Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 747–752
DOI: 10.15439/2025F6805
ISSN 2300-5963 ACSIS, Vol. 43

DivCrypt: A Structured Framework for Validating Cryptographic Implementations

Artur Misztal
0000-0003-2402-937X
AGH University of Krakow, al. Mickiewicza 30, 30-059 Krakow, Poland
Łukasiewicz - AI, ul. Leopolda 31, 40-189 Katowice, Poland
Email: misztal@agh.edu.pl, artur.misztal@ai.lukasiewicz.gov.pl

***Abstract*—This paper introduces the DivCrypt framework, a step-by-step guidance for validation of cryptographic implementations. The core idea is to decompose the evaluated implementation into components across complexity layers and validate each component through a structured five-step process. DivCrypt is not intended to replace existing standards, but rather to serve as a practical audit playbook. It can be easily adopted by evaluators in formal certification processes, researchers conducting security audits, and developers performing internal testing. The framework is also intended to encourage creators of novel, non-standardized cryptographic constructs to publish and maintain DivCrypt-aligned knowledge base. Such contributions, including test vectors and known implementation vulnerabilities, would support not only the application of DivCrypt, but also benefit the broader research and development community.**

I. Introduction

MODERN services increasingly depend on complex, multi-component IT/OT systems, highlighting the critical need to enhance their robustness. To establish trust, the security of these systems should be validated through external, independent security audits, typically performed in accordance with family of standards such as ISO/IEC 15408 [1] or ISA/IEC 62443 [2]. Compliance with such norms may be enforced by law or required in procurement for the defense sector. Given their fundamental role, cryptographic mechanisms are integral to these certification processes, serving as essential trust anchors. Even minor implementation flaws can significantly undermine security of the system, emphasizing the need for consistent and rigorous validation. Consequently, there are standards for both developing [3], [4] and testing [5], [6] of the cryptographic modules.

However, existing norms do not cover the validation of non-standardized cryptographic mechanisms. Notably, widely adopted community or industry tested cryptographic solutions sometimes enable advanced features in modern applications, yet the formal standardization processes can take years[1] resulting in significant gaps in official guidance. Security auditors evaluating novel cryptographic schemes currently rely on limited informal guidelines [9]. As a result, a comprehensive, high-level framework for the consistent validation of cutting-edge cryptography implementations is currently lacking.

[1]For example, the NIST Post Quantum Cryptography competition selected new standards: ML-KEM [7] and ML-DSA [8], after eight years of evaluation.

To address this gap, we introduce a methodology consisting of five sequential steps, grounded in a core principle: decomposing cryptographic solutions into smaller, manageable components. As illustrated in Figure 1, each component undergoes comprehensive security evaluation across four dimensions: *theoretical*, *functional*, *computational*, and *contextual*. The advantages of our framework include:

- **Efficiency.** Defines high-level procedural steps and methodologies without prescribing specific implementation details. The decomposition of the evaluated implementation and reuse of knowledge bases make the framework efficient by avoiding redundant efforts;
- **Inclusivity.** Accessible to a diverse range of users including developers, security auditors, and evaluators - thanks to its simplicity and flexibility in tool selection;
- **Universality.** Applicable to both standardized and emerging cryptographic protocols, including homomorphic encryption and non-interactive zero-knowledge proofs;
- **Estimability.** Once an implementation is decomposed, the total time required for evaluation becomes easy to estimate.

