The McEliece Cryptosystem After Nearly Five Decades: A Survey of Security, Cryptanalysis, and Future Directions

Shabnam Jafarzade Mojaveri* Adel Khosravi†

Abstract

Almost fifty years after its introduction, the McEliece cryptosystem occupies an unusual place in the post-quantum landscape. Its public keys are far larger than those of most competing schemes, its original parameters no longer provide adequate security, and several compact variants proposed to reduce key size have subsequently been broken. Nevertheless, the binary Goppa code foundation retained in Classic McEliece continues to resist known practical attacks for the selected Classic McEliece parameter sets.

This survey asks why McEliece has remained relevant despite these limitations. We trace its development from the original 1978 encryption scheme to the modern Classic McEliece key-encapsulation mechanism and organize nearly five decades of cryptanalysis into generic decoding, structural recovery, attacks on compact variants, protocol-level attacks, implementation leakage, and quantum speedups. We emphasize several distinctions that are often blurred in discussions of the scheme: breaking an obsolete parameter set is not the same as recovering the hidden Goppa structure; distinguishing a public code does not necessarily lead to practical key recovery; and compromising a modified or structured variant does not automatically compromise Classic McEliece.

This history does not support either of two simple narratives: that McEliece has remained unchanged, or that it has simply been broken. Its longevity reflects a conservative mathematical foundation that has survived repeated reassessment, together with parameters, security models, and implementations that have evolved in response to new attacks. We close by outlining the main questions that will shape its future: whether public-key and key-distribution costs can be reduced without exposing exploitable structure, how far modern algebraic cryptanalysis can be extended, how classical and quantum security estimates should be refined, and how secure implementations can be integrated into practical systems.

Keywords: post-quantum cryptography; code-based cryptography; Classic McEliece; binary Goppa codes; information-set decoding; structural cryptanalysis

1 Introduction

The rapid development of quantum computing has raised important concerns about the long-term security of classical public-key cryptographic systems. Schemes such as RSA and discrete-logarithm-based systems, including DSA and elliptic-curve cryptography, rely on the difficulty of integer factorization or discrete logarithms [52, 65, 81]. These problems can be solved in polynomial time on a sufficiently powerful quantum computer using Shor's algorithm [85]. This has motivated extensive research into post-quantum cryptography (PQC), whose goal is to develop cryptographic systems that remain secure against both classical and quantum adversaries [9].

*Department of Computer Science, Stony Brook University, NY, USA. Emails: sjafarzademo@cs.stonybrook.edu

†Independent Researcher, NY, USA. Email: adel.khosravi.scholar@gmail.com.

Among the different families of post-quantum cryptosystems, code-based cryptography occupies a distinctive position [36,87]. In particular, the McEliece cryptosystem, introduced by Robert McEliece in 1978 [59], was the first code-based public-key encryption scheme. Its original construction hides a binary Goppa code whose algebraic description enables efficient decoding [72].

Over nearly five decades, McEliece has been the subject of sustained cryptanalytic research, implementation studies, and numerous proposed variants. Although many modified or compact variants have been broken or shown to introduce new vulnerabilities, no published attack currently gives a practical break of the selected Classic McEliece parameter sets based on binary Goppa codes [16, 27, 36].

The continued interest in the McEliece cryptosystem is particularly noteworthy given the rapidly evolving landscape of post-quantum cryptography. Because Classic McEliece deliberately retains the traditional and extensively studied binary Goppa code construction, it can be viewed as an important reference point for conservative code-based cryptographic design and long-term security evaluation [25, 27, 28].

As newer code-based and lattice-based schemes have emerged and standardization has advanced, McEliece has come to represent a distinctive trade-off [1]. Its large public keys and relatively costly key generation make deployment more difficult, but its small ciphertexts and efficient encapsulation remain valuable in applications where these costs can be accommodated [25, 26]. Its long record of public cryptanalysis provides an additional source of confidence [27].

Rather than only describing the construction, this survey traces how McEliece and its security analysis have changed over time. We follow how the scheme has changed over nearly fifty years, examine the main developments in cryptanalysis and implementation, and draw together the lessons that have emerged from both successful attacks and unsuccessful attempts. This perspective also helps explain the questions that continue to shape research on McEliece and code-based cryptography more generally.

The survey makes four main contributions. First, Sections 3 and 4 trace the development of McEliece from the original 1978 proposal to its present position and explain why it remains relevant despite its practical limitations. Second, Section 5 introduces the coding-theoretic ideas behind the scheme and clarifies the relationship among the original McEliece construction, the Niederreiter formulation, and the modern Classic McEliece KEM.

Third, Section 6 brings together the main lines of cryptanalysis, including generic decoding, structural key recovery, attacks on compact variants, protocol-level attacks, implementation leakage, and quantum speedups. For each line of work, it asks what the attack actually changes: the cost of decoding, the secrecy of the hidden code structure, the security of a modified construction, the behavior of a protocol, or the security of an implementation. This makes it possible to distinguish attacks on obsolete parameter sets and modified schemes from results that remain relevant to the selected Classic McEliece parameters. Finally, Section 7 identifies the open questions that will determine whether McEliece can retain its conservative security advantages while becoming easier to implement and deploy.

Scope and Relation to Prior Work

McEliece and code-based cryptography have been surveyed from several different perspectives. Engelbert, Overbeck, and Schmidt provided an early broad account of McEliece-type constructions and their security, with particular attention to alternative code families and structural attacks [36]. Repka and Zajac later focused more directly on the McEliece cryptosystem, reviewing its construction, parameter selection, generic and structural attacks, implementation challenges, and side-channel security [79]. Singh presented a more tutorial-oriented treatment of code-based cryptography

and Classic McEliece, emphasizing the underlying mathematics, parameter sets, and the NIST submission [87].

A second group of works considers McEliece within the broader development of code-based cryptography. Overbeck and Sendrier surveyed the main theoretical foundations, security assumptions, and cryptographic constructions in the area [71]. Subsequent overviews discussed progress in code-based encryption, signatures, implementation, and cryptanalysis [23, 78]. Sendrier provided a concise state-of-the-art perspective on the field and its prospects in post-quantum cryptography [84]. More recently, Weger, Gassner, and Rosenthal offered an extensive survey covering coding theory, the McEliece and Niederreiter frameworks, information-set decoding, algebraic and implementation attacks, and the code-based candidates considered during the NIST process [89]. General treatments of post-quantum cryptography place these systems within the wider landscape of quantum-resistant assumptions and constructions [9].

Among these works, the survey by Weger, Gassner, and Rosenthal is the closest in scope to the present paper [89]. Their work offers a broad, tutorial-oriented treatment of code-based cryptography as a whole. The present survey complements that account by taking a narrower, McEliece-centered approach. Rather than covering the full range of code-based constructions, it follows how the understanding of McEliece's security has changed from 1978 to 2026.

The aim is not simply to collect attacks, but to ask what each result actually established. An attack may lower the cost of generic decoding, recover hidden code structure, expose a weakness in a compact or modified variant, distinguish a public code without recovering the secret key, exploit protocol behavior, or compromise an implementation. These outcomes have different consequences for the security of Classic McEliece, even though they are sometimes grouped together under the general claim that "McEliece has been broken."

The survey connects these attacks with the parameter, implementation, and standardization changes that followed them. Some results made early parameter sets obsolete, while others affected only particular code families or modified constructions. Still others revealed implementation or protocol weaknesses without weakening the underlying decoding problem. By tracing these distinctions through 2026, the survey provides a McEliece-specific security history and includes recent developments that postdate the earlier surveys.

The scope of the paper is centered on the original binary Goppa-code construction and its modern realization in Classic McEliece. Other code-based cryptosystems are discussed when they provide a useful comparison or illustrate a broader design lesson, such as the risks introduced by additional algebraic structure. The paper does not attempt to survey every decoding algorithm, every code-based proposal, or the full theory of error-correcting codes. Instead, it focuses on the developments needed to understand McEliece's long security history, its current strengths and limitations, and the open problems most relevant to its future security and deployment.

For clarity, this survey uses the term *selected Classic McEliece parameter sets* for the parameter sets defined in the 2022 Round-4 specification. In numerical comparisons, these sets are grouped into five parameter families according to their underlying code parameters: `mceliece348864`, `mceliece460896`, `mceliece6688128`, `mceliece6960119`, and `mceliece8192128`. The base and `f` versions within each family use the same code parameters and have the same public-key and ciphertext sizes. The base versions use ordinary systematic key generation, whereas the `f` versions use $(\mu, \nu) = (32, 64)$-semi-systematic key generation. This allows more candidate matrices to be accepted during key generation, improving its success rate without changing the communication sizes [24, 25]. The term *ISO-standardized parameter families* is reserved for the subset included in the 2026 ISO amendment [30].

The literature review was updated through July 2026. We prioritize results that changed concrete decoding estimates, exposed structural information, affected protocol or implementation security, or

altered the standardization and deployment outlook. The coverage is therefore representative rather than exhaustive.

2 The Landscape of Post-Quantum Cryptography

Shor's algorithm showed that integer factorization and discrete logarithms can be solved in polynomial time on a sufficiently powerful quantum computer [85]. This result motivated the study of alternative cryptographic foundations for which no efficient classical or quantum attacks are currently known. These efforts have produced several families of post-quantum cryptographic schemes with different security assumptions, performance characteristics, and levels of maturity [9]. A brief comparison provides the context for understanding the distinctive position of the McEliece cryptosystem.

Hash-based cryptography

Hash-based signatures originate in the one-time signature construction of Lamport and the tree-based extension developed by Merkle [53, 60]. Their security relies mainly on the one-way and collision-resistance properties of cryptographic hash functions, giving them a comparatively conservative foundation. Modern hash-based signatures include the standardized stateless signature scheme SLH-DSA [68]. Their main practical limitations are relatively large signatures and, for stateful constructions, the need to manage signing state carefully.

Lattice-based cryptography

Lattice-based cryptography is built on computational problems involving high-dimensional lattices. An especially influential foundation is the Learning With Errors problem introduced by Regev, together with its structured variants [73, 77]. Lattice techniques now support standardized key-encapsulation and digital-signature schemes, including ML-KEM and ML-DSA [66, 67]. They have also enabled advanced primitives such as fully homomorphic encryption [42]. Their efficiency and versatility have made lattice-based constructions central to current post-quantum standardization, although their security and performance depend on the specific lattice problem, structure, and parameter choices.

Multivariate cryptography

Multivariate cryptography bases its security on the difficulty of solving systems of multivariate polynomial equations over finite fields. Early constructions showed how quadratic polynomial maps could support public-key encryption and digital signatures [55]. Multivariate schemes have often been attractive for their fast signature generation and verification. However, their security depends strongly on the hidden algebraic structure of each construction. For example, the prominent Rainbow signature scheme was practically broken by a key-recovery attack in 2022 [13]. Such results illustrate the difficulty of designing multivariate systems that remain secure under long-term structural cryptanalysis.

Isogeny-based cryptography

Isogeny-based cryptography uses computational problems involving maps between elliptic curves. A prominent line of research began with the Supersingular Isogeny Diffie–Hellman approach, which attracted attention partly because of its comparatively compact keys [48]. This area changed sharply after efficient key-recovery attacks were discovered against SIDH [21]. These attacks undermined

SIDH and closely related constructions, but they should not be interpreted as a break of every isogeny-based approach. Research continues on alternative isogeny problems and constructions that do not expose the auxiliary information exploited by the SIDH attacks.

