POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

C++ redesign of attack algorithms to code-based cryptography

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Alessandro Conti

Advisor: Prof. Alessandro Barenghi

Co-advisors: Prof. Gerardo Pelosi, Dr. Simone Perriello

Academic year: 2025-2026

1. Introduction

The rapid advances in quantum computing pose a concrete and imminent threat to widely deployed public-key cryptographic systems. Algorithms such as Rivest, Shamir, Adleman (RSA) and Elliptic Curve Cryptography (ECC) base their security on the computational intractability of integer factorization and discrete logarithms, problems that Shor's quantum algorithm [19] can solve in polynomial time. A further risk is the *harvest now, decrypt later* paradigm, in which adversaries store today's ciphertext to decrypt it once quantum hardware becomes available, making the transition to quantum-resistant solutions urgent.

Post-Quantum Cryptography (PQC) addresses this challenge by designing cryptographic primitives that are hard for both classical and quantum adversaries. Among the various families of post-quantum candidates, *code-based cryptography* is one of the most mature. The security of these schemes, exemplified by the McEliece [13] and Niederreiter [14] cryptosystems, rests on the NP-hardness of the General Decoding Problem (GDP) and the Syndrome Decoding Problem (SDP), problems for which no efficient quantum algorithm is known.

Assessing the practical security of code-based schemes requires evaluating the best known attack algorithms, which belong to the Information Set Decoding (ISD) family, first introduced by Prange [15] and subsequently refined over decades. Existing security analyses consider only serial adversaries; no systematic study has evaluated the impact of hardware parallelism on ISD algorithms. This thesis fills that gap.

2. Background

2.1. Code-Based Cryptography

A linear $\mathcal{C}[n, k, d]$ code over a finite field $\mathbb{F}_2$ is defined by a generator matrix $G \in \mathbb{F}_q^{k \times n}$ or, equivalently, by a parity-check matrix $H \in \mathbb{F}_q^{(n-k) \times n}$ satisfying $GH^\intercal = 0$.

The McEliece cryptosystem [13] encrypts a message $\mathbf{m} \in \mathbb{F}_2^k$ as

$$\mathbf{y} = \mathbf{m}SGP + \mathbf{e} \tag{1}$$

where G is the generator matrix of a binary Goppa code, S is a random invertible dense scrambling matrix and P is a random permutation matrix, and $\mathbf{e}$ is an intentional error vector of Hamming weight w. The security of the cryptosystem relies on the hardness of decoding a random linear code. The objective of the GDP is to find a message vector $\mathbf{m} \in \mathbb{F}_2^k$ and an error

vector $\mathbf{e} \in \mathbb{F}_2^n$ such that

$$\begin{cases} \mathbf{y} = \mathbf{m}\widehat{G} + \mathbf{e} \\ \mathrm{HW}(\mathbf{e}) \leq w \end{cases} \tag{2}$$

where $\widehat{G} = SGP$ is known.
The Niederreiter cryptosystem [14] encrypts a message $\mathbf{m} \in \mathbb{F}_2^k$ as

$$\mathbf{s}^\intercal = SHP\mathbf{e}^\intercal \tag{3}$$

where H is the parity-check matrix, S is a random invertible dense scrambling matrix and P is a random permutation matrix, and $\mathbf{e}$ is an intentional error vector of Hamming weight w. Security relies on the hardness of decoding a random linear code. The objective of the Syndrome Decoding Problem (SDP) (that is equivalent to the GDP) is to find an error vector $\mathbf{e} \in \mathbb{F}_2^n$ such that

$$\begin{cases} \mathbf{s}^\intercal = \widehat{H}\mathbf{e}^\intercal \\ \mathrm{HW}(\mathbf{e}) \leq w \end{cases} \tag{4}$$

where $\widehat{H} = SHP$ is known.