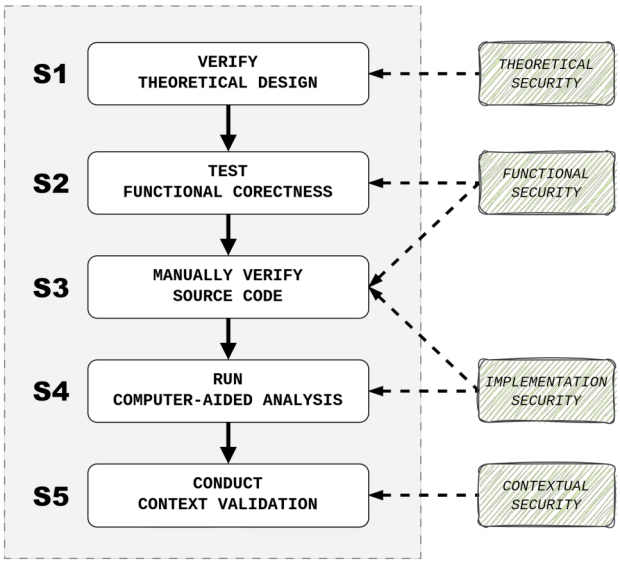

Fig. 1. The five sequential steps of DivCrypt comprehensively address four essential aspects of implementation security.

The absence of standards for novel cryptographic solutions poses significant challenges, especially under regulatory scrutiny of blockchain-based applications. In this domain, there is a growing adoption of zero-knowledge proofs, primarily to enhance privacy and scalability. Our framework is intended to bridge the gap between innovation and compliance, enabling consistent evaluation of cryptographic implementations across industry.

II. DivCrypt Framework

A. Terminology

The DivCrypt framework adopts a structured perspective on cryptographic implementations by organizing them into a hierarchy of four abstraction levels, referred to as *Cryptographic Layers* (CL):

CL1. cryptographic protocols;
CL2. cryptographic schemes;
CL3. cryptographic primitives;
CL4. underlying arithmetic.

We also introduce the notion of *Cryptographic Layer Object* (CLO) which denotes a specific instance at any given layer. For example, CLO can be: TLS v1.3 (CL1), Ed25519 (CL2), SHA-512 (CL3) or prime field (CL4). As a natural extension of CL and CLO, we define the notion of a *Cryptographic Layer Object Implementation* (CLOI), representing the actual implementation of a given CLO. The implementation under evaluation within the DivCrypt framework is referred to as the DivCrypt Target (DCT). DCT is subsequently decomposed into a directed acyclic graph, known as the DCT-graph, where the DCT itself serves as the root node with no parents. An illustrative example of such a decomposition is presented in Figure 2.

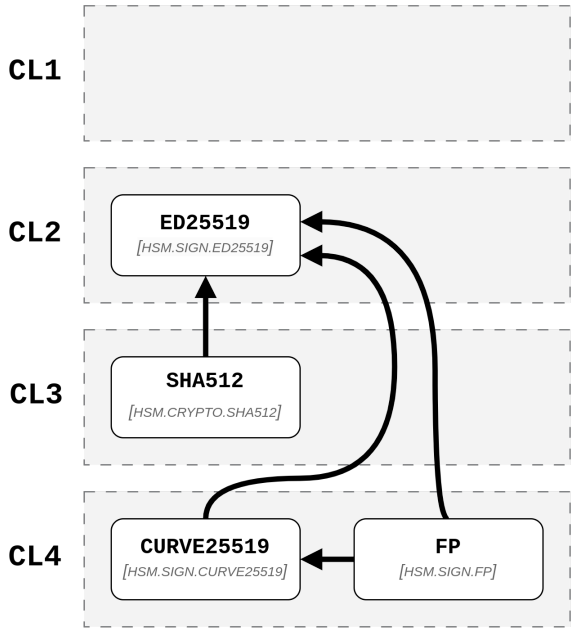

Fig. 2. The DCT-graph of the software implementation of Ed25519 digital signature scheme.

B. Testing procedure

The main idea behind DivCrypt is to conduct methodological analysis of cryptographic primitives, schemes, or protocols by applying divide-and-conquer-like approach, which materializes through the construction of the DCT-graph. The DivCrypt framework incorporates a range of validation techniques to comprehensively address all relevant security aspects of cryptographic algorithm implementations. It consists of three main parts:

1) ***Building the Knowledge Base****:* This is a continuous process of maintaining a database comprising two core elements described in the Section III: the *Cryptographic Validation Card* (CVC) and the *Cryptographic Vulnerability Reference* (CVR). This component is verification-independent and serves as a shared foundation for evaluations.