Symmetric cryptography in the post-quantum setting

Symmetric cryptography is affected differently by quantum computing than public-key systems based on factoring or discrete logarithms. Grover's algorithm provides a generic quadratic speedup for exhaustive search, reducing a search over N possibilities to approximately $O(\sqrt{N})$ quantum operations [43]. Unlike Shor's algorithm, however, this does not produce a polynomial-time break of standard symmetric encryption. Its effect can generally be addressed by choosing larger security parameters. Symmetric encryption and hash functions therefore remain central to post-quantum systems, although their parameters must account for the relevant quantum search and collision-finding algorithms [9].

Code-based cryptography

Code-based cryptography derives its security from computational problems associated with error-correcting codes, most prominently the difficulty of decoding a random linear code [36,87]. Introduced in the late 1970s, it is one of the oldest approaches to post-quantum public-key cryptography. Its best-known representative is the McEliece cryptosystem, whose original construction hides an efficiently decodable binary Goppa code behind a public matrix that is intended to resemble a random linear code [59].

The modern Classic McEliece KEM retains this conservative binary Goppa code foundation. Its main advantage is the unusually long period of public cryptanalysis that the underlying construction has received. Its principal practical limitation is the size of its public keys, which can be much larger than those of competing post-quantum schemes. Classic McEliece nevertheless offers small ciphertexts, efficient encapsulation and decapsulation, and a high level of confidence derived from decades of analysis [25, 27].

To distinguish decoding-based security from other cryptographic uses of coding theory, it is useful to note that error-correcting codes can also serve as reconstruction mechanisms rather than hardness assumptions. For example, ProxCode uses Reed–Solomon codeword symbols as correlated shares in an approximate proximity-searchable-encryption construction for noisy biometric data [80]. Locality-sensitive hash outputs serve as lookup keys in a secure map. A query retrieves the corresponding codeword symbols, and the record can be reconstructed only after sufficiently many consistent symbols have been obtained.

This example illustrates two different roles of coding theory in cryptography. In McEliece, the hardness of decoding a random-looking code forms the security foundation of a post-quantum public-key system. In ProxCode, error-correcting codes instead provide an algebraic mechanism for privately searching and reconstructing noisy data.

The major post-quantum families differ substantially in their security assumptions, practical efficiency, and maturity. Table 1 summarizes their main strengths and limitations and places McEliece within this broader landscape.

Table 1: Comparison of the main families of post-quantum public-key cryptography.

Family	**Security Foundation**	**Main Strength**	**Main Limitation**
Hash-based	Cryptographic hash functions	Mature and conservative security assumptions	Primarily supports digital signatures, often with relatively large signatures
Lattice-based	LWE, Ring-LWE, and related lattice problems	Efficient and versatile; supports encryption, signatures, and advanced primitives	Security and performance depend on the chosen lattice structure and parameters; a shorter history of public cryptanalysis than McEliece
Code-based	**Decoding and related computational problems on linear codes**	**Long history of public cryptanalysis and small ciphertexts in Classic McEliece**	**Large public keys, especially in Classic McEliece**
Multivariate	Systems of multivariate polynomial equations	Potentially fast signature generation and verification	Many prominent constructions have been broken by structural and algebraic attacks
Isogeny-based	Hard problems involving elliptic-curve isogenies	Originally offered unusually compact keys	The practical break of the SIDH/SIKE line weakened confidence in leading constructions

Table 1 makes clear that no post-quantum family offers every desirable property. McEliece illustrates this trade-off especially well: its unusually large public keys make deployment more difficult, but its small ciphertexts and the confidence built over decades of public scrutiny remain valuable in settings where the key-size cost can be tolerated. Reducing that cost without introducing exploitable structure remains one of the central challenges for the scheme [1, 25, 27].

3 The Evolution of the McEliece Cryptosystem

The history of the McEliece cryptosystem is unusual in public-key cryptography. Proposed in 1978, in the same year as RSA, it was built on a very different mathematical foundation: the difficulty of decoding a random-looking linear code. Although its very large public keys have limited practical adoption, its binary Goppa code foundation has remained resistant to known practical attacks at modern parameter sets [27, 59].

The modern system is not identical in every detail to the original proposal. Nevertheless, its central design principle—hiding an efficiently decodable binary Goppa code behind a public code intended to resemble a random linear code—has remained at the core of McEliece-based cryptography for nearly five decades [27, 59]. Figure 1 summarizes the main developments in the design, cryptanalysis, and standardization of the system.

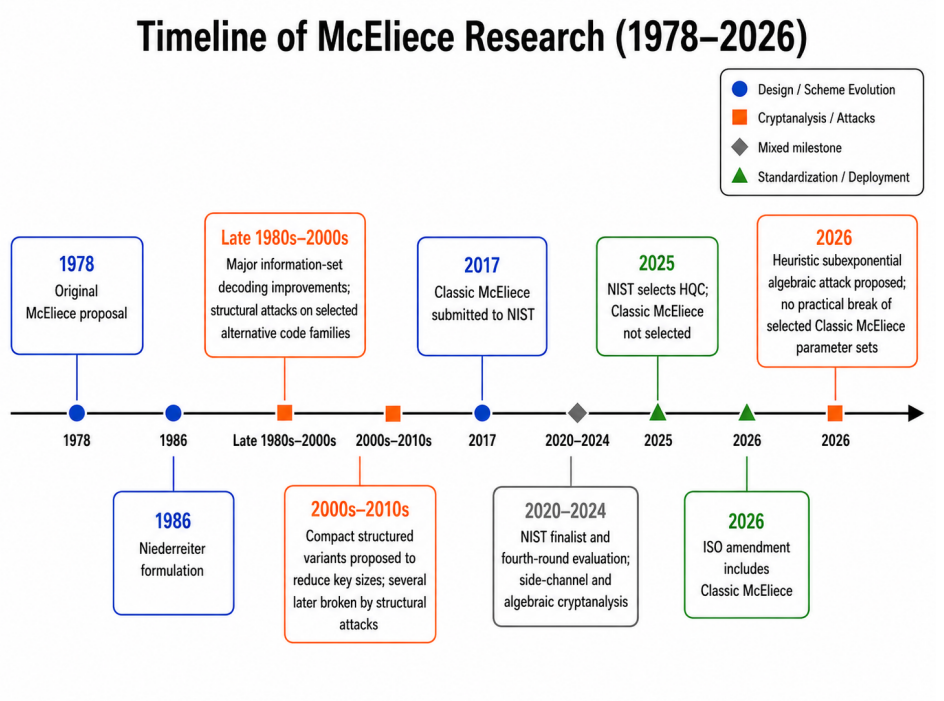


Figure 1: Timeline of key developments in the McEliece cryptosystem from 1978 to 2026.

3.1 The Original Proposal

McEliece's original construction used a binary Goppa code equipped with an efficient secret decoding algorithm. Starting from a generator matrix of this code, the private key applied an invertible scrambling matrix and a permutation of the code coordinates to produce the public generator matrix. Encryption encoded a message using this public matrix and added a random error vector of prescribed Hamming weight. A legitimate receiver could undo the secret transformation and correct the errors, whereas an attacker faced the much harder problem of decoding a seemingly unstructured linear code [59, 72].

The construction offered efficient encryption and decryption, but its public key was considerably larger than those of contemporary systems such as RSA. This limitation restricted practical adoption. Binary Goppa codes nevertheless proved to be a durable design choice: they combine efficient secret decoding with a public representation for which no practical structural key-recovery attack is currently known at the selected Classic McEliece parameter sets [25, 27].

3.2 The Niederreiter Reformulation and the Growth of Cryptanalysis

In 1986, Niederreiter introduced a closely related code-based construction formulated with a parity-check matrix and syndrome decoding rather than a public generator matrix [69]. When instantiated with the same code family, the McEliece and Niederreiter formulations rely on closely related decoding problems. The syndrome-based form later became particularly important because the modern Classic McEliece KEM is built from a Niederreiter-style encryption core using binary Goppa codes [24].

In parallel, researchers continued to improve generic decoding attacks, especially information-set decoding. These attacks do not attempt to recover the hidden Goppa structure; instead, they treat the public code essentially as a generic linear code and search for the error vector. Successive

improvements eventually made the parameter set proposed in 1978 inadequate by modern security standards [11, 17]. This was a failure of the original parameter sizes, not a structural break of the underlying binary Goppa code construction: security could be restored by choosing larger parameters.

A separate line of research sought smaller public keys by replacing binary Goppa codes with alternative code families or by adding algebraic structure. Some of these choices exposed recognizable patterns in the public code. In particular, generalized Reed–Solomon-based systems were broken by structural key-recovery attacks, and later distinguisher techniques compromised additional structured variants [31, 86]. These results helped establish an important design lesson: additional structure may improve efficiency and reduce key sizes, but it can also reveal information needed to reconstruct the secret code. The comparatively conservative binary Goppa construction has so far proved more resistant to practical structural recovery [16, 27].

3.3 The Post-Quantum Revival and Classic McEliece

For much of its history, McEliece was regarded as secure but difficult to deploy because of its large public keys. Interest in the scheme revived after Shor's algorithm showed that sufficiently powerful quantum computers could break cryptosystems based on factoring and discrete logarithms. In contrast, no polynomial-time quantum algorithm is known for decoding random linear codes [9,34,85].

This renewed interest led to Classic McEliece, which was submitted to the NIST Post-Quantum Cryptography Standardization Project in 2017. Classic McEliece retains the binary Goppa code foundation of the original system but uses a Niederreiter-style encryption core and is specified as a key-encapsulation mechanism. It is designed for IND-CCA2 security, and honestly generated ciphertexts do not suffer from decapsulation failures [24, 28].

Classic McEliece eventually reached NIST's fourth round of evaluation. Its long history of public cryptanalysis remained a major strength, but its very large public keys and relatively expensive key generation continued to limit its appeal for general-purpose deployment [1, 25, 26].

3.4 Standardization and McEliece in 2026

In March 2025, NIST selected HQC for standardization and did not select Classic McEliece. This was not presented as evidence of a practical security break. NIST nevertheless noted that independent estimates placed `mceliece460896` and `mceliece460896f` below their claimed Category 3 target, although they were still assessed as providing at least Category 2 security. The report also emphasized the scheme's very large public keys and uncertainty about whether it would be adopted widely enough to justify another standard [1].

Bernstein, writing in a personal capacity as a member of the Classic McEliece team, disputes this framing. He argues that the parameter sets were chosen by optimizing security under key-size constraints rather than by working backward from fixed NIST categories, and points to higher bit-operation estimates from CryptAttackTester [8, 10].

The decision illustrates the difference between security confidence and deployability. Classic McEliece is most attractive when a public key can be distributed once and reused for many encapsulations. In such settings, its small ciphertexts and low cost per encapsulation can help offset the initial cost of distributing and generating the key [25, 26].

Classic McEliece nevertheless gained an international standardization path. In June 2026, it was included in ISO/IEC 18033-2:2006/Amd 2:2026 [47]. The amendment includes the `mceliece460896`, `mceliece6688128`, `mceliece6960119`, and `mceliece8192128` families, together with their `f`, `pc`, and `pcf` variants, but not the `mceliece348864` family. The Classic McEliece team recommends

the `mceliece6*` sizes, corresponding to the `mceliece6688128` and `mceliece6960119` families, for long-term security [30, 47]. Structural cryptanalysis also continued to develop. A 2026 preprint proposed a heuristic subexponential algebraic key-recovery method for certain binary Goppa code instances, but it did not break the selected Classic McEliece parameter sets [16].

By 2026, the parameters, security model, and standardization status of McEliece had changed considerably, but its central reliance on binary Goppa codes had remained intact.

4 Why McEliece Still Matters

The continued relevance of McEliece does not come from efficiency alone. Classic McEliece requires much larger public keys than many competing post-quantum schemes, which limits its suitability for some applications. Its importance instead rests on a conservative binary Goppa code foundation, an unusually long history of public analysis, and a comparatively well-understood security trade-off [1, 25, 27].

