible for the generation of pseudo-random numbers: the Advanced Encryption Standard (AES) [3], SHAKE [4], and ChaCha [8].
- **PRNG Abstraction Layer**: This intermediate layer is responsible for the internal buffering and the management of the pseudo-random bits obtained by invoking the primitives from the layer below.
- **Object Shapers Layer**: Starting from the pseudo-random bits provided by the PRNG Abstraction Layer, this highest level is responsible for the secure generation of more complex pseudo-random objects (i.e., modular arrays, fixed-weight vectors, and permutations).

2.2. Generation of pseudo-random objects

As stated in the architectural description of the RandShaper framework, existing cryptographic primitives—namely AES, SHAKE, and ChaCha—are employed to generate plain pseudo-random binary strings.
For the generation of pseudo-random arrays of numbers modulo n, where $n \in \mathbb{N}$, four algorithms are implemented. The Rejection Sampling method is a non-constant-time approach that generates a pseudo-random number x and rejects it if $x \geq n$. The other three are constant-time methods that exploit the decoding phase of arithmetic coding [10]. Among these, the Monolithic Arithmetic Decoding method achieves optimal entropy usage but is computationally inefficient. Conversely, the Chunked Arithmetic Decoding and the Chunked Tree Arithmetic Decoding are significantly more efficient approaches, at the cost of slightly higher entropy waste.
For the generation of fixed-weight binary vectors—defined as binary vectors with a fixed number of non-zero elements—five existing approaches have been efficiently implemented:

- **RepeatedAND method** [2]: A non-constant-time approach that takes advantage of the statistical properties of the bit-wise AND operation applied to uniformly distributed random binary vectors to obtain the target vector.
- **Sorting method**: A strictly constant-time approach that employs an efficient integer sorting algorithm to uniformly shuffle the positions of ones and zeros within a binary vector.
- **Rejection method**: A non-constant-time approach that generates pseudo-random integers acting as positional indices to manipulate the output binary vector.
- **Fisher-Yates method**: A non-constant-time approach that uniformly randomizes

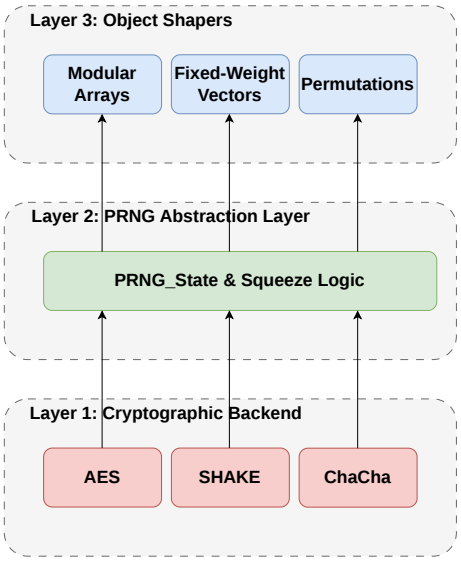

Figure 1: High-level software architecture of the RandShaper framework.

a pre-initialized vector by swapping its elements with others chosen at random.
- **Sendrier's Fisher-Yates method** [9]: A variant of the classical Fisher-Yates shuffle designed to offer secure, constant-time execution.

Finally, for the generation of permutations, the RandShaper library implements the method proposed by Daniel J. Bernstein [1] to generate control bits and route elements through an in-place Beneš network, a routing topology capable of applying any arbitrary permutation to an array of $n = 2^m$ elements, where $m \in \mathbb{N}$.

3. Benchmarks

The tested parameters are those of the NIST Round 2 candidates for additional digital signatures. For modular arrays, fixed-weight binary vectors, and permutations, the employed PRNG is a parallelized version of SHAKE-256.

Regarding binary string generation (Figure 2), the ChaCha20 implementation achieves the highest throughput among the evaluated PRNGs for cryptographically relevant data sizes. It is followed in efficiency by the hardware-accelerated AES-256-NI-CTR-DRBG, the vectorized SHAKE-256-X4, and the standard SHAKE-256. Conversely, the purely software-based AES-256-CTR-DRBG exhibits the poorest performance, ranking last across the benchmark.

For modular array generation (Figure 3), the Rejection Sampling method is the fastest across all tests. This performance is due to its non-constant-time nature, as the number generation and rejection processes are computationally inexpensive. Among the constant-time Arithmetic Decoding methods, the Chunked and Chunked Tree approaches are the fastest. The Monolithic approach is consistently the slowest; while it achieves optimal entropy usage, it requires much more expensive multi-precision arithmetic operations.

In fixed-weight binary vector generation (Figure 4), a clear performance pattern emerges based on vector density. Among the non-constant-time approaches, the RepeatedAND and Rejection methods consistently outperform the others, whereas the classical Fisher-Yates method is strictly dominated in all evaluated scenarios. Specifically, the RepeatedAND method provides the highest throughput for dense vectors (i.e., those with a low number of zero elements), while the Rejection approach proves to be the most efficient for sparse vectors. A symmetric behavior is observed within the constant-time approaches: the Sorting method is highly optimized and dominates the generation of dense vectors, whereas Sendrier's Fisher-Yates variant scales much better and represents the fastest constant-time solution for sparse vectors.