2) ***Decomposing into DCT-graph****:* By identifying each subcomponent according to its corresponding cryptographic layer, the DCT is decomposed into a directed acyclic graph with four levels of nodes, each visualizing a specific Cryptographic Layer (CL). The DCT-graph can be understood as a graph of dependencies, where edges represent implementation-level reliance. An example DCT-graph for Ed25519 [10] is shown in Figure 2. It is important to note that every node must be identified by its package name, as a DCT may contain multiple implementations of the same primitive (e.g., three different implementations of SHA-256). In Figure 2, package names are enclosed in square brackets. Each package name must be unique to ensure clarity and avoid ambiguity.

3) ***Validating DCT-graph****:* Validation is carried out through five sequential steps described in the Section IV, applied recursively from the leaf nodes to the root, ensuring that each node is verified before its parent. These steps, illustrated in Figure 1, define a strict sequence to follow, while allowing flexibility in how precisely each step is tailored to the context and assurance requirements of the evaluation.

III. Reusable Knowledge Base

During the decomposition of the DCT into the DCT-graph, it is necessary to ensure that every CLO is associated with a corresponding *Cryptographic Validation Card* (CVC) and *Cryptographic Vulnerability Reference* (CVR). The creation and maintenance of both artifacts are verification-independent tasks, performed prior to the validation phase. The purpose of this initial step is to establish and continuously update a knowledge base containing essential information about various CLOs. The CVC (Subsection III-A) captures the detailed specification and structure of CLO, incorporating both official and internally constructed test vectors. The CVR (Subsection III-B) enumerates known sources of security issues, including academic papers, vulnerability disclosures, and implementation-specific attack reports relevant to the examined CLO. The concept of utilizing a reusable knowledge

base to improve the efficiency and consistency of security evaluations has been applied in other contexts - for example, to streamline the complex documentation process for Common Criteria certification [11].

A. Cryptographic Validation Card

[CVC-1] Specifications. Reference to the official CLO specification, including relevant standards and technical documentation.

[CVC-2] Algorithms. A list of algorithms that collectively define the CLO.

[CVC-3] Dependencies. A list of other CLOs from lower cryptographic layers that are directly utilized by the CLO.

[CVC-4] Sources. Collection of useful materials, such as reference implementations and technical analyses.

[CVC-5] Official Test Procedures. References to official testing procedures developed by the authors of the cryptographic standard or proposal. Usually, this is a set of test vectors called Known Answer Tests (KATs).

[CVC-6] Other Test Procedures. Supplementary internal testing strategies aimed at verifying both correctness and robustness against failure scenarios. This can be achieved by designing test vectors that cover edge cases and applying them to both the reference and the evaluated implementation.

B. Cryptographic Vulnerability Reference

[CVR-1] Queries Table. A complete list of search queries used to collect entries for CVR-2, CVR-3, and CVR-4.

[CVR-2] CVE Entries. A list of relevant Common Vulnerabilities and Exposures (CVE) entries associated with the CLO.

[CVR-3] Academic Papers. A curated list of academic publications addressing implementation-specific aspects or vulnerabilities of the CLO.

[CVR-4] Media. A collection of publicly available sources, such as blog posts, technical write-ups, or incident reports, that refer to observed vulnerabilities or behaviors.

IV. DivCrypt's Evaluation Steps

S1. Verify theoretical design

Task. Evaluator shall verify that the DCT is listed in widely accepted standards.

Example. The evaluator confirmed that the Ed25519 signature scheme is included in NIST FIPS 186-5 [12].

Remarks. DivCrypt is not intended to analyze the security of the cryptographic protocols, schemes, or primitives, but only the implementation correctness of standardized ones.