2.2. Information Set Decoding

Code-based cryptosystems derive their security from the computational difficulty of solving the GDP or the SDP; as the two problems are equivalent, we will focus on analyzing the SDP without loss of generality.
The most significant family of algorithms for attacking these two problems to date is ISD. The aim of ISD algorithms, in solving the SDP, is to find an error $\mathbf{e} \in \mathbb{F}_2^n$ given a parity-check matrix $H \in \mathbb{F}_2^{(n-k)\times n}$ and a syndrome $\mathbf{s} \in \mathbb{F}_2^{n-k}$ more efficiently than by performing an exhaustive search on the positions of the asserted coefficients of the error
The basic idea behind every ISD algorithm is to randomly select an Information Set (IS), $\mathbf{I}$, and assume that the error vector $\mathbf{e}$ follows a specific distribution associated with that set. The error vector $\mathbf{e}$ is ideally split into two disjoint subvectors, $\mathbf{e}_\mathbf{s} \in \mathbb{F}_2^k$ and $\mathbf{e}_{\mathbf{s}^*} \in \mathbb{F}_2^{n-k}$, corresponding to the positions indexed by the IS and the Redundancy Set (RS) respectively.
Each iteration consists of three phases: a *column permutation phase* in which linear transformations are applied to transform the system of equations represented by the parity-check matrix H into a simpler one, $\widehat{H}$; a *search phase* that enumerates candidate error $\widehat{\mathbf{e}}$ of the transformed system of equation and identifies collisions; and the last phase is the *post processing phase* that reconstructs the original error vector $\mathbf{e}$ starting from the $\widehat{\mathbf{e}}$.

2.3. The CryptAttackTester Framework

CryptAttackTester (CAT) [4] provides a formalized environment for quantitative analysis of cryptographic attacks using a Boolean circuit computational model. Attacks are represented as circuits whose cost is measured in terms of circuit depth (the critical path) and gate count. ISD algorithms can exploit a meet-in-the-middle search strategy, which relies on the existence of collisions in a pseudorandom function. Rather than exhaustively enumerating all candidate error vectors, this approach splits the search space into two halves that are explored independently, identifying matches where the partial solutions collide.
The three algorithm families implemented in the CAT differ in the number of meet-in-the-middle levels employed during the collision search phase:

- `isd0`: the variant with zero collision levels (Prange [15], Lee-Brickell [9], Leon [10], Ball-Collision [5]);
- `isd1`: the variant with one collision levels (Stern [20], Dumer [6]);
- `isd2`: the variant with two collision levels (May-Meurer-Thomae (MMT) [12], Becker-Joux-May-Meurer (BJMM) [2]).

2.4. The Parallel Brute-Force Model

The parallel model adopted in this thesis follows Bernstein's parallel brute-force idea of [3]. Compared to other parallel models, this model is considered more realistic due to the sparse connectivity between the concurrently operating machines. Rather than performing the entire computation on a single computing machine operating sequentially, the work will be distributed across `PF` separate machines operating concurrently. The concurrent machines are arranged on a two-dimensional mesh of size $\sqrt{\texttt{PF}} \times \sqrt{\texttt{PF}}$, where each machine is connected to four neighboring machines (to the north, south, west and east) within the mesh.
In particular, the collision search phase will be

accelerated by combining the parallel Bernstein mesh model from [3] with a two-dimensional sorting network. Several sorting network families were evaluated for this purpose; the LS3 sorting network [8] was selected as it is designed natively for two-dimensional grids of concurrent machines and achieves the best trade-off between depth and implementation complexity among the candidates considered.
The cost metric for a parallel circuit is the area-time product at, where a is the total Boolean gate count and t is the circuit depth.

3. `isd1_parallel`

The main contribution of this thesis is the design and implementation of `isd1_parallel`, a parallel variant of the `isd1` algorithm, integrated into the CAT framework [4]. The algorithm distributes the search phase across `PF` concurrent machines while keeping the column permutation and post processing phases sequential (as they are inherently non-parallelizable).