Finally, for control bit generation in Beneš networks (Figure 5), performance strictly scales with the size of the network. Increasing the network size to the next power of 2 results in a doubling or tripling of the computational overhead. The same scaling applies to the routing process, although the actual data routing remains exponentially faster than the initial control bit generation.

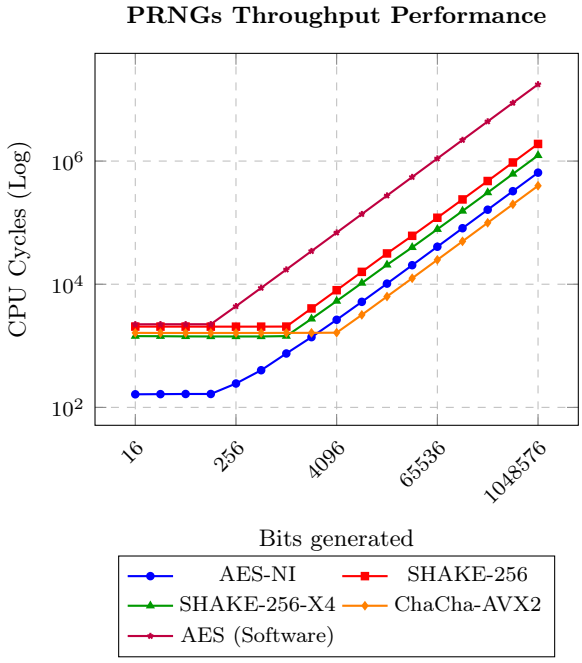

Figure 2: Performance comparison of binary string generation (CPU cycles).

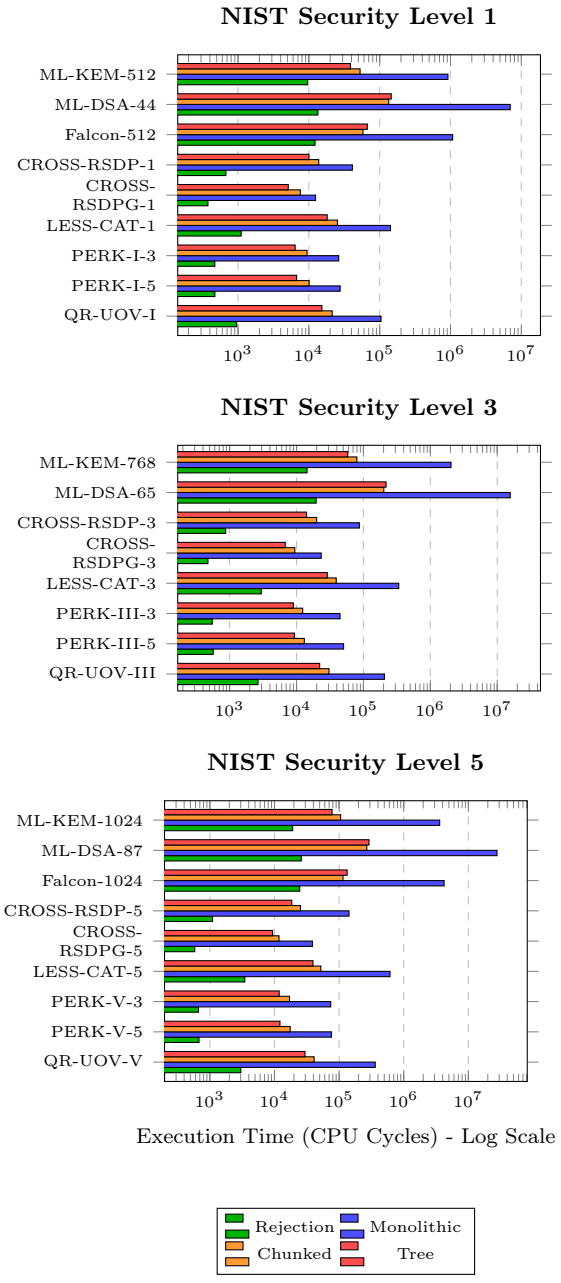

Figure 3: Performance comparison of modular array generation across all three NIST Security Levels.