This combination makes McEliece useful in two ways. It remains a practical option in settings where a public key can be distributed once and reused for many encapsulations, allowing its small ciphertexts to offset part of the key-distribution cost. It also serves as a useful reference point for evaluating the security claims of newer code-based constructions [25, 26].

4.1 The Security Record of McEliece

The age of a cryptosystem is not, by itself, evidence of security. What distinguishes McEliece is the range of attacks that have been developed against it. Researchers have improved generic decoding algorithms, searched for hidden algebraic structure, reconsidered parameter choices, and studied implementation leakage. These efforts made the parameters proposed in 1978 obsolete and broke several modified constructions, but they did not produce a practical structural break of the binary Goppa code foundation used in the selected Classic McEliece parameter sets [11, 27, 36].

Classic McEliece should therefore be judged separately from obsolete parameters and broken variants. Confidence in Classic McEliece does not rest on the claim that every McEliece-type construction survived. Rather, insecure parameter sets were enlarged, and variants whose additional structure created recognizable weaknesses were discarded. The conservative binary Goppa code construction remained after these changes.

Recent work also shows that structural cryptanalysis is still developing. A 2026 study introduced a new algebraic model and a heuristic subexponential key-recovery direction for certain binary Goppa code instances. The method succeeded against selected challenge and related instances, but it did not provide a practical break of the selected Classic McEliece parameter sets [16].

McEliece therefore matters not because its security history is finished, but because its central construction has remained viable while attacks, parameters, and security requirements have continued to evolve.

4.2 A Conservative Security Foundation

Classic McEliece is described as conservative not because its security is formally proved, but because it retains the traditional binary Goppa code construction and avoids additional structure introduced mainly to reduce key sizes. Its two principal cryptanalytic threats are generic decoding, especially information-set decoding, and structural attacks that attempt to recover an equivalent private key. Both directions have been studied extensively, but no published attack currently gives a practical break of the selected Classic McEliece parameter sets [16, 25, 27].

Known quantum algorithms can reduce the cost of generic search and decoding attacks, but no polynomial-time quantum algorithm is known for the decoding problems underlying the scheme. Quantum information-set decoding therefore affects concrete security estimates and parameter selection rather than invalidating the basic construction [7, 34, 49, 50].

The experience with compact McEliece variants reinforces this design choice. Several constructions based on alternative or highly structured code families were broken by algebraic and structural attacks. Classic McEliece instead accepts larger public keys in order to avoid many of the structural shortcuts introduced by such variants [25, 31, 36, 86].

4.3 Security Versus Practical Deployment

The main practical cost of Classic McEliece is its public-key size. Across the five Round-4 parameter families, the base and `f` variants have public keys ranging from 261,120 to 1,357,824 bytes and ciphertexts ranging from 96 to 208 bytes. For the ISO-standardized `pc` and `pcf` variants, ciphertexts are 32 bytes longer. Key generation is also comparatively expensive [1, 29, 30].

The trade-off is more favorable when a long-lived public key can be distributed once and reused across multiple encapsulations. In that setting, the cost of the large key is amortized, while the small ciphertexts limit the communication required for each subsequent exchange. This benefit must nevertheless be considered together with multi-instance security bounds and an appropriate key-rotation policy [25, 26, 58].

This distinction helps explain the scheme's standardization history. In 2025, NIST selected HQC as the only fourth-round KEM for standardization and did not select Classic McEliece, with deployment costs and uncertainty about widespread adoption playing important roles in the decision. Classic McEliece was nevertheless included in ISO/IEC 18033-2:2006/Amd 2:2026. These outcomes show that confidence in a scheme's security and its suitability for general-purpose deployment are related but separate considerations [1, 47].

4.4 McEliece as a Security Benchmark

McEliece remains important even when it is not the preferred choice for general-purpose deployment. Its conservative binary Goppa code construction provides a useful reference point for evaluating newer code-based schemes. Proposals with smaller keys can be compared against McEliece by asking what additional structure they introduce, what efficiency they gain, and whether those choices create new structural weaknesses [25, 27, 36].

The scheme is also an important case study in implementation security. Practical studies have shown that unprotected implementations can leak enough information through power consumption or other physical effects to support message or key recovery [14, 44]. These attacks do not contradict the hardness assumptions underlying McEliece; they show that mathematical security must be supported by careful implementation. Constant-time software and protected hardware designs have therefore been developed to address timing, side-channel, and fault-injection threats [26, 41].

McEliece thus serves both as a practical cryptosystem for suitable settings and as a long-running test of conservative code-based design. Its history helps researchers study the trade-off among key size, algebraic structure, implementation cost, and confidence gained through sustained cryptanalysis.

5 Foundations of the McEliece Cryptosystem

The long security history of the McEliece cryptosystem is closely tied to its mathematical design. Binary Goppa codes can be decoded efficiently when their hidden algebraic description is known,

while the public generator or parity-check matrix is intended not to reveal that structure. The legitimate receiver therefore has an efficient decoding trapdoor, whereas an attacker must either solve a difficult decoding problem using the public code or reconstruct an equivalent hidden code [27,59,72].

5.1 Linear Codes and Decoding Hardness

A binary linear $[n, k, d]$ code C is a k-dimensional subspace of $\mathbb{F}_2^n$, where n is the code length and d is the minimum Hamming distance between distinct codewords. A message $m \in \mathbb{F}_2^k$ is encoded as $x = mG$ using a generator matrix $G \in \mathbb{F}_2^{k\times n}$. Such a code can uniquely correct up to $t = \lfloor (d-1)/2 \rfloor$ errors.

The same code may be represented by a parity-check matrix $H \in \mathbb{F}_2^{(n-k)\times n}$ satisfying $GH^T = 0$. If a codeword x is altered by an error vector e, the received vector is $y = x + e$, and its syndrome is $s = Hy^T = H(x + e)^T = He^T$. The syndrome therefore depends only on the error pattern. For a code equipped with an efficient decoding algorithm, it can be used to recover e and reconstruct the original codeword.

Code-based cryptography turns this error-correction process into a cryptographic mechanism. The sender deliberately introduces an error, while the legitimate receiver uses hidden decoding information to remove it efficiently. An attacker, given only the public code, must either recover the error through generic decoding or reconstruct the secret algebraic structure that enables efficient decoding [27, 59].

The decision version of syndrome decoding for general binary linear codes is NP-complete [6]. This result provides theoretical motivation for code-based cryptography, but it is not by itself a proof of McEliece security. The security of a concrete parameter set depends on the difficulty of the decoding instances it generates and on whether the hidden Goppa code can be recognized or recovered from its public representation.

5.2 Binary Goppa Codes as a Cryptographic Trapdoor

The original McEliece construction uses binary Goppa codes [59]. Let $\mathbb{F}_{2^m}$ be an extension of $\mathbb{F}_2$, let $L = (\alpha_1, \ldots, \alpha_n)$ be a sequence of distinct elements of $\mathbb{F}_{2^m}$, and let $g(X) \in \mathbb{F}_{2^m}[X]$ be a polynomial of degree t satisfying $g(\alpha_i) \neq 0$ for every i. The corresponding binary Goppa code is

$$\Gamma(L, g) = \left\{ c = (c_1, \ldots, c_n) \in \mathbb{F}_2^n \;\middle|\; \sum_{i=1}^{n} \frac{c_i}{X - \alpha_i} \equiv 0 \pmod{g(X)} \right\}.$$

For the square-free binary Goppa codes used in McEliece-type constructions, the minimum distance is at least $2t + 1$ and the dimension satisfies $k \geq n - mt$. The code can therefore correct up to t errors. Knowledge of the support L and the Goppa polynomial $g(X)$ enables efficient decoding, for example through Patterson's algorithm [27, 72].

The pair (L, g) forms the essential decoding trapdoor. The public key contains a generator or parity-check matrix derived from the code but does not reveal this algebraic description directly. An attacker must therefore either decode using the public code or recover enough of the hidden Goppa structure to construct an equivalent decoder [27].

This choice involves a deliberate trade-off. Several McEliece variants introduced alternative code families or additional algebraic symmetry to reduce public-key sizes, but some of that added structure enabled distinguishing and key-recovery attacks. Classic McEliece instead retains the traditional binary Goppa construction without such key-compression symmetries [16, 25, 36].

5.3 The Original McEliece Encryption Scheme

Let C be a binary $[n, k, d]$ Goppa code with generator matrix $G' \in \mathbb{F}_2^{k\times n}$ and a secret decoding algorithm capable of correcting up to t errors, where $d \geq 2t+1$. Key generation chooses an invertible matrix $S \in \mathbb{F}_2^{k\times k}$ and a permutation matrix $P \in \mathbb{F}_2^{n\times n}$, and publishes $G = SG'P$ together with t. The private key contains S, P, and the algebraic description needed to decode the hidden Goppa code [36, 59].

To encrypt a message $m \in \mathbb{F}_2^k$, the sender chooses a random error vector $e \in \mathbb{F}_2^n$ of Hamming weight t and computes

$$c = mG + e.$$

For decryption, the receiver first removes the secret permutation:

$$cP^{-1} = mSG' + eP^{-1}.$$

Because a permutation preserves Hamming weight, eP^{-1} is still correctable by the secret Goppa decoder. Decoding therefore recovers the information vector mS, after which multiplication by S^{-1} gives the original message m.

The construction faces two main classes of attack. A generic decoding attack attempts to recover the error or message directly from the public code, while a structural attack tries to reconstruct an equivalent efficiently decodable code from the public matrix [27]. These possibilities motivate the division between decoding and structural cryptanalysis used later in this survey.

The original construction should also be distinguished from its 1978 parameter values. Improvements in information-set decoding made those parameters inadequate by modern security standards, but did not yield a structural break of the binary Goppa code design. Modern parameter sets are therefore substantially larger and are chosen using updated attack estimates [11, 17].

5.4 Niederreiter and Classic McEliece

Niederreiter introduced a closely related code-based construction in 1986. Instead of publishing a generator matrix and encrypting a message by adding errors, the Niederreiter formulation publishes a parity-check matrix and represents the secret value as a low-weight error vector [69].

Let $H \in \mathbb{F}_2^{(n-k)\times n}$ be a parity-check matrix of the secret code. In the standard description, key generation chooses an invertible matrix $S \in \mathbb{F}_2^{(n-k)\times(n-k)}$ and a permutation matrix $P \in \mathbb{F}_2^{n\times n}$, and publishes

$$H_{\text{pub}} = SHP.$$

For an error vector $e \in \mathbb{F}_2^n$ of Hamming weight t, the ciphertext is its public syndrome,

$$s = H_{\text{pub}}e^T.$$

The receiver first computes $S^{-1}s = HPe^T$. The secret decoder therefore recovers the permuted error vector $\widehat{e}^T = Pe^T$. The original error vector is then obtained by applying the inverse permutation:

$$e^T = P^{-1}\widehat{e}^T.$$

The McEliece and Niederreiter formulations use the same underlying trapdoor but package the encoded information differently. In Niederreiter encryption, the recovered low-weight vector can

itself be used as secret material, making the syndrome formulation particularly natural for key encapsulation [36].

Classic McEliece adopts this syndrome-based viewpoint. Its public key is stored in a systematic representation rather than directly as the textbook matrix SHP, but the central mechanism is the same: encapsulation samples a low-weight vector, computes a syndrome-based ciphertext, and derives a shared key from the resulting values. Decapsulation uses the hidden binary Goppa decoder to recover the vector and derive the same key [24].

The modern construction also includes confirmation and deterministic processing of invalid ciphertexts to obtain chosen-ciphertext security. These additions strengthen the security model without changing the underlying trapdoor: the public key describes a code through a parity-check representation, while the private key contains the algebraic information required for efficient Goppa decoding [24, 27].