S2. Test functional correctness

Task. Evaluator shall execute the test procedures specified in CVC-5 and CVC-6.

Example. The evaluator developed interfaces in Python for both the tested and reference implementations. A test script was run, invoking procedures described in CVC-5 and CVC-6. The results indicated all tests completed successfully.

Remarks. Testing with predefined or dynamically generated test vectors is the basic method of verifying implementation correctness. The reference implementation serves as a benchmark to validate the behavior of the evaluated cryptographic module.

S3. Source code verification

Task. Evaluator shall perform a comprehensive review of source files, focusing on critical functions and their usage, including input handling and side-channel resistance.

Example. The evaluator manually reviewed the codebase and, using all accessible CVC/CVR artifacts, concluded that the implementation conforms to current best practices and is appropriate for the intended application context.

Remarks. Although time-consuming, manual source code analysis provides deep insight and allows the evaluator to understand the implementation at a level comparable to its developers.

S4. Computer-aided analysis

Task. The evaluator shall apply static and/or dynamic analysis tools to identify implementation flaws.

Example. The evaluator used Wycheproof [13] and Cryptofuzz [14] to check the Ed25519 implementation. Then he applied a generic C++ static analyzer to detect unsafe coding practices.

Remarks. Static analysis does not require execution of the evaluated codebase and includes tools like Wycheproof [13]. Dynamic analysis is performed during execution and includes fuzzers and tools such as Valgrind [15] to detect runtime issues like memory leaks.

S5. Context validation

Task. The evaluator shall examine the interaction of the DCT with its external environment to identify context-related misuse or misconfiguration.

Example. The evaluator used the recursive `grep` command to list function calls and further analyzed the invoking context.

Remarks. Contextual validation ensures that even correctly implemented algorithms are not undermined by improper use. This step often goes beyond the scope of the DCT itself but is essential for assessing overall system security.

V. Case Study: ED25519 Implementation

To validate the implementation of the Ed25519 digital signature scheme [10], we constructed a knowledge base comprising a *Cryptographic Validation Card* and a *Cryptographic Vulnerability Reference* for each component shown in Figure 2. For brevity, Subsection V-A presents CVC of just one component - Ed25519 digital signature scheme. In the Subsection V-B we cover the concise record of the validation process. All future updates to the framework and its knowledge base will be made available in the project's GitHub repository at *https://github.com/arturmisztal/divcrypt*.

A. Cryptographic Validation Card for Ed25519

[CVC-1] Specifications.

- NIST FIPS 186-5 [12].
- RFC 8032 [16].
- EdDSA for more curves [17].
- High-speed high-security signatures [10].

[CVC-2] Algorithms.

- `ed25519-gen`
  - *description:* key generation
  - *input domain:* -
  - *output domain:* $\{0,1\}^{256} \times \{0,1\}^{256}$
- `ed25519-sig`
  - *description:* signature generation
  - *input domain:* message $\mathcal{M}$ of length less than 2^{128} bits
  - *output domain:* $\{0,1\}^{512}$
- `ed25519-ver`
  - *description:* signature verification
  - *input domain:* $\mathcal{M} \times \{0,1\}^{512} \times \{0,1\}^{256}$
  - *output domain:* $\{$`invalid, valid`$\}$

[CVC-3] Dependencies (Table I).

[CVC-4] Sources.

- NIST ACVP EdDSA [18].
- Ed25519 reference implementation [19].
- CRYPTREC Review of EdDSA [20].
- EdDSA signature verification edge cases [21].
- The Provable Security of Ed25519 [22].

[CVC-5] Official Test Procedures.[2]

- For the given set of messages and private keys, the signature is generated with `ed25519-sig` and the expected result is to be correctly verified by `ed25519-ver` [19].
- For the given set of messages and private keys, the signature is generated with `ed25519-sig`, incremented, and the expected result is that the forgery is detected by `ed25519-ver` [19].
- Ed25519 test vectors from RFC 8032 [16].
- EdDSA test procedures located in CCN-STIC 2100 [23].