3.1. Parallelization Strategy

The parallelization focuses on the computationally dominant search phase. Given a candidate list of size ls, each machine independently enumerates a disjoint subset of $\frac{ls}{\mathtt{PF}}$ candidates. The resulting partial lists are then merged and globally sorted via a parallel sorting network; finally, each machine checks for any collisions among the globally sorted candidates.
Several two-dimensional sorting network families were evaluated for the global sorting phase, where n denotes the side length of the two-dimensional array of size $n \times n$ to be sorted:

- 2D Odd-Even Transposition Sort [7]: $\mathcal{O}(n^2)$ depth;
- Shearsort [16]: $n \log_2 n + 3n - 2$ depth;
- Rotatesort [11]: $10n + 5\sqrt{n} + 11$ depth;
- LS3-Sort [8]: $9n - 9$ depth;
- 4-Way Mergesort [17]: $7n - 6$ depth;
- 3n-sort of Schnorr and Shamir [18]: $3n + 22n^{\frac{3}{4}} - 18$ depth.

The choice of sorting network directly influences the depth overhead of the parallel circuit and hence the achievable speedup.

3.2. Implementation

The parallel algorithm, `isd1_parallel`, was implemented in C++ within the CAT codebase [4]. CAT models each attack as a set of functions that operate on Boolean circuits; accordingly, the following C++ functions were implemented for `isd1_parallel`:

- *circuit simulation*: executes the full Boolean circuit of the parallel attack on a given problem instance, producing the exact gate-level result;
- *gate count estimator*: computes the total number of logic gates analytically, without running the circuit, enabling fast parameter sweeps;
- *depth estimator*: analogously predicts the circuit depth (critical path) a priori. This function was missing in CAT, only a gate-count estimator existed, and was therefore introduced as a new contribution;
- *success probability estimator*: computes the expected probability that an attack finds the target error vector, given the attack parameters;
- *parameter enumerator*: returns all valid attack-parameter configurations for a given problem instance, used to identify the optimal configuration for each experimental run.

The correctness of the analytical predictions (`predicteddepth`) was verified against exact circuit simulation (`circuitdepth`) on reduced problem instances before scaling to full cryptographic parameters.
To support the experimental campaign, a Python orchestrator was developed to automate the concurrent execution of multiple parameter configurations and collect the resulting data. Finally, a Python notebook was produced for the statistical analysis and visualization of the collected experimental results.

4. Experimental Evaluation

4.1. Methodology

The experimental campaign evaluated `isd1_parallel` against its serial counterpart `isd1` on the candidate schemes in the fourth round of the National Institute of Standards and Technology (NIST) standardization process for PQC [1].
Three metrics were used:

- **Depth speedup**: ratio of serial depth to parallel depth as a function of `PF`;

Table 1: Representative sets of problem parameters used in the experimental campaign, along with the corresponding NIST security level and security margin (only one case per security level is shown)

n	k	w	Security level	Margin
3488	2720	64	L1	2^{143}
4608	3360	96	L3	2^{207}
6688	5024	128	L5	2^{272}

- **Area-time product**: overall hardware cost of the parallel circuit;
- **Security margin**: minimum circuit depth expressed as a power of 2, compared to the target value.

4.2. Depth Speedup

For all tested parameter sets and attack configurations, increasing `PF` consistently reduces circuit depth compared to the serial `isd1`. The speedup follows a characteristic pattern: rapid initial growth followed by saturation, in accordance with *Amdahl's law*. The non-parallelizable phases (column permutation and post processing) constitute an asymptotic lower bound on circuit depth independent of `PF`.
The maximum speedup observed is approximately 2.2 for the smallest parameter set ($n = 3488, k = 2720, w = 64$) Figure 1. As problem size increases, the speedup asymptotically tends toward unity, because the polynomial-complexity column permutation phase (which scales as $\mathcal{O}\left((n-k)^3\right)$) becomes dominant over the parallelizable search phase.