The matrices S and P should not be viewed as separate sources of security. The matrix S changes the basis used to represent the same code, while P permutes its coordinates. An attacker therefore does not need to recover the original matrices: finding any equivalent Goppa code together with an efficient decoder would be enough. Sendrier's support-splitting algorithm shows that, once two permutation-equivalent codes are known, their coordinate permutation can often be recovered from permutation-invariant properties of the codes. The algorithm does not recover the hidden Goppa code from the public key by itself. Instead, it helps clarify that McEliece security rests on hiding an efficiently decodable equivalent code, rather than on keeping S and P secret in isolation [27, 82].

6 Five Decades of Cryptanalysis: Attacks and Lessons Learned

McEliece has been studied through several distinct forms of cryptanalysis. Generic decoding attacks ignore the hidden Goppa code structure and attempt to recover the error vector directly from the public code. Structural attacks instead search for algebraic properties that distinguish the public code from a random linear code or reveal an equivalent secret decoder. Other attacks target compact variants, encryption transformations, or concrete implementations rather than the underlying decoding problem.

These distinctions are essential when interpreting the security record of McEliece. Improvements in decoding algorithms made the original parameter choices obsolete, structural attacks broke several modified constructions, and side-channel attacks exposed weaknesses in unprotected implementations [11, 17, 31, 36, 44]. None of these results, however, should automatically be interpreted as a practical break of the binary Goppa code construction at the selected Classic McEliece parameter sets. No published attack currently provides such a break, although recent algebraic work shows that structural cryptanalysis remains active [16, 27].

This section organizes the principal attacks according to their target and objective, and then examines what each line of research has revealed about the security of McEliece.

6.1 A Taxonomy of Attacks

The cryptanalytic literature on McEliece-type systems can be organized into six broad categories, summarized in Table 2. The categories are not completely disjoint. Variant-specific attacks often use structural techniques, while quantum algorithms generally accelerate generic decoding or search procedures. They are separated here because they target different parts of the system and have different implications for the security of the underlying binary Goppa code construction.

The distinction among these categories prevents overly broad conclusions from individual attack results. A successful attack on a compact variant, protocol transformation, or implementation may

be serious without constituting a structural break of binary Goppa codes. Likewise, an improvement in generic decoding may require larger parameters without revealing the hidden private code [27, 36].

Table 2: Taxonomy of the main attack categories against McEliece-type cryptosystems.

Attack Category	**Main Target**	**Goal**	**Typical Scope**
Generic decoding (ISD)	Ciphertext and public code	Recover the error vector or plaintext without exploiting the hidden algebraic structure	Original construction and McEliece-type variants
Structural	Algebraic structure of the public code	Distinguish the public code from a random code or reconstruct an equivalent secret decoder	Depends strongly on the underlying code family
Structured-variant attacks	Additional symmetry or key-compression structure	Use structural properties introduced to reduce key size or improve efficiency	Compact or modified McEliece-type constructions
Protocol-level	Encryption or encapsulation interface	Exploit malleability, reaction behavior, invalid-ciphertext handling, repeated messages, or weaknesses in a security transformation	Basic encryption schemes and improperly protected protocols
Implementation	Secret-dependent software or hardware behavior	Recover information through timing, power, electromagnetic, cache, or fault leakage	Particular implementations rather than the abstract scheme
Quantum	Generic decoding and search procedures	Reduce the computational cost of classical attacks using quantum search or quantum-walk techniques	Any construction attacked through generic decoding or search

6.2 Generic Decoding Attacks

Generic decoding attacks ignore the hidden algebraic structure of the secret Goppa code. They treat the public code as a general linear code and attempt to recover the deliberately introduced error vector. In the original McEliece formulation, the attacker is given

$$c = mG + e,$$

where G is the public generator matrix and e has Hamming weight t. In the Niederreiter formulation used by Classic McEliece, the corresponding public value is the syndrome

$$s = H_{\text{pub}} e^T.$$

In both cases, the underlying task is to find a low-weight error vector without using the hidden Goppa structure [27, 59, 69].

For clarity, we describe Prange's attack in the generator-matrix formulation; the same decoding problem can equivalently be expressed in the syndrome formulation.

Information-set decoding remains the main framework used in concrete security estimates for Classic McEliece, although recent RLPN-style methods have broadened the landscape of generic decoding attacks.

6.2.1 Prange's Basic Attack

Prange's method begins by choosing a set I of k coordinates. Let $G \in \mathbb{F}_2^{k\times n}$ generate an $[n, k]$ code. The set I is an information set when the submatrix G_I, formed by the corresponding columns of G, is invertible. If none of the errors lies in I, then

$$c_I = mG_I,$$

and the attacker can recover a candidate message as

$$m = c_I G_I^{-1}.$$

The candidate is accepted if $c - mG$ has Hamming weight t; otherwise, another information set is tested.

For a uniformly random set I of k coordinates, the probability that it contains none of the t error positions is

$$p_0 = \frac{\binom{n-t}{k}}{\binom{n}{k}} = \frac{\binom{n-k}{t}}{\binom{n}{t}}.$$

A valid iteration also requires G_I to be invertible. For a uniformly random binary $k \times k$ matrix, the probability of invertibility is

$$\rho_k = \prod_{j=1}^{k} \left(1 - 2^{-j}\right) \approx 0.29.$$

Under the standard assumption that invertibility and error-freeness are effectively independent for the public matrices under consideration, one iteration of Prange's algorithm therefore succeeds with probability approximately $\rho_k p_0$. The expected number of iterations is accordingly

$$\frac{1}{\rho_k p_0},$$

apart from the polynomial cost of linear algebra and candidate verification [27, 75]. For the selected Classic McEliece parameters, the resulting attack costs remain far beyond practical computation under current estimates [27].

6.2.2 From Prange to Modern ISD

Prange's algorithm provides the basic framework for information-set decoding, but later work introduced increasingly sophisticated ways to reduce its running time. Lee and Brickell allowed a small number of errors inside the information set, so that the cost of row reduction could be reused across several candidate error patterns [54]. Stern introduced collision-based partial-syndrome matching [88]. Dumer then refined this approach by working with an enlarged information set and using a two-list meet-in-the-middle search [35]. Later algorithms developed these ideas through multi-level list merging and multiple representations of the same candidate solution. The MMT algorithm adapted the representation technique of Howgrave-Graham and Joux, originally developed for subset-sum algorithms, to information-set decoding [46, 56]. BJMM expanded the number and structure of these representations [4]. May and Ozerov then replaced exact list matching with nearest-neighbor search, obtaining a further asymptotic improvement [57].

To compare these algorithms, we write their asymptotic time and memory requirements as

$$T(n) = \widetilde{O}(2^{\tau n}) \qquad \text{and} \qquad M(n) = \widetilde{O}(2^{\mu n}),$$

where τ and μ denote the time and memory exponents, respectively. These exponents generally depend on the code rate, the relative error weight, and the internal parameters of the algorithm.

Dumer's algorithm is discussed above, but Table 3 lists selected benchmark exponents rather than every intermediate ISD variant.

Table 3 reports commonly cited exponents for the standard asymptotic maximum-likelihood-decoding benchmark for random binary codes, in which the target error weight is taken near one-half of the Gilbert–Varshamov distance. The Lee–Brickell through May–Ozerov values follow the comparative analysis of Bouillaguet, Delaplace, and Hamdad [15]. The Prange value is the corresponding half-Gilbert–Varshamov-distance estimate reported in the Classic McEliece security analysis [27]. These figures illustrate the asymptotic development of ISD and should not be interpreted as concrete security estimates for Classic McEliece.

Table 3: Selected milestones in classical information-set decoding. The time and memory requirements are expressed as $T(n) = \widetilde{O}(2^{\tau n})$ and $M(n) = \widetilde{O}(2^{\mu n})$. The entries are benchmark asymptotic exponents for maximum-likelihood decoding of random binary codes near one-half of the Gilbert–Varshamov distance; they are not concrete attack costs for Classic McEliece [15, 27].

Algorithm	**Year**	**Main technique**	τ	μ
Prange [75]	1962	Search for an error-free information set	0.0576	0
Lee–Brickell [54]	1988	Allow a small number of errors in the information set	0.0575	0
Stern [88]	1989	Collision search and partial-syndrome matching	0.0556	0.0135
MMT [56]	2011	Multi-level list merging with multiple representations	0.0536	0.0216
BJMM [4]	2012	Generalized and recursively constructed representations	0.0493	0.0331
May–Ozerov [57]	2015	Nearest-neighbor search in place of exact list matching	0.0473	0.0346

The progression in Table 3 illustrates a clear time–memory trade-off. Later algorithms reduce the asymptotic time exponent through increasingly sophisticated list-merging, representation, and nearest-neighbor techniques, generally at the cost of exponential memory. The improvement from Prange to May–Ozerov is therefore meaningful in the conventional half-distance benchmark, but the lowest asymptotic time exponent does not automatically correspond to the most practical attack at cryptographic parameter sizes.

For McEliece, however, the error weight behaves differently. The half-distance benchmark assumes an error weight proportional to the code length, whereas in asymptotic McEliece parameter families the error weight satisfies $t = o(n)$. In that regime, the known ISD improvements do not change the leading asymptotic term obtained from Prange's algorithm; instead, they improve lower-order terms and concrete costs [18, 27]. Moreover, the apparent advantage of representation-based algorithms can depend strongly on how access to large amounts of memory is priced [27]. Concrete security must therefore be evaluated separately for the actual Classic McEliece parameter sets and under an explicit memory-cost model.

6.2.3 Generic Decoding Beyond ISD

Information-set decoding is not the only modern approach to generic decoding. Carrier, Debris-Alazard, Meyer-Hilfiger, and Tillich revisited statistical decoding by using low-weight parity checks to construct samples for a Learning Parity with Noise problem. Their method, known as RLPN decoding, was reported to outperform ISD in asymptotic regimes with code rates below approximately 0.3 [19].

A later refinement, Dual Attack 3.0, observes that the intermediate problem has a sparse secret and introduces a further reduction from sparse LPN to ordinary LPN. The resulting analysis reports an improvement over both the earlier RLPN method and ISD for code rates below approximately 0.42 [20].

RLPN and Dual Attack 3.0 broaden the range of generic decoding methods, but ISD remains the relevant concrete baseline for Classic McEliece. The selected Classic McEliece parameter sets have code rates between approximately 0.73 and 0.80, well above the regimes in which the known RLPN-style methods report an advantage [24]. Consequently, the concrete security estimates considered in the next subsection continue to focus on ISD.

6.2.4 Concrete ISD Costs for Classic McEliece

The asymptotic exponents discussed above do not by themselves determine the security of these finite instances. Concrete estimates also depend on n, k, and t, the internal attack parameters, and the cost assigned to constructing and accessing large amounts of memory.

Table 4 summarizes the lowest classical ISD cost reported for each of the five Round-4 Classic McEliece parameter families under three memory-cost models [27, 37]. The entries are base-2 logarithms of the estimated attack cost. The free-memory model assigns no cost to memory access, while the other two models assign cube-root and square-root costs, respectively.

The values are reproduced from the concrete estimates reported in the Classic McEliece security analysis; they were not independently recomputed for this survey.

Table 4: Representative concrete classical ISD estimates for the five Round-4 Classic McEliece parameter families. Each entry is the minimum reported $\log_2$ cost among the ISD variants considered by the Esser–Bellini estimator. The three columns use different assumptions about the cost of memory access [27].