[CVC-6] Other Test Procedures.[2]

- EdDSA-oriented adaptation of ECDSA test procedures located in SOG-IS Crypto Evaluation Scheme [6] and ISO/IEC 18367 [24].

[2]Although a standalone guide would describe the test procedures in details using pseudocode, this paper provides only a high-level overview due to space constraints.

TABLE I
LIST OF DIRECT `ED25519` DEPENDENCIES.

ALGORITHM	`ed25519-gen`	`ed25519-sig`	`ed25519-ver`
`curve25519-mul`	✓	✓	✓
`curve25519-add`	✗	✗	✓
`sha2-512`	✓	✓	✓
`fp-mul`	✗	✓	✗
`fp-add`	✗	✓	✗

B. Validation Process Record

1) The implementation under evaluation was identified as a portable C implementation of the Ed25519 digital signature scheme.
2) The DCT-graph for the Ed25519 implementation was constructed.
3) The implementation was investigated to ensure that each *Cryptographic Layer Object* (CLO) was uniquely represented. As the hash function was located in a separate file, two package families were created: [HSM.SIGN] and [HSM.CRYPTO] (see Figure 2).
4) The CVC and CVR artifacts were created for the following components:
   - prime field (`fp`),
   - elliptic curve Curve25519 (`curve25519`),
   - hash function SHA512 (`sha2-512`),
   - digital signature scheme Ed25519 (`ed25519`).

   In cases where CVC/CVR artifacts already existed, they only needed to be updated.
5) The five DIVCRYPT steps were performed sequentially for each *Cryptographic Layer Object Implementation* (CLOI). Test procedures (Step S2) were conducted using the Python programming language, while SageMath [25] was used to generate reference test vectors for `fp` and `curve25519`. Computer-aided analysis (Step S4) was performed using the general-purpose static code analysis tool CodeChecker [26].
6) As the evaluation of each CLOI was successful, the final verdict was PASS.

VI. COMPARISION WITH EXISTING STANDARDS

In the domain of systems security, standards can generally be classified into three categories: *theoretical components*, *evaluation criteria*, and *testing guidance*. When narrowing the focus specifically to cryptographic mechanisms, one can readily identify numerous specifications published by National Institute of Standards and Technology [27], whether these define a family of cryptographic solutions [12] or detail a single standardized scheme [7]. In addition, there exist global [3], regional [4] and national [28] standards that specify how and which schemes should be deployed. Unfortunately, when it comes to testing guidance, such documents are almost always tailored to specific schemes within particular application domains. Existing standards that describe testing methodologies for cryptographic implementations are summarized in Table II. According to our case study, we provide a brief overview of the procedures dedicated to the Ed25519 digital signature scheme recommended by each standard.

*a) **ISO/IEC 24759 [3]:*** This document defines the specific test procedures that a laboratory must use to check a module's conformance to the requirements in ISO/IEC 19790 [5]. For digital signatures, its procedures ensure a module can perform critical self-tests, such as the pair-wise consistency test upon key generation and various KATs.

TABLE II
A SELECTION OF ENGLISH-LANGUAGE STANDARDS FOR TESTING CRYPTOGRAPHIC IMPLEMENTATIONS.

YEAR	TITLE	SCOPE
2025	*ISO/IEC 24759: Test requirements for cryptographic modules* [5]	Global
2025	*CCN-STIC 2100: Cryptographic Mechanisms Evaluation Methodology* [23]	Spain
2023	*AEPD Guidelines for the validation of cryptographic systems in data protection processing* [29]	Spain
2022	*EN 17640: Fixed-time cybersecurity evaluation methodology for ICT products (Sections 6.11 and 6.12)* [30]	Europe
2020	*SOG-IS Crypto Evaluation Scheme: Harmonised Cryptographic Evaluation Procedures* [6]	Europe
2019	*FIPS 140-3: Security Requirements for Cryptographic Modules* [31]	USA
2016	*ISO/IEC 18367: Information technology - Security techniques - Cryptographic algorithms and security mechanisms conformance testing* [24]	Global
2015	*ANSSI-CC-CRY-P01: Methods For Carrying Out Cryptographic Analysis And Random Number Evaluations* [32]	France