4.3. Area-Time Product

The area-time product analysis reveals a fundamental trade-off: parallelization reduces circuit depth (time) at the cost of additional gates introduced by the sorting network. As `PF` grows, the area-time product *at* increases monotonically, since the gate overhead of the sorting network outweighs the depth savings. The benefit of parallelism therefore manifests as a depth reduction rather than an overall cost reduction: for configurations where meeting a tight depth budget is the primary constraint, increasing `PF` trades hardware area for execution time. Beyond an optimal `PF` (specific to each configuration), however, the sorting network overhead grows disproportionately, and parallelism reduces execution time only at the expense of greater total hardware resources.

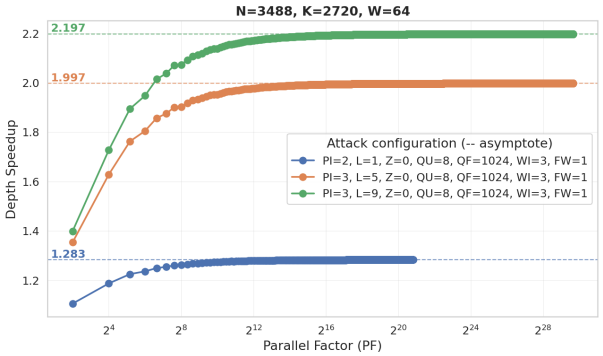

Figure 1: Depth speedup as a function of `PF`, for ($n = 3488, k = 2720, w = 64$)

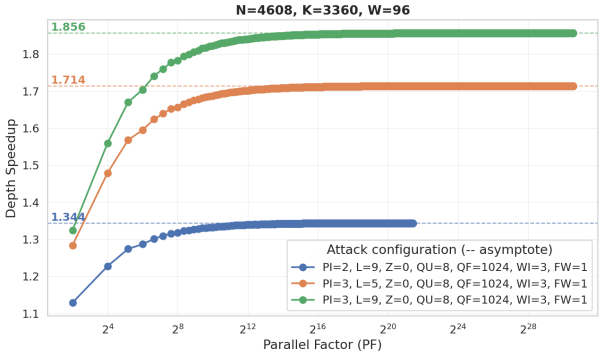

Figure 2: Depth speedup as a function of `PF`, for ($n = 4608, k = 3360, w = 96$)

4.4. Security Margin

Figure 7 summarizes the minimum achievable execution time for each NIST security level under the parallel adversary model, expressed in millennia and computed assuming gate delays of 2^{-12}s. In all cases and for all cryptosystem instances, the minimum execution time amounts to an astronomically large number of millennia, far beyond any conceivable attack horizon, confirming the practical infeasibility of the attack even under full parallelization.
This is further corroborated by the circuit depth analysis: the observed security margins in depth exceed the respective NIST thresholds 2^{λ} by a substantial factor across all tested parameter

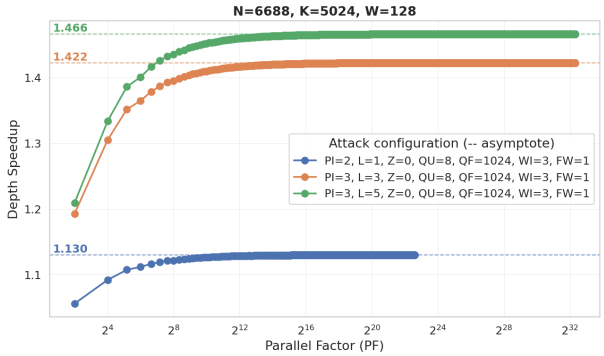

Figure 3: Depth speedup as a function of `PF`, for $(n = 6688, k = 5024, w = 128)$

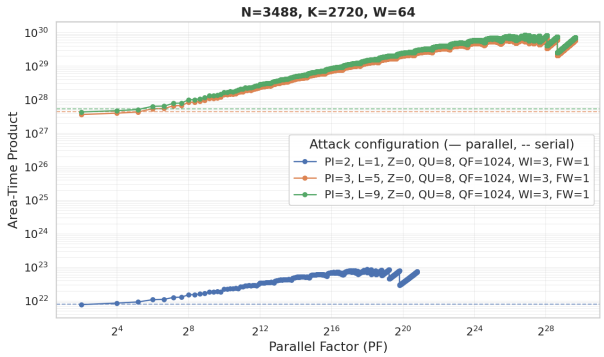

Figure 4: Area-time product as a function of `PF`, for $(n = 3488, k = 2720, w = 64)$