Parameter family	Free memory	Cube-root memory cost	Square-root memory cost
`mceliece348864`	140.8	156.4	158.6
`mceliece460896`	179.8	198.6	201.0
`mceliece6688128`	246.2	274.9	278.2
`mceliece6960119`	245.7	275.8	279.2
`mceliece8192128`	275.6	311.9	315.6

The differences among the columns illustrate how strongly concrete ISD estimates depend on the treatment of memory. For `mceliece348864`, for example, the lowest estimate rises from $2^{140.8}$ in the free-memory model to $2^{158.6}$ under the square-root model. Similar gaps appear for the larger parameter sets. Memory-intensive algorithms therefore look substantially more advantageous when access to large lists is treated as free.

Concrete experiments also show the practical side of this progress. Esser, May, and Zweydinger implemented the representation-based MMT and BJMM algorithms and used them to set decoding records for McEliece instances up to code length 1284 and quasi-cyclic instances up to length 2918. Their extrapolations suggested that the security of the Round-3 192-bit parameter set had been overestimated by roughly 20 bits, and that of two of the three Round-3 256-bit parameter sets by roughly 10 bits. The solved instances were nevertheless much smaller than the selected Classic McEliece parameter sets, so these computations improved concrete ISD estimates rather than providing a practical break of the scheme [38].

These estimates should not be interpreted as exact security levels or compared directly with thresholds stated in bit operations. Table 4 uses the operation-counting conventions of the Esser–Bellini estimator, whereas CryptAttackTester uses a more detailed bit-operation model and reports a cost of approximately $2^{150.59}$ for `mceliece348864`. The difference from the $2^{140.8}$ free-memory estimate reflects differences in the attacks included and in how their costs are counted, rather than a direct ten-bit disagreement about the security of the parameter set [10, 27].

These differences among cost models also provide context for NIST's qualification of the claimed security category of the `mceliece460896` family, discussed in Section 3.4 [1].

The gradual improvement of ISD also explains why the parameters proposed in 1978 are no longer adequate by modern standards. Bernstein, Lange, and Peters estimated that an optimized decoding attack against the original parameter set was within reach of a moderately sized parallel computation [11]. This result concerned the size of the original parameter set; it neither recovered the hidden Goppa structure nor provided an efficient decoder for arbitrary modern parameter sets.

Classic McEliece therefore uses substantially larger parameters selected to withstand the best known classical and quantum decoding attacks [7, 25, 27]. As decoding algorithms and the models used to estimate their costs improve, these security estimates must be revisited even though the underlying binary Goppa-code construction remains unchanged [27].

For the selected Classic McEliece parameters, ISD remains the principal concrete generic baseline against which other attacks are measured. Because it does not exploit a special weakness in the public code, a structural attack becomes especially significant only when it uses the Goppa-code origin of the key to outperform the best generic decoding methods.

Multi-instance security. A separate consideration arises when one public key is reused for many encapsulations. In this setting, an attacker obtains several syndromes associated with the same public code and needs to decode only one of them. This problem was studied earlier under the name decoding one out of many [83].

May and Sá Diogo recently applied this setting to modern code-based KEMs. Under the classical cost model used in the revised version of their analysis, the estimated work factor for `mceliece348864` falls below their 143-bit Level-1 comparison benchmark when at least $M = 2^{21}$ session keys are generated under the same public key. This threshold depends on the attack and cost assumptions used in that analysis, so it should not be read as a universal key-rotation limit. The result is a multi-instance decoding speedup rather than a structural attack on the hidden Goppa code. It nevertheless shows that deployment analyses should account for the number of encapsulations performed under one key and choose an appropriate key-rotation policy [58].

6.3 Structural Distinguishers and Key-Recovery Attacks

Generic decoding attacks ignore the origin of the public code. Structural attacks instead ask whether its public representation reveals algebraic properties of the hidden code family. Their goal may range from distinguishing the public code from a random linear code to recovering an equivalent private description that supports efficient decoding. A successful key-recovery attack is potentially more damaging than message recovery because the recovered decoder may compromise all ciphertexts created under the same public key [27, 36].

From distinguishers to key recovery. A distinguisher identifies a statistical or algebraic property that occurs differently in the secret code family than in a random code. This shows that some information about the code's origin remains visible, but it does not by itself break the cryptosystem. A full key-recovery attack must go further and reconstruct enough information about

the support, multipliers, defining polynomial, or an equivalent code to build an efficient decoder. The recovered description need not be identical to the legitimate private key; any equivalent decoding trapdoor is sufficient.

Early structural attacks. An important early example was the attack of Sidelnikov and Shestakov on McEliece-type systems based on generalized Reed–Solomon codes. They showed that the evaluation structure of a disguised GRS code could be recovered efficiently from its public matrix, yielding an equivalent private decoder [86]. This result demonstrated that generic decoding hardness alone is not enough: the code family must also resist recognition and structural reconstruction.

Binary Goppa codes belong to the broader family of alternant codes, but their subfield construction conceals much of the structure that is directly visible in a GRS code. The Sidelnikov–Shestakov method therefore does not directly recover the binary Goppa codes used in Classic McEliece. Structural attacks were nevertheless highly successful against several alternative code families and compact variants that exposed additional algebraic symmetry [31, 36].

Modern algebraic techniques. Later work developed distinguishers based on component-wise products of codewords, commonly studied through square codes. Evaluation codes such as GRS codes satisfy algebraic relations that can cause their square codes to have smaller dimension than those of comparable random codes. These techniques led to distinguishers and key-recovery attacks against several GRS-based constructions [31]. Subsequent work clarified analogous square-code structure in alternant and Goppa codes [64].

This line of research was later extended to a polynomial-time key-recovery attack against generic high-rate random alternant codes. The attack applies in the parameter regimes where the underlying high-rate distinguisher is effective [3].

A more recent approach studies the matrix code of quadratic relations among codewords. For Goppa codes, this matrix space can contain unusually low-rank elements that reflect the hidden polynomial structure. Finding such elements can distinguish Goppa-code instances and, in favorable parameter ranges, support recovery of an equivalent secret key [32, 63].

A different recent direction uses syzygies and graded algebraic invariants to distinguish alternant and Goppa codes. The resulting distinguisher applies to parameter regimes covering the codes used in Classic McEliece, but it does not by itself recover an equivalent private key or provide a practical break of the selected Classic McEliece parameter sets [76].

Building on the quadratic-relation and low-rank-matrix approach, a 2026 study introduced a new algebraic model for finding rank-two matrices. The authors demonstrated full key recovery on several McEliece challenge instances and on a key from the Courtois–Finiasz–Sendrier (CFS) signature scheme, and conjectured that the attack has subexponential complexity in the relevant security parameter. This complexity claim is heuristic, however, and the reported results do not provide a practical break of the selected Classic McEliece parameter sets [16].

The paper reports full key recovery on TII challenge instances advertised at generic-decoding security levels ranging from 83 to 248 bits, as well as on a key from the CFS signature scheme. These labels estimate the cost of generic decoding and therefore cannot be compared directly with the cost of the structural algebraic attack. The feasibility of the algebraic system depends on the underlying code parameters, and its extension to the selected Classic McEliece parameter sets still rests on a heuristic complexity conjecture. This is a significant structural result, but it does not provide a practical break of Classic McEliece [16].

Implications for Classic McEliece. These results rule out the stronger claim that binary Goppa codes are indistinguishable from random codes for every parameter choice. Algebraic distinguishers now apply to several Goppa and alternant-code regimes. The security question, however, is not distinguishability alone, but whether the detected structure can be converted into reliable key recovery at a cost below both generic decoding and the intended security level.

Current structural attacks are effective against particular high-rate, low-degree, or specially chosen instances, but they do not presently give a practical attack on the selected Classic McEliece parameter sets [16, 27]. These attacks open a serious cryptanalytic direction, but they do not yet break the selected Classic McEliece parameter sets.

6.4 Structured Variants: Smaller Keys and New Weaknesses

The large public key has always been the main practical limitation of McEliece. This motivated many variants that replaced the full public matrix with a more compact representation based on cyclic blocks, sparse matrices, or specially chosen code families [24, 25].

The attraction of these designs is straightforward. If a matrix consists of repeating or circulant blocks, only a small number of rows must be stored. Similar savings are possible when the secret code belongs to a family with a short algebraic description. The same restriction, however, also gives an attacker more information: instead of searching over a broad family of codes, the attacker knows that the secret satisfies particular symmetries or algebraic equations.

Quasi-cyclic and quasi-dyadic variants. Quasi-cyclic and quasi-dyadic constructions were proposed to reduce key sizes while retaining efficient decoding. In these systems, the public matrix can be reconstructed from a small number of blocks because its coordinates are related by known permutations [5, 61].

Several specific proposals were later broken by attacks that exploited exactly these relations. Otmani, Tillich, and Dallot recovered secret information from two quasi-cyclic McEliece variants, while Faugère, Otmani, Perret, and Tillich used algebraic equations to reconstruct equivalent decoders for compact quasi-cyclic and quasi-dyadic constructions [39, 70]. The imposed symmetry reduced not only the public-key size, but also the number of unknown quantities in the attacker's algebraic system.

These results do not imply that every quasi-cyclic code-based scheme is insecure. They show instead that security cannot be inferred from the hardness of generic decoding alone. The additional symmetry used for compression must itself be shown to resist structural recovery.

Sparse and moderate-density constructions. Another approach uses sparse parity-check matrices, often together with quasi-cyclic structure. Such matrices support compact storage and efficient decoding, but their low-weight rows may become targets for dual-code or structural attacks. The quasi-cyclic LDPC construction studied by Otmani, Tillich, and Dallot illustrates how cyclic symmetry and sparsity can jointly expose information about the secret parity-check matrix [70].

Later MDPC-based variants increased the density of the secret matrix to make direct recovery of low-weight dual codewords more difficult [62]. This changes the security assumption rather than eliminating it: the security of an MDPC construction must be assessed using the particular density, symmetry, and decoding behavior of that code family, rather than inherited automatically from the binary Goppa construction.

Restricted Goppa-code families. Some compact proposals remained within the Goppa-code family but restricted the defining polynomial, extension field, or automorphism group. These

restrictions can create algebraic identities that are not available for general binary Goppa codes. For example, Couvreur, Otmani, and Tillich developed a polynomial-time structural attack against Wild McEliece instantiated with Goppa codes over quadratic extensions [33].

These attacks have a limited scope: they apply only to specific Goppa subfamilies and parameter ranges. Their success in those settings does not extend automatically to binary Goppa codes in general, and they do not practically break the selected Classic McEliece parameter sets [27].

The lesson for Classic McEliece. Classic McEliece retains the traditional binary Goppa construction without imposing the quasi-cyclic, quasi-dyadic, or sparse key-compression structures used by many compact variants. Its large public keys are therefore not simply an overlooked implementation problem; they are partly the cost of avoiding additional public regularity whose security would require new assumptions [25].

Structure itself is not the problem, since binary Goppa codes rely on algebraic structure for efficient decoding. The risk comes from additional relations exposed through the public key, which may help an attacker recognize or reconstruct the secret code.

6.5 Chosen-Ciphertext and Protocol-Level Attacks

Generic decoding and structural cryptanalysis concern the mathematical problem underlying McEliece. Protocol-level attacks ask a different question: what can an adversary learn by modifying ciphertexts, repeating related messages, or observing how a receiver responds? The original McEliece transformation was not designed to provide modern security against adaptive chosen-ciphertext attacks [12, 45, 59].

Malleability and related messages. The linear form of the original encryption scheme makes its ciphertexts malleable. Given $c = mG + e$, an attacker can choose $\Delta \in \mathbb{F}_2^k$ and form

$$c' = c + \Delta G = (m + \Delta)G + e.$$

Thus, c' encrypts a known modification of the original plaintext. If an attacker can submit c' to a decryption service, the response may reveal m without solving the decoding problem or recovering the private key.