b) ***CCN-STIC 2100 [23]****:* This document from Spain's National Cryptologic Center is a highly detailed and prescriptive methodology for evaluating specific, CCN-agreed cryptographic mechanisms against defined certification levels. It provides an exhaustive list of conformity tests for EdDSA (see Table 66 in [23]), including specific Validation Tests for signature generation and KATs for signature verification.

c) ***AEPD [29]****:* These guidelines from the Spanish Data Protection Agency provide a framework for assessing cryptographic systems, specifically to ensure they comply with GDPR [33] and effectively protect personal data. Its focus is on the governance and risk management surrounding the cryptosystem rather than detailing technical verification procedures for specific algorithms like EdDSA.

d) ***EN 17640 [30]****:* EN 17640 defines a modular evaluation methodology for ICT products designed to be performed within a fixed time budget, including tasks for basic and extended analysis of cryptographic mechanisms. Its recommendations cover conformance testing with KATs and source code analysis to check for implementation errors.

e) ***SOG-IS [6]****:* These harmonized procedures detail the evaluation tasks for cryptographic mechanisms within the formal SOG-IS scheme, which is used for Common Criteria evaluations in Europe. Crucially, this document defines a complete suite of evaluation tasks specifically for ECDSA-like digital signatures, including conformity testing and analysis of implementation pitfalls related to the underlying elliptic curve cryptography.

f) ***FIPS 140-3 [31]****:* This standard specifies the security requirements for a cryptographic module as a whole, covering areas from physical security to roles and services. It delegates the specific test procedures for approved algorithms like EdDSA to related standards.

g) ***ISO/IEC 18367 [24]****:* This standard provides guidelines for black-box and white-box conformance testing to ensure a cryptographic implementation correctly adheres to its specification. ISO/IEC 18367:2016 is based on conformance testing methods employed by the Japan Cryptographic Module Validation Program (JCMVP) and the NIST Cryptographic Algorithm Validation Program (CAVP).

h) ***ANSSI-CC-CRY-P01 [32]****:* This French procedure defines the formal roles and responsibilities for how approved evaluation labs must conduct cryptographic analyses for the national certification scheme, distinguishing between theoretical and implementation analysis. It specifies that an evaluator should perform analysis of conformity and vulnerability in cryptography implementation.

Many standards such as CCN-STIC 2100 [23], EN 17640 [30] or SOG-IS procedures [6] cover important topics related to cryptographic scheme sub-primitives, implementation pitfalls and usage context. However, a universal, formalized methodology that is equally applicable to both a simple stream cipher like ChaCha20 [34] and a complex non-interactive zero-knowledge protocol zk-STARK [35] is currently lacking, given their significant differences in cryptographic structure, complexity, and maturity. What these diverse cryptographic constructs do share is the need to address four fundamental aspects of implementation security, which DIVCRYPT covers through five dedicated actions (see Figure 1).

VII. CONCLUSION AND FUTURE WORK

This work introduced DIVCRYPT, a structured and layered framework for validating cryptographic implementations. By decomposing cryptographic solutions into layered components and evaluating each through structured steps, DIVCRYPT ensures that both standardized and emerging cryptographic mechanisms are subjected to thorough and context-aware analysis. The framework balances flexibility with rigor, offering evaluators a practical methodology that is both scalable and reusable across implementations. DIVCRYPT addresses a critical gap in current evaluation practices by introducing a reusable knowledge base (CVC and CVR), a clear abstraction hierarchy (CLO, CLOI and DCT-graph), and a comprehensive five-step validation process that covers theoretical design, functional correctness, code-level assurance, computer-aided analysis, and contextual robustness.