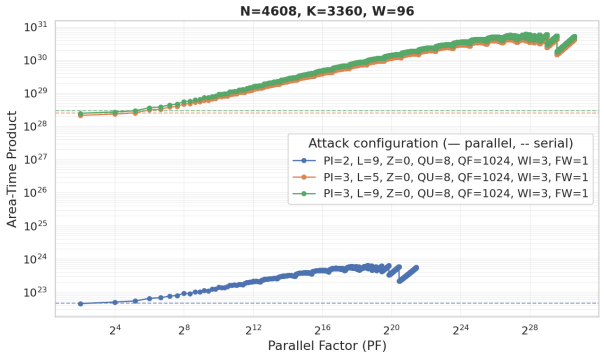

Figure 5: Area-time product as a function of `PF`, for $(n = 4608, k = 3360, w = 96)$

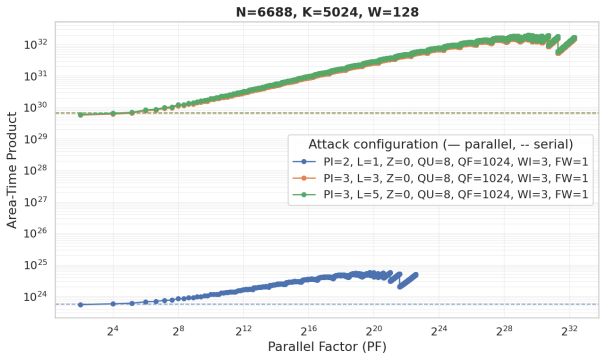

Figure 6: Area-time product as a function of `PF`, for $(n = 6688, k = 5024, w = 128)$

sets, ranging from 2^{189} to 2^{215} at security level L1 ($\lambda = 143$), from 2^{233} to 2^{287} at L3 ($\lambda = 207$), and from 2^{312} to 2^{360} at L5 ($\lambda = 272$).

5. Conclusion

This thesis presented a systematic study of the impact of hardware parallelism on ISD attacks against code-based cryptosystems, a family of post-quantum candidates. The main contributions are the design and implementation of `isd1_parallel`, a parallel variant of the `isd1` ISD algorithm, within the CAT framework [4], and an experimental analysis of depth speedup, area-time product, and security margin across the candidate schemes in the fourth round of the NIST standardization process for PQC [1].
The results show that, while a parallel adversary can significantly reduce the circuit depth of an ISD attack, the speedup is bounded by Amdahl's law and saturates due to inherently sequential phases. The area-time trade-off further limits the practical advantage of parallelism, and for large problem instances the speedup approaches unity.
Crucially, the NIST candidate parameters maintain a robust exponential security margin even under the parallel circuit model, confirming their compliance with the required security levels. The attack complexity remains exponential regardless of the parallelization factor.
Future work includes: extending the parallelization to the `isd2` variant; and transferring the analysis to the parallel Random-Access Memory (RAM) computational model to determine whether the observed limitations are intrinsic to ISD or specific to the Boolean circuit model.