The basic transformation also requires careful handling of repeated or related messages. Berson showed that encrypting the same message more than once with different error vectors, or encrypting messages with a known linear relation, can reveal substantially more information than an isolated ciphertext [12]. These attacks do not make generic decoding easy; rather, they show that message formatting, randomness, and protocol use are part of the security argument.

Decryption oracles and reaction attacks. A chosen-ciphertext attack does not require the receiver to return a complete plaintext. Even a visible distinction between successful and failed decryption can act as an oracle. Hall, Goldberg, and Schneier showed that carefully modified McEliece ciphertexts, together with the receiver's reactions, could be used to recover information about a target plaintext [45].

The relevant reaction might be an explicit error message or a later change in protocol behavior. Timing, power, cache, and electromagnetic leakage are related concerns, but they belong to implementation security and are considered separately in the next subsection.

Conversions for stronger security. Several conversions were developed to protect McEliece against these protocol-level weaknesses. Kobara and Imai proposed modified McEliece constructions with semantic and adaptive chosen-ciphertext security in the random-oracle model [51]. More generally, the Fujisaki–Okamoto approach showed how hashing, ciphertext validation, and symmetric encryption can be combined with a weaker public-key primitive to obtain stronger chosen-ciphertext security [40].

The common idea is to avoid encrypting predictable application data directly. Instead, the public-key primitive protects a randomly generated value from which a session key is derived. The actual message is then protected by authenticated symmetric encryption, while decapsulation verifies that the ciphertext is consistent with the recovered value.

The Classic McEliece KEM. Classic McEliece follows this modern KEM design rather than using the original 1978 transformation for direct message encryption. It uses the Niederreiter formulation to encapsulate a randomly selected low-weight vector and derives a shared session key from that vector and the ciphertext [24].

During decapsulation, the ciphertext is decoded and checked for consistency. At a high level, the returned key has the form

$$K = \begin{cases} \mathcal{H}(1, e', C), & \text{if decoding and validation succeed,} \\ \mathcal{H}(0, s_{\mathrm{fb}}, C), & \text{otherwise,} \end{cases}$$

where e' is the recovered error vector and s_{fb} is a secret fallback value. Returning a pseudorandom fallback key rather than an explicit failure is known as *implicit rejection*. The domain-separation values 0 and 1 distinguish the valid and invalid cases [24, 27].

Implicit rejection prevents the abstract KEM from exposing a simple valid-versus-invalid decoding oracle. It does not, however, guarantee that an implementation leaks nothing about which path was taken. Constant-time execution and protection against physical leakage remain necessary.

The transition from direct McEliece encryption to a validated KEM is therefore more than a change in notation. It reflects the broader lesson that decoding hardness alone is not enough: security also depends on ciphertext validation, failure handling, randomness generation, and safe composition with symmetric encryption.

6.6 Implementation and Side-Channel Attacks

The attacks discussed so far treat encryption and decapsulation as ideal mathematical operations. Real implementations may additionally reveal information through execution time, memory access, power consumption, electromagnetic emissions, or induced faults. Such leakage can expose a private key even when generic decoding and structural key recovery remain computationally infeasible [2, 44].

This issue is especially relevant during Classic McEliece decapsulation, which performs several computations involving secret data, including syndrome decoding, evaluation of the error-locator polynomial, and application of a secret permutation. Security therefore depends not only on the mathematical algorithm, but also on whether its physical execution reveals information about those secret values.

Timing and microarchitectural leakage. Timing and cache attacks exploit differences in control flow or memory access that depend on private data. The official Classic McEliece software is designed to avoid secret-dependent branches and memory addresses, and compiled implementations can be tested for such data flow [29].

Constant-time programming is an important defense, but it does not prevent every form of leakage. A computation with fixed control flow may still produce data-dependent power consumption or electromagnetic emissions. Moreover, compiler optimizations and processor behavior can invalidate assumptions made at the source-code level. Constant-time claims must therefore be checked on the compiled implementation and on the intended platform.

Power and electromagnetic attacks. Guo, Johansson, and Johansson demonstrated a practical key-recovery side-channel attack against Classic McEliece implementations. Their attack submits specially chosen ciphertexts whose decoding involves a single entry of the secret permutation. Power leakage from the additive FFT used to evaluate the error-locator polynomial is then classified to recover that entry. Repeating the procedure reveals enough private structure for full key recovery [44].

The attack was demonstrated against an FPGA implementation and software running on an ARM Cortex-M4. It does not weaken the hardness of decoding binary Goppa codes. Instead, it shows that carefully selected ciphertexts can simplify an internal computation until its physical leakage becomes closely related to a small part of the private key.

Earlier and subsequent work has identified other leakage targets in McEliece and Niederreiter implementations, including the secret permutation, Goppa polynomial, syndrome computation, and decoding procedure [2, 44]. These results also connect chosen-ciphertext and side-channel attacks: the adversary controls the ciphertext mathematically and then observes the physical behavior caused by its decapsulation.

Fault-injection attacks. Fault attacks actively disturb a computation rather than merely observing it. An attacker may use clock or voltage glitches, electromagnetic or laser injection, or software instrumentation to corrupt an intermediate value or skip a security-critical operation.

Cayrel et al. demonstrated a message-recovery laser fault attack against a Classic McEliece implementation by corrupting syndrome computation [22]. Pircher et al. later presented a key-recovery fault attack targeting the error-locator polynomial and the ciphertext-validity checks. The resulting faulty outputs provide algebraic information from which an alternative secret key can be reconstructed [74].

These attacks are particularly relevant to implicit rejection. The abstract KEM returns a fallback key when decoding or validation fails, avoiding an explicit valid-versus-invalid response. A fault that changes the recovered vector, bypasses validation, or alters the choice between the valid and fallback paths may undermine this protection.

Countermeasures and security implications. Different threats require different defenses. Constant-time programming reduces timing and cache leakage, while masking and randomization aim to separate physical observations from sensitive intermediate values. Fault resistance may require redundant computations, consistency checks, or verification that every validation step was executed correctly [2, 74].

These protections introduce costs in execution time, memory, hardware area, and implementation complexity. No single countermeasure is sufficient for every environment: protection appropriate for a remote software attacker may be inadequate when an adversary has physical access to power traces or fault-injection equipment.

Implementation attacks do not contradict the mathematical security of McEliece. Instead, they show that a difficult decoding problem is only one part of a secure deployed system. Classic McEliece implementations must protect the entire decapsulation path, including decoding, validation, implicit rejection, and key derivation, under a threat model appropriate to the device and its expected use.

6.7 Quantum Cryptanalysis

McEliece is regarded as a post-quantum cryptosystem because no known quantum algorithm solves its underlying decoding or structural key-recovery problems in polynomial time. This differs from RSA and discrete-logarithm-based systems, whose foundations are directly undermined by Shor's algorithm. Known quantum attacks on McEliece mainly accelerate existing search and decoding methods rather than replacing them with a fundamentally different efficient attack [7, 49, 85].

Quantum information-set decoding. The simplest quantum improvement applies Grover's search algorithm to the repeated trials in information-set decoding. In Prange's method, an attacker tests coordinate sets until finding one with the required error distribution. Grover's algorithm reduces an unstructured search over N possibilities to approximately $\sqrt{N}$ quantum iterations. Bernstein used this idea to obtain a quantum version of Prange's attack [7, 43].

More advanced ISD algorithms contain collision searches, list merging, and nearest-neighbor procedures. Kachigar and Tillich used quantum walks to analyze quantum versions of several such methods, while Kirshanova developed a broader framework and studied the use of quantum nearest-neighbor techniques [49, 50].

These algorithms reduce the asymptotic cost of generic decoding, but they do not make it polynomial-time. Their practical cost also depends on quantum memory, circuit depth, data access, and other resources that are not fully reflected by an asymptotic exponent. Classic McEliece parameters are therefore selected using both classical and quantum ISD estimates [27].

Quantum Fourier sampling and hidden structure. A different possible attack arises from the code-equivalence problem, which can be formulated as a non-abelian hidden subgroup problem. This raises the question of whether quantum Fourier sampling could recover the permutation or algebraic structure concealed in a McEliece-type public key.

Dinh, Moore, and Russell showed that, under the scrambling and automorphism conditions considered in their analysis, McEliece and Niederreiter systems based on Goppa codes resist strong Fourier-sampling attacks and, more generally, measurements of individual coset states [34]. This rules out a natural family of Shor-like structural attacks, but it does not prove that every possible future quantum key-recovery method must fail.

Current interpretation. Quantum computers reduce the estimated security margin of McEliece, mainly by accelerating generic decoding and search. They do not currently overturn the scheme's basic security argument, and no known quantum attack provides a practical break of the selected Classic McEliece parameter sets [27].

Classic McEliece is better described as resistant to known quantum attacks than as quantum-proof. Future advances may improve quantum decoding or discover new ways to exploit Goppa-code structure, so quantum attack estimates must continue to be revisited as algorithms and cost models develop.

6.8 Lessons from Five Decades of Cryptanalysis

The cryptanalytic history of McEliece is more complicated than the claim that the scheme has remained unchanged and unbroken since 1978. Its original parameters became obsolete, several compact variants were broken, the basic encryption interface required stronger protocol protection, and practical implementations were shown to leak secret information. What endured was the binary

Goppa-code foundation, together with the ability to update parameters and security mechanisms as attacks improved.

Lesson 1: Public structure creates an additional assumption. Many variants reduced public-key size by introducing cyclic symmetry, sparse representations, or more narrowly defined code families. These changes improved efficiency, but sometimes gave attackers algebraic relations that made the hidden code easier to recognize or reconstruct [33, 39, 70].

Structure itself is not the problem. McEliece relies on the algebraic structure of binary Goppa codes for efficient decoding. The risk arises when additional regularity is exposed through the public key, since each such relation introduces a separate security assumption.

Lesson 2: A construction may survive while its parameters do not. Improved information-set decoding eventually made the parameters proposed in 1978 inadequate by modern standards [11, 17]. This was not a general structural recovery of binary Goppa codes. It showed instead that a long-lived construction must support larger parameters as algorithms, hardware, and cost models improve.

Lesson 3: The scope of an attack must be stated precisely. A distinguisher is not automatically a key-recovery attack, and an attack on a specially selected challenge instance need not threaten the selected Classic McEliece parameters. Likewise, breaking a quasi-cyclic, quasi-dyadic, or GRS-based variant does not by itself break the traditional binary Goppa construction.

Claims such as "McEliece has been broken" or "McEliece has never been broken" are therefore too broad. A meaningful assessment should specify the code family, parameter range, attack model, computational cost, and whether the result is a distinguisher, message-recovery attack, or full key recovery [16, 27, 36].

Lesson 4: Mathematical hardness is only one layer of security. Resistance to decoding and structural recovery does not guarantee that a deployed system is secure. The original encryption transformation is malleable, decryption behavior can create an oracle, and implementations may leak through timing, power, electromagnetic emissions, memory access, or induced faults [12, 44, 45, 74].

Classic McEliece addresses protocol-level concerns through its KEM design, ciphertext validation, and implicit rejection. These mechanisms still require careful, constant-time and fault-resistant implementation [24].

Lesson 5: Long analysis provides confidence, not certainty. Nearly five decades of public scrutiny have produced a detailed understanding of generic decoding costs, structural weaknesses, protocol requirements, and implementation risks. This history is an important reason for continued confidence in Classic McEliece, but it is not a proof that no stronger attack exists.

Recent algebraic work illustrates this point. New distinguishers and key-recovery techniques have succeeded against selected Goppa-code instances without yet providing a practical break of the selected Classic McEliece parameter sets [16]. The evidence supports continued confidence, not a claim of permanent security.

These lessons help explain why Classic McEliece accepts large public keys rather than relying on aggressive structural compression. This conservative choice preserves a security foundation whose limitations are comparatively well understood. The scheme is valuable not only because of the attacks it has resisted, but also because its trade-offs and remaining assumptions can be assessed relatively clearly.

