ORIGINAL ARTICLE

INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE

ISSN: 2692-5206, Impact Factor: 12,23
American Academic publishers, volume 05, issue 08,2025

Journal: https://www.academicpublishers.org/journals/index.php/ijai

SELECTING CRYPTOGRAPHIC SIGNATURE ALGORITHMS TO ENSURE SMART CONTRACT SECURITY

Raimov Ulugbek Yorqinbek ugli
Teacher at Andijan State Technical Institute
E-mail: uraimov0111@gmail.com
ORCID: 0009-0009-9304-5980

Abstract: Blockchain platforms based on smart contracts rely on cryptographic signature algorithms to verify user actions and provide protection against malicious attacks. Selecting the right algorithm is one of the most important factors in ensuring smart contract security, as cryptography is constantly evolving and new technologies may affect the reliability of existing algorithms. In particular, the development of quantum computers poses a threat to widely used digital signature protocols. This paper proposes a flexible approach to selecting cryptographic signature algorithms to ensure the security of smart contracts. The approach involves managing algorithms and their parameters through special smart contracts deployed on the blockchain ledger, enabling the rapid introduction of new, secure signature methods or the deprecation of outdated ones. This method increases the adaptability of the blockchain to technological advancements, facilitates the adoption of post-quantum signature algorithms, and eliminates the fork problem that often arises during updates. The proposed approach is effective for real-world applications, including industry, complex supply chains, and IoT devices.

Keywords: blockchain; smart contracts; cryptographic signature algorithms; distributed ledger; post-quantum cryptography; quantum-resistant signatures; digital signatures; cybersecurity

Introduction: Smart-contract-based blockchain platforms rely heavily on cryptographic signature algorithms to verify the validity of user transactions and to protect the network from malicious actors. Two fundamental requirements underpin this security model:
Immutability of the ledger — once consensus is reached, there must not be multiple conflicting versions of the same ledger. Protection of data ownership — only the legitimate owner of a digital asset or record should be able to manage it. Most consensus protocols depend on cryptographically secure algorithms to enforce these properties. For example, Proof-of-Work protocols such as Hashcash rely primarily on cryptographic hash functions, while Proof-of-Stake protocols often depend on digital signature schemes. However, the security of cryptographic primitives is neither absolute nor permanent. Advances in computing power or cryptanalysis can expose previously unknown vulnerabilities. This is especially critical with the advent of quantum computing, which poses a serious threat to widely used algorithms like ECDSA and EdDSA. Shor's algorithm, in particular, can break such schemes in relatively short timeframes, jeopardizing the security of blockchain systems that store sensitive financial data and execute smart contracts [1].
Background: Currently, the most widely used digital signature schemes in blockchain networks — ECDSA and EdDSA — rely on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP). In the classical computing model, ECDLP is considered intractable for sufficiently large parameters. However, quantum algorithms, most notably Shor's algorithm, can solve this problem efficiently, undermining the security foundations of these schemes.

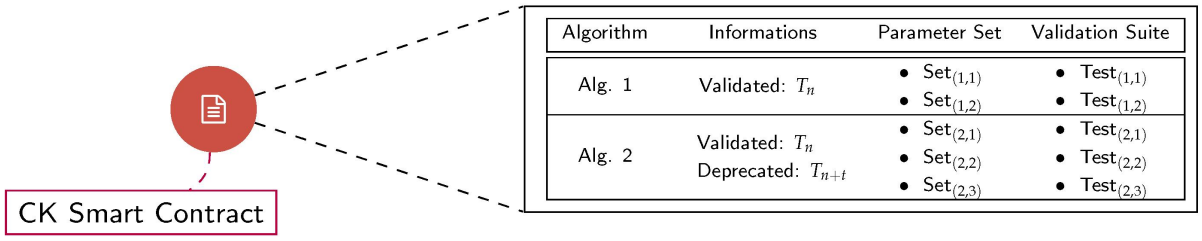
ORIGINAL ARTICLE

INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE

ISSN: 2692-5206, Impact Factor: 12,23
American Academic publishers, volume 05, issue 08,2025