References

[1] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. Status report on the third round of the NIST post-quantum cryptography standardization process. NIST Internal Report (NISTIR)

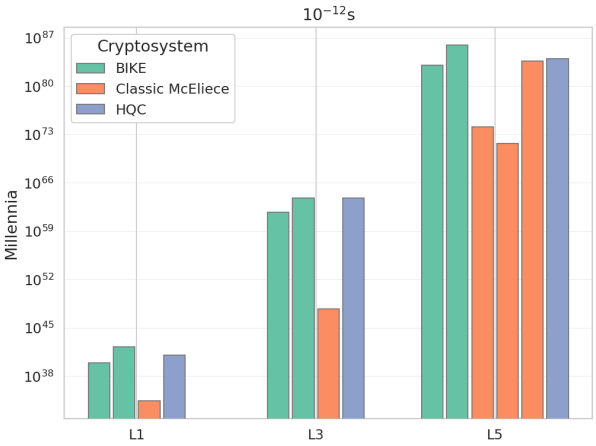

Figure 7: Representation of execution times in millennia for each security level, assuming a gate delay of 2^{-12}s. Individual instances for each cryptosystem are plotted separately.

8413, National Institute of Standards and Technology, Gaithersburg, MD, jul 2022.

[2] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in 2 n/20: How 1 + 1 = 0 improves information set decoding. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, Lecture Notes in Computer Science, pages 520–536. Springer, 2012.

[3] Daniel J. Bernstein. Understanding brute force, 2005.

[4] Daniel J. Bernstein and Tung Chou. Cryptattacktester: high-assurance attack analysis. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VI*, Lecture Notes in Computer Science, pages 141–182. Springer, 2024.

[5] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Smaller decoding exponents: Ball-collision decoding. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, Lecture Notes in Computer Science, pages 743–760. Springer, 2011.

[6] Ilya Dumer. Two decoding algorithms for linear codes. *Problems of Information Transmission*, 25, 03 1989.

[7] Donald E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley, 1973.

[8] Hans-Werner Lang, Manfred Schimmler, Hartmut Schmeck, and Heiko Schröder. Systolic sorting on a mesh-connected network. *IEEE Trans. Computers*, 34(7):652–658, 1985.

[9] Pil Joong Lee and Ernest F. Brickell. An observation on the security of mceliece's public-key cryptosystem. In Christoph G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, Lecture Notes in Computer Science, pages 275–280. Springer, 1988.

[10] Jeffrey S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Trans. Inf. Theory*, 34(5):1354–1359, 1988.

[11] John M. Marberg and Eli Gafni. Sorting in constant number of row and column phases on a mesh. *Algorithmica*, 3:561–572, 1988.

[12] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, Lecture Notes in Computer Science, pages 107–124. Springer, 2011.

[13] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory.

Technical Report 44, Jet Propulsion Laboratory, 1978.

[14] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):157–166, 1986.

[15] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Trans. Inf. Theory*, 8(5):5–9, 1962.

[16] Isaac D. Scherson and Sandeep Sen. Parallel sorting in two-dimensional VLSI models of computation. *IEEE Trans. Computers*, 38(2):238–249, 1989.

[17] Manfred Schimmler. Fast sorting on the instruction systolic array. Bericht 8709, Institut für Informatik, Christian-Albrechts-Universität Kiel, 1987.

[18] Claus-Peter Schnorr and Adi Shamir. An optimal sorting algorithm for mesh connected computers. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 255–263. ACM, 1986.

[19] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, November 20-22, 1994*, pages 124–134. IEEE Computer Society, 1994.

[20] Jacques Stern. A method for finding codewords of small weight. In Gérard D. Cohen and Jacques Wolfmann, editors, *Coding Theory and Applications, 3rd International Colloquium, Toulon, France, November 2-4, 1988, Proceedings*, Lecture Notes in Computer Science, pages 106–113. Springer, 1988.

Acronyms

BJMM Becker-Joux-May-Meurer. 2

CAT CryptAttackTester. 2, 3, 5

ECC Elliptic Curve Cryptography. 1

GDP General Decoding Problem. 1, 2

IS Information Set. 2
ISD Information Set Decoding. 1, 2, 5

MMT May-Meurer-Thomae. 2

NIST National Institute of Standards and Technology. 3–5

PQC Post-Quantum Cryptography. 1, 3, 5

RAM Random-Access Memory. 5
RS Redundancy Set. 2
RSA Rivest, Shamir, Adleman. 1

SDP Syndrome Decoding Problem. 1, 2