7 Open Challenges and Future Directions

The long history of McEliece does not mean that its main research questions have been resolved. In fact, several of its strongest design choices also create its most difficult practical problems. Avoiding additional public algebraic structure supports confidence in the scheme, but leads to very large public keys. Its decoding foundation has survived decades of analysis, but recent algebraic results show that the visible structure of Goppa codes is not yet fully understood. Secure deployment also depends on implementations that protect the entire decapsulation process from leakage and faults.

Therefore, the central research question is how to preserve the conservative security advantages of Classic McEliece while reducing the costs that limit its practical use.

Reducing key size without exposing structure. Across the five Round-4 parameter families, the base and `f` variants have public keys ranging from 261,120 to 1,357,824 bytes and ciphertexts ranging from 96 to 208 bytes. For the ISO-standardized `pc` and `pcf` variants, ciphertexts are 32 bytes longer [29, 30].

Large public keys are more manageable when they can be distributed once and reused across multiple encapsulations, although deployment analyses should also account for multi-instance security and an appropriate key-rotation policy [58]. They remain difficult to accommodate in constrained devices, certificates, and protocols that transmit public keys frequently.

The history of compact variants shows why reducing these key sizes is difficult. Quasi-cyclic symmetry, sparse matrices, and restricted algebraic code families have provided substantial compression, but in several constructions the same regularity enabled structural recovery [33, 39, 70].

Future proposals must therefore be evaluated against both generic decoding and structural key recovery. Possible directions include safer public representations, better distribution and caching of long-lived keys, and compact code families supported by strong evidence against structural attacks. A reduction in key size is worthwhile only if the information enabling compression does not also reveal an equivalent decoder.

Clarifying structural and decoding security. Recent work on square codes, quadratic relations, low-rank matrices, and algebraic distinguishers has identified detectable properties of alternant and Goppa codes in parameter ranges that were previously difficult to analyze [16, 32, 63, 64, 76].

The main unresolved question is whether such distinguishers can be converted into reliable key-recovery attacks that scale to the selected Classic McEliece parameters and outperform generic decoding. The heuristic subexponential approach proposed in 2026 is important in this respect, but its complexity analysis depends on assumptions about the behavior of the algebraic solver, and the demonstrated attacks do not constitute a practical break of the selected Classic McEliece parameter sets [16].

The next step is to tighten the complexity estimates, test the heuristics independently, determine where the algebraic systems become tractable, and clarify whether the observed invariants remain useful at the code rates and Goppa degrees used in cryptographic parameter sets.

Information-set decoding remains the baseline against which these structural attacks are measured. Classical and quantum estimates must continue to be revised as list-merging methods, nearest-neighbor techniques, parallel implementations, and hardware improve [7, 27, 49, 50]. Quantum analyses should report realistic resource requirements, including circuit depth, memory, data access, and error-correction costs, rather than relying only on asymptotic query complexity.

Secure and efficient implementation. Implementation security will remain a major area of McEliece research. Constant-time software reduces timing and cache leakage, but does not by

itself prevent power analysis, electromagnetic observation, or fault injection. Practical attacks have already shown that chosen ciphertexts can simplify internal decoding computations and expose information about the private key [22, 44, 74].

Further work is needed on masked decoding, randomized arithmetic, fault-resistant validation, and protections covering the complete decapsulation path, including implicit rejection and key derivation. Countermeasures should also be evaluated together: a defense against one leakage source may introduce new intermediate values or control flow that becomes vulnerable elsewhere.

Efficiency is part of this problem. Masking, redundancy, and fault detection increase execution time, memory use, hardware area, and implementation complexity. A more useful goal is to build and evaluate implementations for specific threat models, including servers, embedded devices, and hardware accelerators.

Deployment and protocol integration. Recent standardization decisions have clarified the position of Classic McEliece without resolving every deployment question. NIST selected HQC for standardization in 2025 and did not select Classic McEliece, citing its large public keys and uncertainty about sufficiently widespread adoption while maintaining confidence in its security [1]. Classic McEliece was subsequently included in ISO/IEC 18033-2:2006/Amd 2:2026 [47].

Further experimental work should identify the settings in which the scheme's small ciphertexts and long security history justify its initial key-distribution cost. Relevant questions include its effect on certificate chains, network handshakes, constrained links, long-term storage systems, and applications using static or infrequently replaced public keys.

Hybrid deployment is another important direction. Classic McEliece can be combined with classical or other post-quantum mechanisms during the transition to post-quantum cryptography, but secure composition requires careful key derivation, downgrade protection, failure handling, and multi-user analysis. Simply combining two KEM outputs does not by itself establish the desired security guarantee.

Formal analysis should also be connected more closely to implementation and protocol behavior. Long-lived keys, repeated decapsulation, physical leakage, induced faults, and partially protected implementations may create risks not captured by an idealized black-box KEM model [24, 27].

Research outlook. Classic McEliece need not replace every public-key system to remain valuable. Its strongest role may be in settings where conservative security, small ciphertexts, perfect correctness, and long-term confidence are more important than compact public keys.

Future work must preserve the design choices that have contributed to McEliece's long-term security while continuing to test its structural assumptions, security estimates, implementations, and use in protocols. Improvements should be introduced cautiously, but McEliece should not be treated as a closed research problem.

8 Conclusion

Nearly five decades after its introduction, the McEliece cryptosystem continues to occupy a distinctive position in post-quantum cryptography. Its original parameters are no longer secure, several modified and compact variants have been broken, and its large public keys remain a serious obstacle to widespread deployment. Nevertheless, the conservative binary Goppa-code foundation retained in Classic McEliece has not been practically broken at the selected Classic McEliece parameter sets.

The history reviewed in this survey shows why statements such as "McEliece has been broken" or "McEliece has never been broken" are both incomplete. Improvements in information-set decoding

made the 1978 parameters obsolete without recovering the hidden Goppa structure. Structural attacks exposed weaknesses in alternative code families and compressed variants, but did not automatically extend to the traditional Classic McEliece construction. Protocol-level, side-channel, and fault attacks further showed that decoding hardness alone is not enough for secure deployment. Modern Classic McEliece therefore combines updated parameters and a Niederreiter-based KEM with ciphertext validation, key derivation, and implicit rejection, while preserving the central coding-theoretic design.

McEliece's main strength is not an absence of cryptanalytic progress, but the depth and variety of the analysis it has received. Its security has been examined through generic decoding, structural recovery, protocol attacks, implementation leakage, and quantum acceleration. This long record provides substantial confidence, although it cannot guarantee that stronger attacks will never be found. Recent algebraic results reinforce the need to distinguish carefully between detection of hidden structure, attacks on particular parameter regimes, and practical key recovery against the selected Classic McEliece parameter sets.

The future role of Classic McEliece will probably remain specialized rather than universal. Its very large public keys make it unattractive in settings where keys must be transmitted frequently, but its small ciphertexts, perfect correctness, and conservative security history may be valuable when public keys can be distributed infrequently and reused for many encapsulations. Continued work on structural cryptanalysis, realistic classical and quantum security estimates, protected implementations, and protocol integration will determine whether this role can expand.

McEliece also shows how a cryptosystem can change substantially without giving up its core mathematical foundation. Its parameters, interfaces, security models, and implementation requirements have evolved over time, but binary Goppa codes have remained central to the construction. The scheme remains important not only because of the security it has retained, but also because its failures, compromises, and revisions offer useful guidance for future code-based designs.

References

[1] Gorjan Alagic, Maxime Bros, Pierre Ciadoux, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Hamilton Silberg, Daniel Smith-Tone, and Noah Waller. Status report on the fourth round of the NIST post-quantum cryptography standardization process. NIST Internal Report NIST IR 8545, National Institute of Standards and Technology, Gaithersburg, Maryland, March 2025.

[2] Roberto Avanzi, Simon Hoerder, Dan Page, and Michael Tunstall. Side-channel attacks on the McEliece and Niederreiter public-key cryptosystems. *Journal of Cryptographic Engineering*, 1(4):271–281, 2011.

[3] Magali Bardet, Rocco Mora, and Jean-Pierre Tillich. Polynomial time key-recovery attack on high rate random alternant codes. *IEEE Transactions on Information Theory*, 70(6):4492–4511, 2024.

[4] 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*, volume 7237 of *Lecture Notes in Computer Science*, pages 520–536. Springer, 2012.

[5] Thierry P. Berger, Pierre-Louis Cayrel, Philippe Gaborit, and Ayoub Otmani. Reducing key length of the McEliece cryptosystem. In *Progress in Cryptology – AFRICACRYPT 2009*, volume 5580 of *Lecture Notes in Computer Science*, pages 77–97. Springer, 2009.

[6] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[7] Daniel J. Bernstein. Grover vs. McEliece. In Nicolas Sendrier, editor, *Post-Quantum Cryptography – PQCrypto 2010*, volume 6061 of *Lecture Notes in Computer Science*, pages 73–80, Berlin, Heidelberg, 2010. Springer.

[8] Daniel J. Bernstein. McEliece standardization. Personal blog post, `https://blog.cr.yp.to/20250423-mceliece.html`, April 2025. Published 23 April 2025; accessed 23 July 2026.

[9] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors. *Post-Quantum Cryptography*. Springer, Berlin, Heidelberg, 2009.

[10] Daniel J. Bernstein and Tung Chou. CryptAttackTester: High-assurance attack analysis. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024*, volume 14925 of *Lecture Notes in Computer Science*, pages 141–182, Cham, 2024. Springer.

[11] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the McEliece cryptosystem. In Johannes Buchmann and Jintai Ding, editors, *Post-Quantum Cryptography*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46, Berlin, Heidelberg, 2008. Springer.

[12] Thomas A. Berson. Failure of the McEliece public-key cryptosystem under message-resend and related-message attack. In Burton S. Kaliski, editor, *Advances in Cryptology – CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 213–220. Springer, 1997.

[13] Ward Beullens. Breaking Rainbow takes a weekend on a laptop. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, volume 13508 of *Lecture Notes in Computer Science*, pages 464–479, Cham, 2022. Springer. Proceedings, Part II.

[14] Sebastian Bitzer, Jeroen Delvaux, Elena Kirshanova, Sebastian Maaßen, Alexander May, and Antonia Wachter-Zeh. How to lose some weight: A practical template syndrome decoding attack. *Designs, Codes and Cryptography*, 93(7):2503–2519, 2025.

[15] Charles Bouillaguet, Claire Delaplace, and Mickaël Hamdad. The May–Ozerov algorithm for syndrome decoding is "galactic". *IACR Communications in Cryptology*, 2(1), 2025.

[16] Pierre Briaud, Axel Lemoine, Hugues Randriambololona, and Jean-Pierre Tillich. A heuristic subexponential attack on the McEliece cryptosystem. Cryptology ePrint Archive, Paper 2026/1232, 2026.

[17] Anne Canteaut and Nicolas Sendrier. Cryptanalysis of the original McEliece cryptosystem. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology – ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 187–199, Berlin, Heidelberg, 1998. Springer.

[18] Rodolfo Canto Torres and Nicolas Sendrier. Analysis of information set decoding for a sublinear error weight. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography – PQCrypto 2016*, volume 9606 of *Lecture Notes in Computer Science*, pages 144–161, Cham, 2016. Springer.

[19] Kévin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. Statistical Decoding 2.0: Reducing decoding to LPN. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022*, volume 13794 of *Lecture Notes in Computer Science*, pages 477–507, Cham, 2022. Springer. Proceedings, Part IV.

[20] Kévin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. Reduction from sparse LPN to LPN, Dual Attack 3.0. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024*, volume 14657 of *Lecture Notes in Computer Science*, pages 286–315, Cham, 2024. Springer. Proceedings, Part VII.