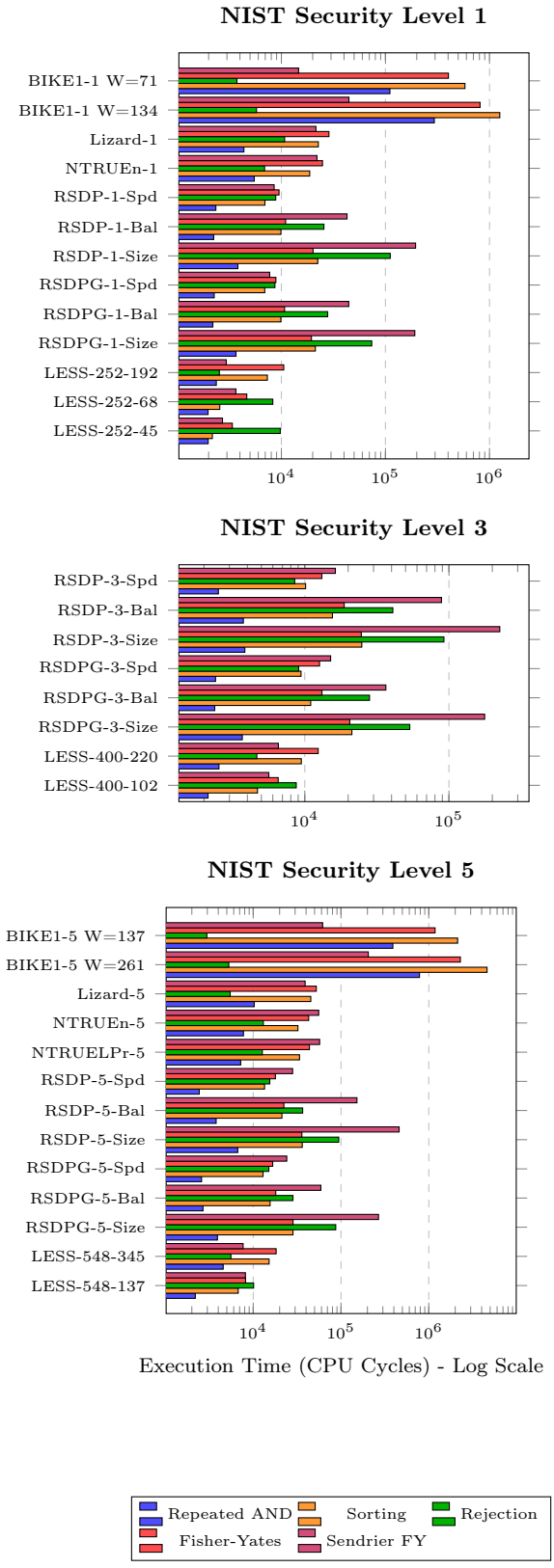

Figure 4: Performance comparison of fixed-weight vector generation across all three NIST Security Levels.

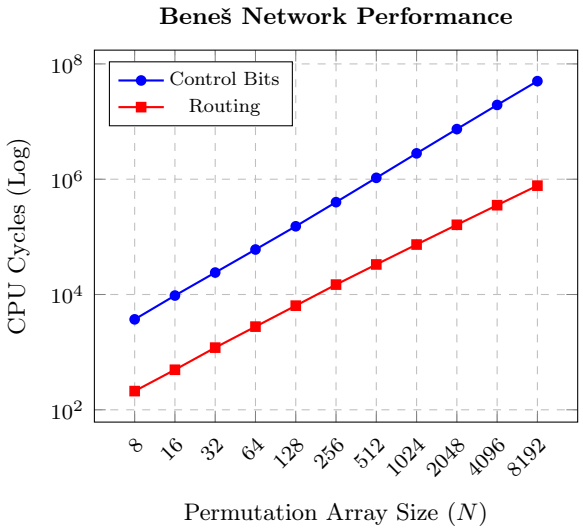

Figure 5: Performance comparison of control bits generation and routing for Beneš networks.

4. Conclusions

This thesis presented the RandShaper framework, a library designed to support post-quantum cryptographic schemes in the generation and shaping of pseudo-random objects: binary strings, fixed-weight vectors, modular numbers, and permutations.
Observing the obtained results, we can conclude that there is no universally dominant method for the generation of a pseudo-random object; some methods excel for certain sets of parameters, while others are more computationally efficient for different ones. Furthermore, the choice of the algorithm to be employed in a post-quantum cryptographic scheme does not depend solely on the efficiency of the method, but also on the specific objectives set by the scheme, such as the requirement of minimal entropy waste or the necessity of the constant-time property.

References

[1] Daniel J. Bernstein. Verified fast formulas for control bits for permutation networks. Document ID: 782136154c6ea5bb93257e2a5f667a579df7a48e, 2020.

[2] Nir Drucker and Shay Gueron. Generating a random string with a fixed weight. In *Cyber Security Cryptography and Machine Learning: Third International Symposium, CSCML 2019, Beer-Sheva, Israel, June 27–28, 2019, Proceedings 3*, pages 141–155. Springer, 2019.

[3] National Institute of Standards and Technology. FIPS 197: Advanced Encryption Standard (AES) . Technical report, U.S. Department of Commerce, November 2001.

[4] National Institute of Standards and Technology. FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Technical report, U.S. Department of Commerce, August 2015.

[5] National Institute of Standards and Technology. FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard. Technical report, U.S. Department of Commerce, August 2024.

[6] National Institute of Standards and Technology. FIPS 204: Module-Lattice-Based Digital Signature Standard. Technical report, U.S. Department of Commerce, August 2024.

[7] National Institute of Standards and Technology. FIPS 205: Stateless Hash-Based Digital Signature Standard. Technical report, U.S. Department of Commerce, August 2024.

[8] Yoav Nir and Adam Langley. ChaCha20 and Poly1305 for IETF Protocols. Technical report, June 2018.

[9] Nicolas Sendrier. Secure sampling of constant-weight words - application to BIKE. HAL archive hal-03534005, 2022.

[10] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.