Future work should focus on several key directions. First, initiating a broader discussion with the community is essential to evaluate the advantages and limitations of the DIVCRYPT framework. Second, establishing a publicly accessible database of CVC and CVR entries would significantly support shared validation efforts and improve reproducibility. Finally, incorporating formal verification methods potentially as an optional step should be considered, as this would further strengthen the rigor and reliability of the validation process where applicable.

Acknowledgment

The author would like to thank Paweł Topa (AGH University) and Dariusz Rogowski (Łukasiewicz – AI) for critically reviewing the manuscript, as well as the anonymous reviewers for their constructive feedback.

References

[1] International Organization for Standardization, "ISO/IEC 15408: Information security, cybersecurity and privacy protection - Evaluation criteria for IT security," International Organization for Standardization, Tech. Rep., 2022. [Online]. Available: https://www.commoncriteriaportal.org

[2] International Society of Automation, "ISA/IEC 62443: Security for industrial automation and control systems," IEC Central Office, Tech. Rep., 2019. [Online]. Available: https://www.isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-standards

[3] International Organization for Standardization, "ISO/IEC 19790: Security Requirements for Cryptographic Modules," International Organization for Standardization, Tech. Rep. ISO/IEC 19790, 2025. [Online]. Available: https://www.iso.org/standard/82423.html

[4] SOG-IS Crypto Working Group, "Crypto Evaluation Scheme: Agreed Cryptographic Mechanisms," Senior Official Group Information Security Systems, Tech. Rep., 2023. [Online]. Available: https://www.sogis.eu/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.3.pdf

[5] International Organization for Standardization, "ISO/IEC 24759: Test Requirements for Cryptographic Modules," International Organization for Standardization, Tech. Rep. ISO/IEC 24759, 2025. [Online]. Available: https://www.iso.org/standard/82424.html

[6] SOG-IS Crypto Working Group, "Crypto Evaluation Scheme: Harmonised Cryptographic Evaluation Procedures," Senior Official Group Information Security Systems, Tech. Rep., 2020. [Online]. Available: https://www.sogis.eu/documents/cc/crypto/202203-hep-draft16.pdf

[7] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST FIPS 203, Aug. 2024. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf

[8] ——, "Module-Lattice-Based Digital Signature Standard," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST FIPS 204, Aug. 2024. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf

[9] "ZKDocs." [Online]. Available: https://www.zkdocs.com/

[10] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of Cryptographic Engineering*, vol. 2, no. 2, pp. 77–89, Sep. 2012. [Online]. Available: http://link.springer.com/10.1007/s13389-012-0027-1

[11] D. Rogowski, "Software Implementation of Common Criteria Related Design Patterns," in *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems*, M. P. M. Ganzha, L. Maciaszek, Ed. Kraków: IEEE, 2013, pp. pages 1135–1140. [Online]. Available: https://annals-csis.org/Volume_1/pliks/210.pdf

[12] L. Chen, D. Moody, A. Regenscheid, and A. Robinson, "NIST FIPS 186-5: Digital Signature Standard (DSS)," National Institute of Standards and Technology (U.S.), Gaithersburg, MD, Tech. Rep. NIST FIPS 186-5, Feb. 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf

[13] "Wycheproof," May 2025, original-date: 2016-11-08T20:56:25Z. [Online]. Available: https://github.com/C2SP/wycheproof

[14] "Cryptofuzz," Jun. 2025, original-date: 2024-11-25T19:31:28Z. [Online]. Available: https://github.com/MozillaSecurity/cryptofuzz

[15] "Valgrind." [Online]. Available: https://valgrind.org/

[16] S. Josefsson and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)," RFC Editor, Tech. Rep. RFC8032, Jan. 2017, issue: 8032 Num Pages: 60 Series: Request for Comments Published: RFC 8032. [Online]. Available: https://www.rfc-editor.org/info/rfc8032