[21] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, volume 14008 of *Lecture Notes in Computer Science*, pages 423–447, Cham, 2023. Springer.

[22] Pierre-Louis Cayrel, Brice Colombier, Vlad-Florin Drăgoi, Alexandre Menu, and Lilian Bossuet. Message-recovery laser fault injection attack on the Classic McEliece cryptosystem. In *Advances in Cryptology – EUROCRYPT 2021*, volume 12697 of *Lecture Notes in Computer Science*, pages 438–467. Springer, 2021.

[23] Pierre-Louis Cayrel, Sidi Mohamed El Yousfi Alaoui, Gerhard Hoffmann, Mohammed Meziani, and Robert Niebuhr. Recent progress in code-based cryptography. In Tai-hoon Kim, Hojjat Adeli, Rosslin John Robles, and Maricel Balitanas, editors, *Information Security and Assurance*, volume 200 of *Communications in Computer and Information Science*, pages 21–32, Berlin, Heidelberg, 2011. Springer.

[24] Classic McEliece Team. Classic McEliece: Conservative code-based cryptography—cryptosystem specification. Submission document for the NIST Post-Quantum Cryptography Standardization Project, October 2022.

[25] Classic McEliece Team. Classic McEliece: Conservative code-based cryptography—design rationale. Submission document for the NIST Post-Quantum Cryptography Standardization Project, October 2022.

[26] Classic McEliece Team. Classic McEliece: Conservative code-based cryptography—guide for implementors. Submission document for the NIST Post-Quantum Cryptography Standardization Project, October 2022.

[27] Classic McEliece Team. Classic McEliece: Conservative code-based cryptography—guide for security reviewers. Submission document for the NIST Post-Quantum Cryptography Standardization Project, October 2022.

[28] Classic McEliece Team. Classic McEliece: Conservative code-based cryptography—round 4 submission overview. Submission to the NIST Post-Quantum Cryptography Standardization Project, October 2022.

[29] Classic McEliece Team. Classic McEliece: Implementation. `https://classic.mceliece.org/impl.html`, 2026. Accessed: 2026-07-18.

[30] Classic McEliece Team. Classic McEliece: ISO. `https://classic.mceliece.org/iso.html`, June 2026. Version 2026.06.15. Accessed: 2026-07-21.

[31] Alain Couvreur, Philippe Gaborit, Valérie Gauthier-Umaña, Ayoub Otmani, and Jean-Pierre Tillich. Distinguisher-based attacks on public-key cryptosystems using Reed–Solomon codes. *Designs, Codes and Cryptography*, 73(2):641–666, 2014.

[32] Alain Couvreur, Rocco Mora, and Jean-Pierre Tillich. A new approach based on quadratic forms to attack the McEliece cryptosystem. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023*, volume 14441 of *Lecture Notes in Computer Science*, pages 3–38, Singapore, 2023. Springer.

[33] Alain Couvreur, Ayoub Otmani, and Jean-Pierre Tillich. Polynomial-time attack on Wild McEliece over quadratic extensions. *IEEE Transactions on Information Theory*, 63(1):404–427, 2017.

[34] Hang Dinh, Cristopher Moore, and Alexander Russell. McEliece and Niederreiter cryptosystems that resist quantum Fourier sampling attacks. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 761–779, Berlin, Heidelberg, 2011. Springer.

[35] Ilya Dumer. On minimum distance decoding of linear codes. In *Proceedings of the 5th Joint Soviet–Swedish International Workshop on Information Theory: Convolutional Codes and Multi-User Communication*, pages 50–52, Moscow, USSR, 1991.

[36] Daniela Engelbert, Raphael Overbeck, and Arthur Schmidt. A summary of McEliece-type cryptosystems and their security. *Journal of Mathematical Cryptology*, 1(2):151–199, 2007.

[37] André Esser and Emanuele Bellini. Syndrome decoding estimator. In *Public-Key Cryptography – PKC 2022*, volume 13177 of *Lecture Notes in Computer Science*, pages 112–141. Springer, 2022.

[38] André Esser, Alexander May, and Floyd Zweydinger. McEliece needs a break—solving McEliece-1284 and quasi-cyclic-2918 with modern ISD. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022*, volume 13277 of *Lecture Notes in Computer Science*, pages 433–457, Cham, 2022. Springer.

[39] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. In *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 279–298. Springer, 2010.

[40] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.

[41] Peizhou Gan, Prasanna Ravi, Kamal Raj, Anubhab Baksi, and Anupam Chattopadhyay. Classic McEliece hardware implementation with enhanced side-channel and fault resistance. Cryptology ePrint Archive, Paper 2024/1828, 2024.

[42] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 169–178. Association for Computing Machinery, 2009.

[43] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 212–219, New York, 1996. Association for Computing Machinery.

[44] Qian Guo, Andreas Johansson, and Thomas Johansson. A key-recovery side-channel attack on Classic McEliece implementations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):800–827, 2022.

[45] Chris Hall, Ian Goldberg, and Bruce Schneier. Reaction attacks against several public-key cryptosystems. In *Information and Communications Security*, volume 1726 of *Lecture Notes in Computer Science*, pages 2–12. Springer, 1999.

[46] Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 235–256, Berlin, Heidelberg, 2010. Springer.

[47] International Organization for Standardization and International Electrotechnical Commission. ISO/IEC 18033-2:2006/Amd 2:2026: Information technology—security techniques— encryption algorithms—part 2: Asymmetric ciphers— amendment 2. International Standard, June 2026. 58 pages.

[48] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography – PQCrypto 2011*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34, Berlin, Heidelberg, 2011. Springer.

[49] Ghazal Kachigar and Jean-Pierre Tillich. Quantum information set decoding algorithms. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography – PQCrypto 2017*, volume 10346 of *Lecture Notes in Computer Science*, pages 69–89, Cham, 2017. Springer.

[50] Elena Kirshanova. Improved quantum information set decoding. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography – PQCrypto 2018*, volume 10786 of *Lecture Notes in Computer Science*, pages 507–527, Cham, 2018. Springer.

[51] Kazukuni Kobara and Hideki Imai. Semantically secure McEliece public-key cryptosystems: Conversions for McEliece PKC. In *Public Key Cryptography – PKC 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2001.

[52] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[53] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International Computer Science Laboratory, Menlo Park, California, October 1979.

[54] 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*, volume 330 of *Lecture Notes in Computer Science*, pages 275–280. Springer, 1988.

[55] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In Christoph G. Günther, editor, *Advances in Cryptology – EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 419–453, Berlin, Heidelberg, 1988. Springer.

[56] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{O}(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology –*

ASIACRYPT 2011, volume 7073 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2011.

[57] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 203–228. Springer, 2015.

[58] Alexander May and Gabriel Sá Diogo. Multi-instance security degradation of code-based KEMs. Cryptology ePrint Archive, Paper 2026/517, 2026.

[59] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. DSN Progress Report 42–44, Jet Propulsion Laboratory, 1978.

[60] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89 Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238, New York, 1990. Springer.

[61] Rafael Misoczki and Paulo S. L. M. Barreto. Compact McEliece keys from Goppa codes. In *Selected Areas in Cryptography – SAC 2009*, volume 5867 of *Lecture Notes in Computer Science*, pages 376–392. Springer, 2009.

[62] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *2013 IEEE International Symposium on Information Theory*, pages 2069–2073. IEEE, 2013.

[63] Rocco Mora. On the matrix code of quadratic relationships for a Goppa code. *Advances in Mathematics of Communications*, 19(3):829–852, 2025. First published online in 2024.

[64] Rocco Mora and Jean-Pierre Tillich. On the dimension and structure of the square of the dual of a Goppa code. *Designs, Codes and Cryptography*, 91(4):1351–1372, 2023.

[65] National Institute of Standards and Technology. Digital signature standard (DSS). Federal Information Processing Standards Publication FIPS 186-5, National Institute of Standards and Technology, 2023.

[66] National Institute of Standards and Technology. Module-lattice-based digital signature standard. Federal Information Processing Standards Publication FIPS 204, National Institute of Standards and Technology, August 2024.

[67] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard. Federal Information Processing Standards Publication FIPS 203, National Institute of Standards and Technology, August 2024.

[68] National Institute of Standards and Technology. Stateless hash-based digital signature standard. Federal Information Processing Standards Publication FIPS 205, National Institute of Standards and Technology, August 2024.

[69] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.

[70] Ayoub Otmani, Jean-Pierre Tillich, and Léonard Dallot. Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. *Mathematics in Computer Science*, 3(2):129–140, 2010.

[71] Raphael Overbeck and Nicolas Sendrier. Code-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 95–145. Springer, Berlin, Heidelberg, 2009.

[72] Nicholas J. Patterson. The algebraic decoding of Goppa codes. *IEEE Transactions on Information Theory*, 21(2):203–207, 1975.

[73] Chris Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4):283–424, 2016.

[74] Sabine Pircher, Johannes Geier, Julian Danner, Daniel Mueller-Gritschneder, and Antonia Wachter-Zeh. Key-recovery fault injection attack on the Classic McEliece KEM. In Jean-Christophe Deneuville, editor, *Code-Based Cryptography*, volume 13839 of *Lecture Notes in Computer Science*, pages 37–61. Springer, 2023.

[75] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

[76] Hugues Randriambololona. The syzygy distinguisher. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025*, volume 15606 of *Lecture Notes in Computer Science*, pages 324–354, Cham, 2025. Springer.

[77] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 84–93. Association for Computing Machinery, 2005.

[78] Marek Repka and Pierre-Louis Cayrel. Cryptography based on error correcting codes: A survey. In Sattar B. Sadkhan Al Maliky and Nidaa A. Abbas, editors, *Multidisciplinary Perspectives in Cryptology and Information Security*, chapter 5, pages 133–156. IGI Global, Hershey, Pennsylvania, 2014.

[79] Marek Repka and Pavol Zajac. Overview of the McEliece cryptosystem and its security. *Tatra Mountains Mathematical Publications*, 60:57–83, 2014.

[80] Maryam Rezapour and Benjamin Fuller. ProxCode: Efficient proximity searchable encryption from error correcting codes. In Feng-Hao Liu, editor, *Topics in Cryptology – CT-RSAC 2026*, volume 16496 of *Lecture Notes in Computer Science*, pages 32–62, Cham, 2026. Springer.

[81] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[82] Nicolas Sendrier. Finding the permutation between equivalent linear codes: The support splitting algorithm. *IEEE Transactions on Information Theory*, 46(4):1193–1203, July 2000.

[83] Nicolas Sendrier. Decoding one out of many. In Bo-Yin Yang, editor, *Post-Quantum Cryptography – PQCrypto 2011*, volume 7071 of *Lecture Notes in Computer Science*, pages 51–67, Berlin, Heidelberg, 2011. Springer.

[84] Nicolas Sendrier. Code-based cryptography: State of the art and perspectives. *IEEE Security & Privacy*, 15(4):44–50, 2017.

[85] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE, 1994.

[86] Vladimir M. Sidelnikov and Sergei O. Shestakov. On insecurity of cryptosystems based on generalized Reed–Solomon codes. *Discrete Mathematics and Applications*, 2(4):439–444, 1992.

[87] Harshdeep Singh. Code based cryptography: Classic McEliece. *arXiv preprint arXiv:1907.12754*, 2019. Version 2, May 2020.

[88] Jacques Stern. A method for finding codewords of small weight. In Gérard D. Cohen and Jacques Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1989.

[89] Violetta Weger, Niklas Gassner, and Joachim Rosenthal. A survey on code-based cryptography. arXiv preprint arXiv:2201.07119, 2022. Version 5, revised July 2024.