Journal: https://www.academicpublishers.org/journals/index.php/ijai

This growing risk has accelerated the need for post-quantum (quantum-resistant) digital signature algorithms. In response, the U.S. National Institute of Standards and Technology (NIST) has initiated a standardization process to identify and approve new public-key cryptographic schemes capable of withstanding quantum attacks. The most promising categories include: Lattice-based: e.g., CRYSTALS-Dilithium, Falcon, Multivariate cryptography: e.g., Rainbow, GeMSS
Hash-based: e.g., SPHINCS+ While these schemes offer quantum resistance, their efficiency, scalability, and suitability for smart contract environments vary significantly. Therefore, selecting the right cryptographic signature algorithm is a critical step in ensuring both security and performance for blockchain-based smart contracts[2][3].
Structure of the CK Smart Contract. The Cryptographic Kernel (CK) framework allows blockchain systems to use any signature algorithm from a dynamic, non-static pre-approved list. This list begins with an initial selection of algorithms but can be expanded or updated as new cryptographic developments arise. Each algorithm in the CK smart contract is identified by a unique string identifier. This identifier is embedded in every digital signature, enabling verifiers to know exactly which cryptographic algorithm was used for signing.For each listed algorithm, the CK smart contract maintains:
A set of parameter configurations — allowing different security and performance trade-offs.
A validation suite — a collection of tests to verify correctness and security compliance.
Two status fields:
Validated — marked with a timestamp when the algorithm is officially added to the list. Deprecated — marked with a timestamp if the algorithm is later deemed insecure, warning users that it is no longer recommended.This structure allows the blockchain governance committee to react quickly to new cryptographic threats. If a vulnerability is discovered, the affected algorithm can be flagged as deprecated, and a more secure alternative can be added to the list without disrupting existing smart contract functionality. A schematic representation of the CK smart contract is illustrated in **Figure 1[4]**.



the
sm
verifiers and developers can reproduce the exact conditions under which a signature was generated[5][6].
Recognizing that users' computational resources vary widely, the CK framework classifies algorithm parameters into three security-performance tiers:

- Basic — optimized for low-power devices or legacy systems, prioritizing efficiency over maximum security.
- Intermediate — balanced between security strength and computational cost, suitable for general-purpose blockchain applications.
- Enhanced — designed for maximum security, ideal for protecting sensitive or long-term data.

For each tier, the blockchain governance committee defines a standard parameter set that represents the recommended configuration. However, the CK design also allows users to

ORIGINAL ARTICLE
INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE
ISSN: 2692-5206, Impact Factor: 12,23
American Academic publishers, volume 05, issue 08,2025
Journal: https://www.academicpublishers.org/journals/index.php/ijai

propose their own parameter sets for inclusion. These proposals can address specialized needs — for instance, in ECDSA, a user might suggest using a custom elliptic curve or a different base point. Every parameter set in the CK contract is accompanied by: A formal security assessment — documenting the security guarantees achieved. A timestamp — recording when the assessment was conducted.

If later changes in cryptographic testing methods alter the security evaluation, the CK smart contract appends a new assessment record with its own timestamp. The most recent assessment should be used for current operations, while historical data should always be validated against the security information that was available at the time of signature creation. Each algorithm's entry also includes a validation suite — a set of tests for verifying user-proposed parameters. This suite is maintained and updated by the committee in response to new cryptographic discoveries. Updates may involve adding new tests, replacing outdated ones, or modifying existing procedures. Every change to the validation suite is logged with a timestamp to preserve the blockchain's historical integrity. Whenever the validation suite is updated, all previously approved parameter sets must be re-tested using the new procedures. Their associated security levels are then revised accordingly [7-8].

Example:

- At time TkT_kTk, a parameter set is added and rated as Intermediate security.
- At time Tk+t1T_k + t_1Tk+t1, a new test is introduced, reducing its rating to Basic.
- At time Tk+t1+t2T_k + t_1 + t_2Tk+t1+t2, further testing determines that the parameter set is no longer secure, and it is marked as Deprecated.

This lifecycle process, illustrated schematically in Table 1, ensures that all cryptographic signature parameters used in smart contracts remain aligned with current security standards [9].

Managing Algorithm States and the Signing Process

An alternative approach to handling valid and deprecated cryptographic elements — including algorithms, parameter sets, and validation tests — is to maintain two distinct lists for each category: Active list — containing currently valid items [10].

.Deprecated list — containing items that are no longer recommended. Whenever an entry is moved from one list to the other (e.g., from active to deprecated), the change is recorded with the current timestamp, preserving a verifiable history of status changes. Signing and Verification Workflow In blockchain environments, users frequently need to sign transactions or blocks. Under the CK smart contract framework, the process follows these steps: Algorithm Selection.The user selects a cryptographic signature algorithm from those defined in the CK smart contract. The system checks that the chosen algorithm is not marked as deprecated.Security Level Choice The user specifies the desired security tier (Basic, Intermediate, or Enhanced).They can either: Select an existing parameter set associated with that tier, or Propose a new parameter set for evaluation.Key Generation. Based on the selected parameters, the user generates a public–private key pair, or uses a previously generated pair compatible with those parameters. Data Signing. The data is signed according to the selected algorithm and parameter set. The resulting signed object has the structure:sql КопироватьРедактировать algorithm_identifier || parameter_set || (timestamp) || signature Here, || denotes byte concatenation. The timestamp is optional because temporal context can also be inferred from the blockchain itself (e.g., the block's own timestamp). Verification. During verification, the algorithm and parameter set are extracted from the signature.The