[17] D. J. Bernstein, S. Josefsson, T. Lange, P. Schwabe, and B.-Y. Yang, "EdDSA for more curves," 2015, published: Cryptology ePrint Archive, Paper 2015/677. [Online]. Available: https://eprint.iacr.org/2015/677

[18] "ACVP EdDSA Algorithm JSON Specification." [Online]. Available: https://pages.nist.gov/ACVP/draft-celi-acvp-eddsa.html

[19] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "Ed25519 software." [Online]. Available: https://ed25519.cr.yp.to/software.html

[20] S. D. Galbraith, "CRYPTREC Review of EdDSA," Mathematics Department, University of Auckland, Auckland, New Zealand, Tech. Rep. CRYPTREC EX-3003-2020, 2020. [Online]. Available: https://www.cryptrec.go.jp/exreport/cryptrec-ex-3003-2020.pdf

[21] "novifinancial/ed25519-speccheck," Apr. 2025, original-date: 2020-07-28T21:44:25Z. [Online]. Available: https://github.com/novifinancial/ed25519-speccheck

[22] J. Brendel, C. Cremers, D. Jackson, and M. Zhao, "The Provable Security of Ed25519: Theory and Practice," in *2021 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA, USA: IEEE, May 2021, pp. 1659–1676. [Online]. Available: https://ieeexplore.ieee.org/document/9519456/

[23] Centro Criptologico Nacional, "CCN-STIC 2100: Cryptographic Mechanisms Evaluation Methodology," 2025. [Online]. Available: https://oc.ccn.cni.es/en/types-of-certification/crytpologic-certification/criteria-and-methodologies

[24] International Organization for Standardization, "ISO/IEC 18367: Information technology - Security techniques - Cryptographic algorithms and security mechanisms conformance testing," International Organization for Standardization, Tech. Rep., 2016. [Online]. Available: https://www.iso.org/standard/62286.html

[25] "SageMath," Jun. 2025. [Online]. Available: https://github.com/sagemath/sage

[26] "CodeChecker," Jul. 2025. [Online]. Available: https://github.com/Ericsson/codechecker

[27] "National Institute of Standards and Technology," Apr. 2025, last Modified: 2025-06-04T08:37-04:00. [Online]. Available: https://www.nist.gov/

[28] Federal Office for Information Security, "BSI TR-02102-1: Cryptographic Mechanisms - Recommendations and Key Lengths," 2025. [Online]. Available: https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr02102/tr02102_node.html

[29] Spanish Data Protection Agency, "Guidelines for the validation of cryptographic systems in data protection processing," 2023. [Online]. Available: https://www.aepd.es/guides/guidelines-validation-cryptographic-systems-data-protection-processing.pdf

[30] The European Committee for Standardization, "EN 17640: Fixed-time cybersecurity evaluation methodology for ICT products," CEN and CENELEC, Tech. Rep. EN 17640, 2022.

[31] National Institute of Standards and Technology, "NIST FIPS 140-3: Security Requirements for Cryptographic Modules," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST FIPS 140-3, Apr. 2019. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf

[32] French Cybersecurity Agency, "Methods For Carrying Out Cryptographic Analysis And Random Number Evaluations," 2015. [Online]. Available: https://cyber.gouv.fr/sites/default/files/2022-08/ANSSI-CC-CRY-P-01-Modalites-pour-la-realisation-des-analyses-crypto_v003_EN%5B3%5D.pdf

[33] "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)," 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A02016R0679-20160504

[34] D. J. Bernstein, "ChaCha, a variant of Salsa20," 2008. [Online]. Available: https://cr.yp.to/chacha/chacha-20080128.pdf

[35] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," Cryptology {ePrint} Archive, Tech. Rep. 2018/046, 2018. [Online]. Available: https://eprint.iacr.org/2018/046