ORIGINAL ARTICLE

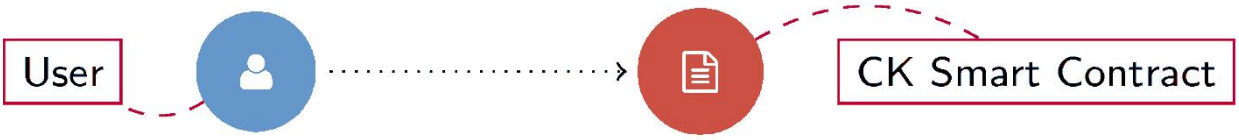
INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE

ISSN: 2692-5206, Impact Factor: 12,23
American Academic publishers, volume 05, issue 08,2025

Journal: https://www.academicpublishers.org/journals/index.php/ijai

verifier ensures they are still valid or, in the case of historical data, were valid at the time of signing.

Figure 2 illustrates the workflow when an existing parameter set is chosen. The case for user-proposed parameters is detailed in Section 2.3[11].



Conclusion: Ensuring the security of smart contracts in blockchain ecosystems requires a flexible and adaptive approach to cryptographic signature algorithm selection. The proposed CK smart contract framework addresses this need by enabling on-chain management of algorithms, parameter sets, and validation tests, with mechanisms for real-time updates, deprecation tracking, and timestamp-based historical validation. By supporting multiple security levels and allowing both committee-approved and user-proposed parameter sets, this model enhances the blockchain's ability to adapt to emerging cryptographic threats, including those posed by quantum computing. Moreover, the use of smart contracts for cryptographic governance eliminates the need for disruptive hard forks, ensuring that security updates and protocol changes can be integrated seamlessly without compromising existing data. This approach not only strengthens the resilience of smart contract platforms but also provides a scalable foundation for deployment across industrial systems, supply chain networks, and IoT devices, where security and adaptability are equally critica[11].

References:

1. Jakobsson, M.; Juels, A. Proofs of work and bread pudding protocols. In *Secure Information Networks*; Springer: Boston, MA, USA, 1999; pp. 258–272. [Google Scholar]
2. King, S.; Nadal, S. Ppcoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. Self-Published Paper. 2012, Volume 19. Available online: **https://bitcoin.peryaudo.org/vendor/peercoin-paper.pdf** (accessed on 20 June 2022).
3. Rivest, R.L.; Shamir, A.; Adleman, L.M. *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*; Routledge: London, UK, 2019. [Google Scholar]
4. Gilles, B.; Hoyer, P.; Tapp, A. Quantum algorithm for the collision problem. *arXiv* **1997**, arXiv:quant-ph/9705002. [Google Scholar]
5. McCurley, K.S. The discrete logarithm problem. In Proceedings of the Symposia in Applied Mathematics, Boulder, CO, USA, 6–7 August 1989; Volume 42, pp. 49–74. [Google Scholar]
6. Rabah, K. Security of the Cryptographic Protocols Based on Discrete Logarithm Problem. *J. Appl. Sci.* **2005**, *5*, 1692–1712. [Google Scholar] [CrossRef] [Green Version]
7. Galbraith, S.D.; Gaudry, P. Recent progress on the elliptic curve discrete logarithm problem. *Des. Codes Cryptogr.* **2016**, *78*, 51–72. [Google Scholar] [CrossRef]
8. Kerry, C.F.; Director, C. FIPS PUB 186-4 Federal Information Processing Standards Publication Digital Signature Standard (DSS). 2013. Available online: **http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.362.5590** (accessed on 20 June 2022).

ISSN: 2692-5206, Impact Factor: 12,23
American Academic publishers, volume 05, issue 08,2025
Journal: https://www.academicpublishers.org/journals/index.php/ijai

9. Josefsson, S.; Liusvaara, I. Edwards-Curve Digital Signature Algorithm (EdDSA). In Proceedings of the Internet Research Task Force, Crypto Forum Research Group, RFC; 2017; Volume 8032. Available online: **https://www.rfc-editor.org/rfc/rfc8032.html** (accessed on 20 June 2022).
10. Shor, P.W. Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings of the Proceedings 35th annual symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; pp. 124–134. [**Google Scholar**]
11. National Institute of Standards and Technology. Post-Quantum Cryptography Standardization—Post-Quantum Cryptography. Available online: **https://csrc.nist.gov/Projects/post-quantum-cryptography** (accessed on 28 June 2022).